*Приложение 2.24.1к ОПОП по специальности 09.02.13 Интеграция решений с применением технологий искусственного интеллекта*

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»

Многопрофильный колледж

# МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ УЧЕБНОЙ ДИСЦИПЛИНЫ

# ОП.02 ДИСКРЕТНАЯ МАТЕМАТИКА С ЭЛЕМЕНТАМИ МАТЕМАТИЧЕСКОЙ ЛОГИКИ

**для обучающихся специальности**

**09.02.13 Интеграция решений с применением технологий искусственного интеллекта**

**Магнитогорск, 2025**

**ОДОБРЕНО**

Предметно-цикловой комиссией «Информатики и вычислительной техники»
Председатель Т.Б. Ремез
Протокол № 5.1 от «17» февраля 2025г.

Методической комиссией МпК

Протокол № 3 от «19» февраля 2025г.

*Разработчики:*
преподаватель отделения №2 «Информационных технологий и транспорта»
Многопрофильного колледжа ФГБОУ ВО «МГТУ им. Г.И. Носова»
Елена Александровна Васильева
преподаватель отделения №2 «Информационных технологий и транспорта»
Многопрофильного колледжа ФГБОУ ВО «МГТУ им. Г.И. Носова»
Светлана Владимировна Меркулова

Методические указания по выполнению практических работ разработаны на основе рабочей программы учебной дисциплины «Дискретная математика с элементами математической логики».

Содержание практических работ ориентировано на подготовку обучающихся к освоению профессиональных модулей программы подготовки специалистов среднего звена по специальности 09.02.13 Интеграция решений с применением технологий искусственного интеллекта и овладению общими компетенциями.

**СОДЕРЖАНИЕ**

1 ВВЕДЕНИЕ ..... 4
2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ ..... 5
Практическое занятие № 1. Операции над множествами: объединение, пересечение, дополнение ..... 5
Практическое занятие № 2. Построение диаграмм Венна для множества и подмножества ..... 5
Практическое занятие № 3. Кардинальные числа: вычисление мощностей множеств ..... 11
Практическое занятие № 4. Построение таблиц истинности для логических операций (AND, OR, NOT) ..... 13
Практическое занятие № 5. Применение законов булевой алгебры для упрощения логических выражений ..... 13
Практическое занятие № 6. Применение булевой алгебры в программировании: реализация логических операций в коде ..... 16
Практическое занятие № 7. Оценка времени выполнения алгоритмов: вычисление сложности O(n) ..... 19
Практическое занятие № 8. Реализация и анализ базовых структур данных: массивы, списки, очереди, деревья ..... 21
Практическое занятие № 9. Построение и анализ графов в представлении "список смежности" ..... 25
Практическое занятие № 10. Реализация алгоритмов сортировки: пузырьковая сортировка, быстрая сортировка, сортировка слиянием ..... 28
Практическое занятие № 11. Сравнение времени выполнения различных алгоритмов сортировки ..... 31
Практическое занятие № 12. Реализация и анализ линейного и бинарного поиска в массивах ..... 34
Практическое занятие № 13. Построение таблиц истинности для логических высказываний 37
Практическое занятие № 14. Формализация предикатов для описания условий в задачах анализа данных ..... 40
Практическое занятие № 15. Применение предикатов в программировании для обработки данных ..... 40
Задания для самостоятельного решения ..... 43
Практическое занятие № 16. Построение графов: ориентированные и неориентированные графы ..... 45
Практическое занятие № 17. Реализация алгоритмов поиска в глубину (DFS) и поиска в ширину (BFS) на графах ..... 49
Практическое занятие № 18. Применение графов для моделирования реальных сетей и анализа данных ..... 52
Практическое занятие № 19. Решение задач на перестановки, сочетания и размещения ..... 55
Практическое занятие № 20. Применение формул комбинаторики для анализа данных. ..... 55
Практическое занятие № 21. Построение деревьев решений с использованием комбинаторных методов ..... 59

# 1 ВВЕДЕНИЕ

Важную часть теоретической и профессиональной практической подготовки обучающихся составляют практические занятия.

Состав и содержание практических занятий направлены на реализацию Федерального государственного образовательного стандарта среднего профессионального образования.

Ведущей дидактической целью практических занятий является формирование профессиональных практических умений.

В соответствии с рабочей программой учебной дисциплины «ОП.02 Дискретная математика с элементами математической логики» предусмотрено проведение практических занятий.

В результате их выполнения, обучающийся должен:

***уметь:***

У.1 Применять логические операции, формулы логики, законы алгебры логики.

У.2 Формулировать задачи логического характера и применять средства математической логики для их решения.

Содержание практических ориентировано на подготовку обучающихся к освоению профессиональных модулей ПМ 01 Разработка кода для искусственного интеллекта, ПМ 03 Обучение готовых моделей искусственного интеллекта программы подготовки специалистов среднего звена по специальности и овладению ***профессиональными и общими компетенциями:***

ПК1.1 Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.

ПК3.3 Проводить обучение и последующую калибровку готовых моделей искусственного интеллекта.

ОК 1 Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

Выполнение обучающихся практических работ по учебной дисциплине «ОП.02 Дискретная математика с элементами математической логики» направлено на:

- *обобщение, систематизацию, углубление, закрепление, развитие и детализацию полученных теоретических знаний по конкретным темам учебной дисциплины;*

- *формирование умений применять полученные знания на практике, реализацию единства интеллектуальной и практической деятельности;*

Практические занятия проводятся после соответствующей темы, которая обеспечивает наличие знаний, необходимых для ее выполнения.

## 2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

### Тема 1.1. Множества и операции над ними

### Практическое занятие № 1. Операции над множествами: объединение, пересечение, дополнение
### Практическое занятие № 2. Построение диаграмм Венна для множества и подмножества

**Цель работы:** формирование умений выполнять операции над множествами.

**Выполнив работу, Вы будете:**

***уметь:***

- строить и анализировать дискретные модели
- выполнять операции над множествами;
- решать задачи на выполнение теоретико-множественных операций.

**Материальное обеспечение:**

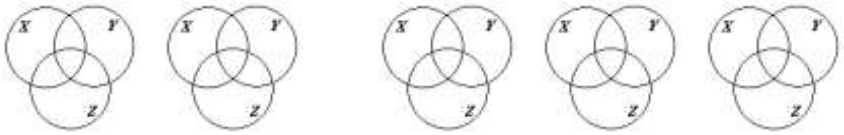
Раздаточный материал (карточки с заданиями).

Технические средства обучения: компьютер с лицензионным программным обеспечением и мультимедиа проектор.

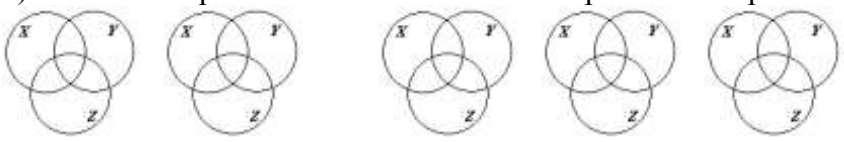
**Задания:**

1. Пусть даны множества *A={-3;-2;-1;0;1;2;3;7}, B={5;3;2;1;0;-2;-3}, C={-4;-3;-2;-1;0;1;2;3;4}*. Найдите множества $A\cup B$, $A\cap B$, $A\cup C$, $B\cup C$ $A\setminus B$, $B\setminus A$.
2. Докажите следующие тождества: а) $(X\cap Y)\cup Z=(X\cup Z)\cap(Y\cup Z)$; б) $(X\setminus Y)\cup Z=(X\setminus Y)\cap(Y\setminus Z)$.

а) левая часть равенства правая часть равенства



б) левая часть равенства правая часть равенства



3. Пусть *N*- множество натуральных чисел, *Z*- множество целых чисел, а множества *A, B*, и *C* определены в задании 1. Найдите $A\cup N$, $A\cap N$, $Z\cup C$, $(A\cap B)\cap N$, $A\setminus Z$.
4. Пусть *A* – множество параллелограммов, *B* – множество прямоугольников, *C* – множество ромбов, *D* – множество квадратов. Запишите результат операций: $A\cap B,\ B\cap C,\ A\cup B\cup C\cup D.$
5. Укажите пустые множества среди следующих: а) множество целых корней уравнения $x^2-16=0$; б) множество целых корней уравнения $x^2+16=0$.
6. Изобразите с помощью диаграмм Эйлера- Венна множества *A, B, C*, если: а) $A\subset B, B\subset C$; б) $A\subset B,\ B\subset C,\ A\setminus B=\varnothing$; в) $A\cup B,\ \ B\cap C,\ \ A\subset B$; г) $A\cap B\neq\varnothing,\ \ A\cap C\neq\varnothing,\ \ B\cap C\neq\varnothing,\ \ A\cap B\cap C\neq\varnothing$.
7. Даны множества $A=\{x\in R\,|\,x^2+4=y\}$, $B=\{x\in R\,|\,x^2+y^2\le 9\}$, $C=\{x\in R\,|\,x+2\le y\}$. Найдите $A\cup B,\ A\cap B,\ A\cup C,\ A\cap C,\ B\cap C, A\setminus C, B\setminus A$.
8. Из цифр 1,2,3,4,5 составьте все двузначные числа. Как связано получившееся множество с декартовым произведением $A\times A,\ \ A=\{1,2,3,4,5\}$.

**Краткие теоретические сведения**

Одним из основных исходных понятий математики является понятие множества и его элементов. Множество состоит из элементов. Множества обозначаются большими латинскими буквами: A; B; C..., а их элементы - малыми буквами: a,b,c,…

Если a является элементом множества A или, что то же самое, a принадлежит множеству A, то применяют запись $a \in A$; в противном случае пишут $a \notin A$.

*Два множества A и B равны* (A=B), если они состоят из одних и тех же элементов. Если множества A и B не равны, то применяется запись $A \neq B$.

Множество, содержащее конечное число элементов, называется конечным, в противном случае множество называется бесконечным. Конечное множество, содержащее n элементов, называется n-множеством.

Множество, не содержащее элементов, называется пустым и обозначается $\varnothing$. Предположим, что все множества, являются подмножествами некоторого множества U, называемого универсальным множеством.

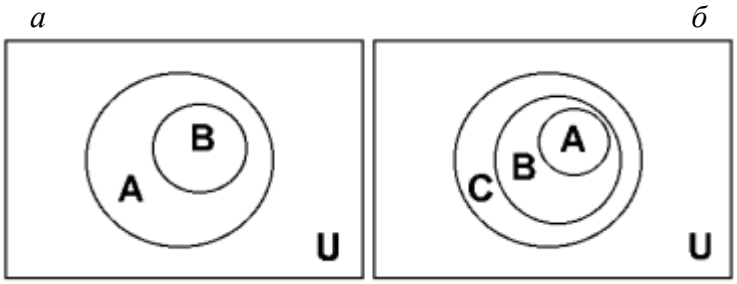



Рис 1.1

Если каждый элемент a множества B, $a \in B$, является элементом множества A, $a \in A$, то B называется подмножеством множества A (рис. 1.1, а). Этот факт записывается с помощью знака включения $\subseteq$ следующим образом: $B \subseteq A$.

Свойства включения

1. $A \subseteq A$;
2. если $A \subseteq B$ и $B \subseteq C$, то $A \subseteq C$ (рис. 1.1, б);
3. из двух включений $B \subseteq A$ и $A \subseteq B$ следует, что A=B.

Принято считать, что пустое множество является подмножеством любого множества.

Если $B \subseteq A$ и при этом $B \neq A$, то этому соответствует запись $B \subset A$ и B называется собственным подмножеством A.

Для описания множества A, состоящего из элементов a1,a2,...,an,... обычно применяется запись A={a1,a2,...,an,...}, причём порядок элементов в фигурных скобках не имеет значения; обычно он определяется соображениями наглядности.

Пример. В записи множества первых n натуральных чисел Nn={1,2,...,n} удобно располагать числа в возрастающем порядке, хотя при этом надо иметь в виду, что

N3 ={1,2,3}={2,1,3}={3,2,1}.

Другой способ задания множества состоит в описании свойств, однозначно определяющих принадлежность элементов данному множеству. Такому способу задания множества соответствует запись:

A ={a/a обладает свойством P(a)}.

Пример. Множество чётных чисел M может быть задано так:

M={i / i - целое число, которое делится на 2 без остатка}.

В случае описания множества с помощью некоторого свойства необходимо следить за тем, чтобы каждый элемент был чётко определён. Так, например, недостаточно чётким является определение множества A как множества слов русского языка, если нет ссылки на один из толковых словарей.

Возможно также рекурсивное задание множества, при котором осуществляется последовательное описание элементов через предыдущие. Например, множество натуральных чисел рекурсивно можно задать так:

$N=\{i$ / если целое $i \in N$, то $i+1 \in N$, $i \geq 1\}$.

*Операции над множествами*

*Объединением двух множеств A и B* называется множество вида:
$A \cup B = \{a / a \in A$ или $a \in B\}$(рис. 1.2, а).
Пересечением двух множеств A и B называется множество вида:
$A \cap B = \{a / a \in A$ и $a \in B\}$ (рис. 1.2, б).
Если множества A и B не имеют общих элементов, то $A \cap B = \varnothing$.

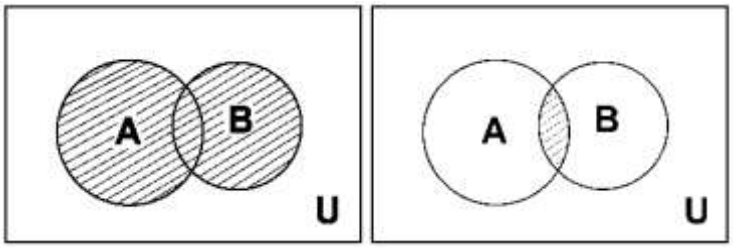


Рис 1.2

Свойства операций объединения и пересечения
*Коммутативность*:
1. $A \cup B = B \cup A$,
2. $A \cap B = B \cap A$;

*Ассоциативность*:
3. $(A \cup B) \cup C = A \cup (B \cup C)$,
4. $(A \cap B) \cap C = A \cap (B \cap C)$.

Объединение и пересечение связаны законами *дистрибутивности*:
5. $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$,
6. $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.

По свойству 3 операции включения следует равенство правой и левой частей доказываемого равенства.
Для операции объединения множеств нейтральным является пустое множество $\varnothing$, а для операции пересечения множеств - универсальное множество U.
*Разность множеств A и B* определяется следующим образом:
$A \setminus B = \{a / a \in A$ и $a \notin B\}$ (рис. 1.3, а).
Разность не обладает свойством коммутативности; эта операция также не является и ассоциативной.

Пользуясь понятием универсального множества, можно определить дополнение $\overline{A}$ к множеству A, как разность вида: $\overline{A} = U \setminus A$ (рис. 1.3, б).
Пример. Пусть в качестве универсального множества выступает множество целых чисел Z и пусть A - это множество всех чётных чисел. Тогда $\overline{A}$ - это множество всех нечётных чисел.
Операции объединения, пересечения и дополнения множеств связаны между собой законами де Моргана:

$$\overline{A \cap B} = \overline{A} \cup \overline{B},$$

$$\overline{A \cup B} = \overline{A} \cap \overline{B}.$$

Если $a \in \overline{A \cap B}$, то $a \notin A \cap B$. Это значит, что или $a \in \overline{A}$, или $a \in \overline{B}$, т.е. $a \in \overline{A} \cup \overline{B}$. Следовательно, $\overline{A \cap B} \subseteq \overline{A} \cup \overline{B}$.

С другой стороны, если $a \in \overline{A} \cup \overline{B}$, то или $a \in \overline{A}$, или $a \in \overline{B}$. Это значит, что $a \notin A \cap B$, т.е. $a \in \overline{A \cap B}$. Таким образом, $\overline{A} \cup \overline{B} \subseteq \overline{A \cap B}$.

Из этих двух включений следует первый закон де Моргана. Второй закон доказывается аналогичным образом.

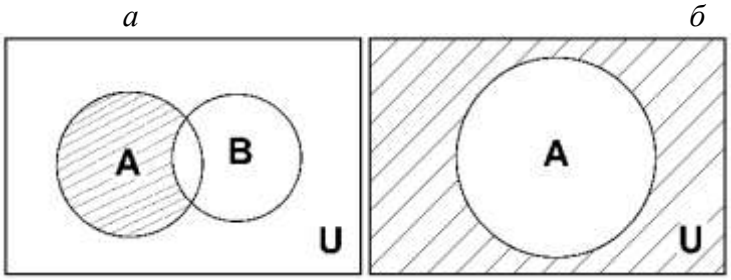



Рис 1.3

Примеры.

1. Найти $A \cup B; A \cap B; A \times B; B \times A; A \setminus B.$ $A = \{7;8;9\}; B = \{7;8;10\}$

Решение:

$$A \cup B = \{7;8;9\} \cup \{7;8;10\} = \{7;8;9;10\}$$

$$A \cap B = \{7;8;9\} \cap \{7;8;10\} = \{7;8\}$$

$$A \times B = \{7;8;9\} \times \{7;8;10\} = \{(7;7);(7;8);(7;10);(8;7);(8;8);(8;10);(9;7);(9;8);(9;10)\}$$

$$B \times A = \{7;8;10\} \times \{7;8;9\} = \{(7;7);(7;8);(7;9);(8;7);(8;8);(8;10);(10;7);(10;8);(10;9)\}$$

$$A \setminus B = \{7;8;9\} \setminus \{7;8;10\} = \{9\}.$$

2. Доказать равенство и записать двойственное ему:

$$(A \cup B)(B \cup C)(C \cup A) = ABC \cup AB \cup AC \cup BC$$

Решение:

Преобразуем левую часть:

$$(A \cup B)(B \cup C)(B \cup A) = (AB \cup C)(B \cup A) = AB \cup AB \cup CB \cup CA =$$

$$= ABC \cup AB \cup AC \cup BC$$

Таким образом, левая часть равна правой части, т.е. равенство верно.

Для того чтобы составить равенство, двойственное данному, пользуемся принципом двойственности. Заменим в данном равенстве знак $\cup$ на $\cap$ и наоборот. Чтобы не поменялся порядок действий, расставим скобки. Получим двойственное равенство:

$$AB \cup BC \cup CA = (A \cup B \cup C)(A \cup B)(A \cup C)(B \cup C)$$

*Мощностью конечного множества* называется количество его элементов.

Для конечного множества A через m (A) обозначим число элементов в множестве A.

Из определение следуют свойства:

1. m (A) + m ($\bar{A}$) = m (E)
2. A = B => m(A) = m(B)

Для любых конечных множеств справедливы так же утверждения:

1. m($A \cup B$)=m (A) + m (B) – m (A∩B)
2. m$(A \cup B \cup C)$=m(A)+m(B)+m(C)–m(A∩B)-m(A∩C)–m(B∩C)–m(A∩B∩C).

Дано множество А.

**Булеаном Р(А)** множества А называется множество всех подмножеств данного множества.

**Пример.** $A = \{1,2,3\}$

$$P(A) = \{\varnothing, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$$

Даны множества А и В.

**Прямым или декартовым произведением А×В** множеств А и В называется множество упорядоченных пар вида *(x; y)*, в которых *x* принимает все значения из множества А, а *y* - все значения множества В.

**Пример.** $A = \{1,2,3\},\ B = \{a,b\}$

$$A \times B = \{(1,a),(1,b),(2,a),(2,b),(3,a),(3,b)\}$$

**Решение задач с помощью кругов Эйлера**

Этот способ решать задачи придумал в XVIII в. великий Леонард Эйлер.

Пример. В олимпиаде по математике приняло участие 40 учащихся, им было предложено решить одну задачу по алгебре, одну по геометрии и одну по тригонометрии. По алгебре решили задачу 20 человек, по геометрии – 18 человек, по тригонометрии – 18 человек. По алгебре и геометрии решили 7 человек, по алгебре и тригонометрии – 9 человек. Ни одной задачи не решили 3 человека. Сколько учащихся решили все задачи? Сколько учащихся решили только две задачи? Сколько учащихся решили только одну задачу?

Решение:

Запишем коротко условие и покажем решение:

m (E)= 40;
m (A) = 20;
m (B) = 18;
m (C) = 18;
m (A∩B) = 7;
m (A∩C) = 8;
m (B∩C) = 9;
m (ABC) = 3 => m (ABC) = 40 – 3 = 37

Изобразим множества А, В, С (рис. 1.4).

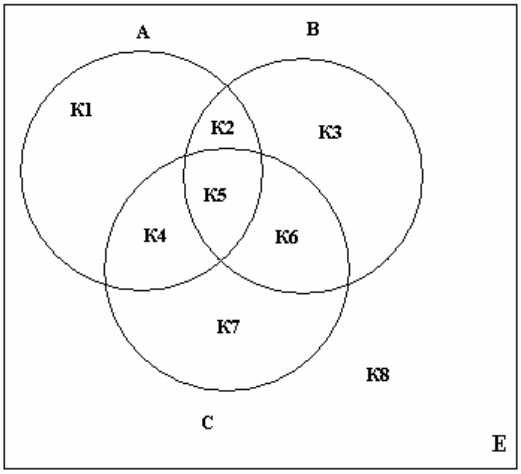



Рис 1.4

К1 – множество учеников, решивших только одну задачу по алгебре;

К2 – множество учеников, решивших только две задачи по алгебре и геометрии;

К3 – множество учеников, решивших только задачу по геометрии;
К4 – множество учеников, решивших только две задачи по алгебре и тригонометрии;
К5 – множество всех учеников, решивших все три задачи;
К6 – множество всех учеников, решивших только две задачи, по геометрии и тригонометрии;
К7 – множество всех учеников, решивших только задачу по тригонометрии;
К8 – множество всех учеников, не решивших ни одной задачи.
Используя свойство мощности множеств и рисунок можно выполнить вычисления:
m(К5)=m(A∩B∩C)=m(ABC)-m(A)-m(B)-m(C)+m(A∩B)+m(A∩C)+m(B∩C);
m (К5) = 37-20-18-18+7+8+9=5;  m (К2) = m (A∩B) - m (К5) = 7-5=2;
m (К4) = m (A∩C) - m (К5) = 8-5=3;  m (К6) = m (B∩C) - m (К5) = 9-5=4;
m (К1) = m (A) - m (К2) - m (К4) - m (К5) = 20-2-3-5=10;
m (К3) = m (B) - m (К2) - m (К6) - m (К5) = 18-2-4-5=7;
m (К7) = m (C) - m (К4) - m (К6) - m (К5) = 18-3-4-5 =6;
m (К2) + m (К4) + m (К6) = 2+3+4=9 – число учеников решивших только две задачи;
m (К1) + m (К3) + m (К7) = 10+7+6=23 – число учеников решивших только одну задачу.
Ответ: 5 учеников решили три задачи; 9 учеников решили только по две задачи; 23 ученика решили только по одной задаче.

**Задания для самостоятельного решения**

**Задача 1.** В классе 35 учеников. Каждый из них пользуется хотя бы одним из видов городского транспорта: метро, автобусом и троллейбусом. Всеми тремя видами транспорта пользуются 6 учеников, метро и автобусом – 15 учеников, метро и троллейбусом – 13 учеников, троллейбусом и автобусом – 9 учеников. Сколько учеников пользуются только одним видом транспорта?

**Задача 2.** Каждый из 35 шестиклассников является читателем, по крайней мере, одной из двух библиотек: школьной и районной. Из них 25 человек берут книги в школьной библиотеке, 20 – в районной.

Сколько шестиклассников:1. Являются читателями обеих библиотек;2. Не являются читателями районной библиотеки;3. Не являются читателями школьной библиотеки; 4. Являются читателями только районной библиотеки;5. Являются читателями только школьной библиотеки?

**Задача 3.** Из сотрудников фирмы 16 побывали во Франции,10-в Италии,6-в Англии; в Англии и Италии-5; в Англии и Франции -6; во всех трех странах - 5 сотрудников. Сколько человек посетили и Италию, и Францию, если всего в фирме работают 19 человек, и каждый из них побывал хотя бы в одной из названных стран?

**Задача 4.** В трёх  группах 70студентов. Из них 27 занимаются в драмкружке, 32 поют в хоре, 22 увлекаются спортом. В драмкружке 10 студентов из хора, в хоре 6 спортсменов, в драмкружке 8 спортсменов; 3 спортсмена посещают и драмкружок и хор. Сколько студентов не поют в хоре, не увлекаются спортом и не занимаются в драмкружке? Сколько студентов заняты только спортом?

**Задача 5.** Часть жителей нашего дома выписывают только газету «Комсомольская правда», часть – только газету «Известия», а часть – и ту, и другую газету. Сколько процентов жителей дома выписывают обе газеты, если на газету «Комсомольская правда» из них подписаны 85%, а на «Известия» – 75%?

**Порядок выполнения работы:**

1. Решить задания в тетради.

2. Получить у преподавателя задания для самостоятельной работы и решить их в тетради.

**Форма представления результата:**

Представить выполненные задания в тетради для практических работ преподавателю.

**Критерии оценки:**

Решение задачи считается безупречным, если правильно выбран способ решения, само решение сопровождается необходимыми объяснениями, верно выполнены нужные вычисления и преобразования, получен верный ответ, последовательно и аккуратно записано решение.

Отметка «5» ставится, если:

- работа выполнена полностью;
- в логических рассуждениях и обосновании решения нет пробелов и ошибок;
- в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);
- допущена одна ошибка или два-три недочета в выкладках, рисунках, чертежах или графиках (если эти виды работы не являлись специальным объектом проверки).

Отметка «3» ставится, если:

- допущены более одной ошибки или более двух-трех недочетов в выкладках, чертежах или графиках, но учащийся владеет обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что учащийся не владеет обязательными умениями по данной теме в полной мере.

## Практическое занятие № 3. Кардинальные числа: вычисление мощностей множеств

**Цель работы:** закрепление теоретических знаний о кардинальных числах и умение вычислять мощности конечных и бесконечных множеств. Развитие навыков сопоставления элементов множеств и использование теоремы Кантора-Бернштейна-Шредера для доказательства равенства мощностей.

**Выполнив работу, Вы будете:**

***уметь:***

- строить и анализировать дискретные модели
- выполнять операции над множествами;
- решать задачи на выполнение теоретико-множественных операций.

**Материальное обеспечение:**

Раздаточный материал (карточки с заданиями).

Технические средства обучения: компьютер с лицензионным программным обеспечением и мультимедиа проектор.

**Задания:**

**Кардинальное число (мощность множества)** — характеристика размера множества, определяемая количеством его элементов.

**Эквивалентные множества** — множества, между которыми можно установить биективное (взаимно однозначное) соответствие.

**Теорема Кантора-Бернштейна-Шредера** — утверждение, согласно которому, если каждое из двух множеств эквивалентно подмножеству другого, то сами множества эквивалентны.

**Часть 1. Мощности конечных множеств**

**Задача №1:**

Определите мощность следующего множества:

$A=\{x \in Z | -5 \leq x \leq 5\}$

Решение: Это множество целых чисел от -5 до 5 включительно. Количество элементов легко подсчитать простым перебором: всего 11 элементов (-5, -4, ..., 0, ..., 4, 5).

$|A|=11$

**Задача №2:**

Докажите, что множество чётных натуральных чисел эквивалентно множеству нечётных натуральных чисел.

Решение: Установим взаимно однозначное соответствие между множеством четных чисел {2,4,6,...}{2,4,6,...} и множеством нечётных чисел {1,3,5,...}. Соответствие задаётся формулой:

$f(n)=n-1, g(m)=m+1$

Эти функции обеспечивают биективное соответствие, следовательно, множества равномощны.

**Часть 2. Мощности бесконечных множеств**

**Задача №3:**

Покажите, что множество рациональных чисел Q счётно (эквивалентно натуральным числам N).

Решение: Метод диагонализации Кантора позволяет расположить все положительные рациональные числа в виде прямоугольника и пройти его диагональю, обеспечивая счётность:

$$\begin{array}{|c|c|c|c|}\hline\frac{1}{1}&\frac{1}{2}&\frac{1}{3}&\frac{1}{4}&...\\\hline\frac{2}{1}&\frac{2}{2}&\frac{2}{3}&\frac{2}{4}&...\\\hline\frac{3}{1}&\frac{3}{2}&\frac{3}{3}&\frac{3}{4}&...\\\hline\vdots&\vdots&\vdots&\vdots&...\end{array}$$

Таким образом, устанавливаем соответствие с натуральными числами, доказывая счётность Q.

**Задача №4:**

Рассмотрим множества $A=\{x|x=\sqrt{n}, n\in N\}$ и $B=R$. Докажите, что $|A|\neq|B|$ (множество вещественных чисел несчётно).

Решение: Множество A*A* представляет собой счётное множество квадратных корней из натуральных чисел. Но множество вещественных чисел RR несчётно (это доказано знаменитым диагональным аргументом Кантора). Следовательно, мощности множеств разные.

**Итоговая задача:**

Привести пример равномощных множеств, доказать их равномощность, используя биекцию.

Пример: Множество чётных положительных чисел и множество всех натуральных чисел.

Решение: Поставим в соответствие каждому чётному числу 2n само число n:

$f(n)=2n, f^{-1}(m)=m/2$

Так как эта функция биективна, множества равномощны.

**Порядок выполнения работы:**

1. Решить задания в тетради.
2. Получить у преподавателя задания для самостоятельной работы и решить их в тетради.

**Форма представления результата:**

Представить выполненные задания в тетради для практических работ преподавателю.

**Критерии оценки:**

Решение задачи считается безупречным, если правильно выбран способ решения, само решение сопровождается необходимыми объяснениями, верно выполнены нужные вычисления и преобразования, получен верный ответ, последовательно и аккуратно записано решение.

Отметка «5» ставится, если:

- работа выполнена полностью;
- в логических рассуждениях и обосновании решения нет пробелов и ошибок;

- в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);

- допущена одна ошибка или два-три недочета в выкладках, рисунках, чертежах или графиках (если эти виды работы не являлись специальным объектом проверки).

Отметка «3» ставится, если:

- допущены более одной ошибки или более двух-трех недочетов в выкладках, чертежах или графиках, но учащийся владеет обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что учащийся не владеет обязательными умениями по данной теме в полной мере.

## Тема 1.2. Булева алгебра

### Практическое занятие № 4. Построение таблиц истинности для логических операций (AND, OR, NOT)

### Практическое занятие № 5. Применение законов булевой алгебры для упрощения логических выражений

**Цель работы:** формирование умений упрощать логические выражения с помощью законов алгебры логики.

**Выполнив работу, Вы будете:**

***уметь:***

- анализировать логику высказываний и утверждений;

- формулировать задачи логического характера и применять средства математической логики для их решения.

**Материальное обеспечение:**

Раздаточный материал (карточки с заданиями).

Технические средства обучения: компьютер с лицензионным программным обеспечением и мультимедиа проектор.

**Задание:**

Внимательно прочитайте теоретические сведения.

Разберите решение задач, приведенных ниже, и запишите их в тетрадь.

Решите задачи из раздела для самостоятельного решения.

**Краткие теоретические сведения**

Рассмотрим законы, которым подчиняются логические функции.

*Переместительный закон* (закон коммутативности) для логического сложения и умножения соответственно:

$A \vee B = B \vee A$, **(1)**

$A \& B = B \& A$. **(2)**

*Сочетательный закон* (закон ассоциативности) для логического сложения и умножения соответственно:

$(A \vee B) \vee C = A \vee (B \vee C) = A \vee B \vee C$; **(3)**

$(A \& B) \& C = A \& (B \& C) = A \& B \& C$. **(4)**

Сочетательный закон устанавливает, что, если формула содержит только дизъюнкции или только конъюнкции, то скобки можно опустить.

*Распределительный закон* (закон дистрибутивности) для логического сложения и умножения соответственно:

$(A \& B) \vee C = (A \vee C) \& (B \vee C)$; **(5)**

$(A \vee B)\&C=(A\&C)\vee(B\&C)$. **(6)**

*Закон инверсии* (правило де Моргана) для логического сложения и умножения соответственно:

$\overline{A \vee B} = \overline{A}\ \&\ \overline{B}$; **(7)**

$A\&B= \overline{A}\ \vee \overline{B}$. **(8)**

Законы инверсии справедливы для любого числа переменных.

Для преобразований логических выражений используют следующие легко доказываемые тождества:

$A\vee0=A$; **(9)** $A\&1=A$; **(10)**

$A\vee1=1$; **(11)** $A\&0=0$; **(12)**

$A\vee A=A$; **(13)** $A\&A=A$; **(14)**

$A\vee \overline{A} =1$; **(15)** $A\& \overline{A} =0$; **(16)**

$\overline{\overline{A}} = A$. **(17)**

Справедливость этих законов можно доказать с помощью таблиц истинности сложных логических связей, описываемых законом.

С помощью законов алгебры логики могут быть доказаны соотношения, получившие названия:

*правила поглощения:*

$A\vee(A\&B)=A$; **(18)**

$A\&(A\vee B)=A$; **(19)**

*правила склеивания:*

$(A\&B)\vee( \overline{A} \&B)=B$; **(20)**

$(A\vee B)\&( \overline{A} \vee B)=B$; **(21)**

*закон Блейк - Порецкого:*

$A\vee( \overline{A} \&B)=AVB$. **(22)**

В алгебре логики все логические функции могут быть сведены путем логических преобразований к трем базовым: конъюнкции, дизъюнкции и инверсии.

1. Логическая операция – *импликация:* $A\rightarrow B= \overline{A}\ VB$.
2. Логическая операция – *эквивалентность:* $A\sim B=( \overline{A} \vee B)\&(A\vee \overline{B})$
3. Логическая операция–*сумма по модулю 2:* $A\oplus B=(A\vee B)\&( \overline{A} \vee \overline{B})$.
4. Логическая операция – *запрет:* $A\Delta B= A\& \overline{B}$.
5. Логическая операция – *«штрих Шеффера»:* $\overline{A} \mid B=A\&B$
6. Логическая операция – *«стрелка Пирса»:* $A\downarrow B= \overline{A \vee B}$.

Для доказательства этих соотношений достаточно построить таблицы истинности для правых частей и сравнить с таблицами, приведенными в определениях.

Пример. Выразить данные логические функции через элементарные операции: а) $F = \overline{(\overline{A} \rightarrow B)} \rightarrow C$; б) $F = \overline{(A \oplus C)} \downarrow \overline{B}$.

Решение. а) Преобразуем сначала вторую импликацию и воспользуемся тождеством (17), затем преобразуем первую импликацию и также воспользуемся тождеством (17):

$$F = \overline{(\overline{A} \rightarrow B)} \rightarrow C = \overline{\overline{(\overline{A} \rightarrow B)}} \vee C = (\overline{A} \rightarrow B) \vee C = \overline{\overline{A}} \vee B \vee C = A \vee B \vee C.$$

б) Преобразуем сначала операцию «стрелка Пирса», воспользуемся законом де Моргана (7) и тождеством (17), затем преобразуем операцию «сумма по модулю 2»:

$$F = \overline{(A \oplus C)} \downarrow \overline{B} = \overline{\overline{(A \oplus C)} \vee \overline{B}} = \overline{\overline{(A \oplus C)}} \& \overline{\overline{B}} = (A \oplus C) \& B = (A \vee C) \& (\overline{A} \vee \overline{C}) \& B.$$

Используя законы логики, формулы склеивания и поглощения и свойства логических операций, можно сложную логическую функцию заменить более простой, но равносильной ей функцией.

Пример. Упростить логические выражения: а) $(A\&B)\vee(A\&\overline{B})$; б) $\overline{\overline{A}\&\overline{B}}\vee A$.

Решение. а) Воспользуемся правилом дистрибутивности (6) и вынесем за скобки $A$:

$(A\&B)\vee(A\&\overline{B})=A\&(B\vee\overline{B})$.

По тождеству (15) получаем $B\vee\overline{B}=1$, следовательно:

$A\&(B\vee\overline{B})=A\&1=A$.

б) Воспользуемся правилом де Моргана (8) и тождествами (17) и (13):

$$\overline{\overline{A}\&\overline{B}}\vee A=\overline{\overline{A}}\vee\overline{\overline{B}}\vee A=A\vee B\vee A=A\vee B.$$

**Задания для самостоятельного решения**

**Задание 1.** Постройте таблицы истинности для данных логических выражений.

**Задание 2.** Докажите с помощью таблиц истинности равносильность данных логических выражений.

| № п/п | Задание 1 | Задание 2 |
|---|---|---|
| 1 | $(A\to B)\to\overline{C}$ | $(A\&B)\vee C$ и $(A\vee C)\&(B\vee C)$ |
| 2 | $(A/B)/C$ | $(A\vee B)\&C$ и $(A\&C)\vee(B\&C)$ |
| 3 | $(A\downarrow B)\downarrow C$ | $\overline{A\vee B\vee C}$ и $\overline{A}\,\&\,\overline{B}\,\&\,\overline{C}$ |
| 4 | $(A\vee\overline{B})\&(\overline{A}\vee B)\vee C$ | $\overline{A\&B\&C}$ и $\overline{A}\vee\overline{B}\vee\overline{C}$ |
| 5 | $(A\oplus B)\&C$ | $\overline{\overline{A\to B}}\to C$ и $A\vee B\vee C$ |
| 6 | $\left(\overline{B\&C\vee\overline{\overline{A}}}\right)\&A$ | $(A\vee B\vee C)\&\overline{A\vee\overline{\overline{B}}\vee C}$ и $B\&\overline{A}\&\overline{C}$ |
| 7 | $\left(\overline{B\&C}\vee\overline{A}\right)\&A$ | $\overline{(A\oplus C)}\downarrow\overline{B}$ и $(A\vee C)\&\left(\overline{A}\vee\overline{C}\right)\&B$ |
| 8 | $(A\oplus\overline{B})\&(A\oplus\overline{C})$ | $(A\vee B)\&A\&C$ и $(A\&\overline{B}\&C)\vee(A\&B\&C)$ |
| 9 | $A\to\overline{B\to C}$ | $(\overline{A}\vee\overline{B}\,\&C)\vee(\overline{A}\oplus\overline{B})$ и $(\overline{A}\vee\overline{B}\vee C)\&(\overline{A}\vee\overline{B}\vee\overline{C})$ |
| 10 | $(A\&\overline{B}\vee\overline{A}\,\&B)\&C$ | $(A\&B\&C)\vee(A\to\overline{B})$ и $\overline{A}\vee\overline{B}\vee C$ |

**Задание 3.** Выразите данные логические функции через элементарные операции.

**Задание 4.** Упростите логические выражения.

| № п/п | Задание 3 | Задание 4 |
|---|---|---|
| 1 | $(A\to B)\to\overline{C}$ | $X\,\&\left(\overline{X}\,\&Y\vee Z\right)\&\left(X\vee\overline{Z}\right)$ |
| 2 | $(A/B)\to\overline{C}$ | $\left(\overline{X}\vee Y\right)\&\left(\overline{Y}\vee X\,\&Z\right)$ |
| 3 | $(A\Delta B)\to C$ | $X\,\&\left(Y\Leftrightarrow X\right)\&\left(\overline{X}\vee\overline{Z}\right)$ |
| 4 | $\overline{\overline{A}\to B}\downarrow C$ | $\left(X\to Y\right)\&\,X\,\&\overline{Y}$ |
| 5 | $(A\oplus B)\&C$ | $\left(\overline{X}\,\&Y\right)\to\left(Z\,\&X\right)$ |
| 6 | $(A\oplus\overline{B})\&(A\to B)$ | $\left(X\,\&Y\Leftrightarrow Z\right)\&\,X\,\&\overline{Z}$ |
| 7 | $(\overline{\overline{A}\vee B})\downarrow C$ | $\left(X\,\&Z\vee\overline{X}\,\&\overline{Y}\right)\&\left(Z\to Y\right)$ |
| 8 | $(A\oplus\overline{B})\&(A\oplus\overline{C})$ | $\left(X\vee Y\,\&\overline{Z}\vee\overline{X}\,\&\overline{Y}\,\&Z\right)\&\,X\,\&\overline{Y}$ |
| 9 | $A\to\overline{B\to C}$ | $\left(X\to Y\right)\&\left(Y\to X\right)$ |
| 10 | $A\to\overline{B\downarrow C}$ | $\left(X\,\&\overline{Y}\,\&Z\vee\overline{X}\,\&\overline{Z}\right)\&Y$ |

**Задание 5.** Докажите равносильность формул, проводя эквивалентные преобразования одной или обеих частей:

№ п/п **Задание 5**

1 $(A \Rightarrow B) \Rightarrow B \sim A \vee B$;
2 $\neg(A \wedge \neg B) \Rightarrow (\neg B \Rightarrow A) \sim \neg(A \Rightarrow B) \vee A \vee B$;
3 $\neg(\neg A \Rightarrow \neg B) \Leftrightarrow A \sim (A \Leftrightarrow B) \wedge \neg A$;
4 $\neg(\neg A \vee \neg B) \vee \neg(A \Rightarrow B) \vee B \sim \neg(A \wedge \neg B) \Rightarrow (\neg B \Rightarrow A)$;
5 $(A \vee B) \wedge (\neg A \vee C) \sim (\neg A \wedge B) \vee (A \wedge C)$;
6 $(A \Rightarrow B) \wedge C \sim (\neg A \wedge B \wedge C) \vee (\neg A \wedge \neg B \wedge C) \vee (B \wedge C)$.

**Задание 6.** Укажите, в каких случаях высказывание истинно, а в каких ложно:

| № п/п | **Задание 6** |
|---|---|
| 1 | $(A \Rightarrow B) \wedge (A \downarrow C)$; |
| 2 | $(A \wedge B) \Rightarrow (C \wedge \neg C \Rightarrow A \vee C)$; |
| 3 | $(A \vee B) \wedge ((A \Rightarrow B) \Rightarrow C)$; |
| 4 | $(A \oplus B) \vee (A \oplus C)$; |
| 5 | $(A \downarrow B) \vee (B \wedge (B \Leftrightarrow B))$; |
| 6 | $(A \Rightarrow (B \wedge \neg C)) \wedge (A \Rightarrow C)$; |
| 7 | $(A \Rightarrow B) \vee (B \Leftrightarrow C)$; |
| 8 | $(A \Rightarrow B) \Rightarrow (A \mid C)$; |
| 9 | $(\neg B \Rightarrow \neg C)$; |
| 10 | $(A \wedge (B \vee \neg C)) \vee (\neg A \wedge B)$. |

**Порядок выполнения работы:**

1. Решить задания в тетради.

2. Получить у преподавателя задания для самостоятельной работы и решить их в тетради.

**Форма представления результата:**

Представить выполненные задания в тетради для практических работ преподавателю.

**Критерии оценки:**

Решение задачи считается безупречным, если правильно выбран способ решения, само решение сопровождается необходимыми объяснениями, верно выполнены нужные вычисления и преобразования, получен верный ответ, последовательно и аккуратно записано решение.

Отметка «5» ставится, если:

- работа выполнена полностью;
- в логических рассуждениях и обосновании решения нет пробелов и ошибок;
- в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);
- допущена одна ошибка или два-три недочета в выкладках, рисунках, чертежах или графиках (если эти виды работы не являлись специальным объектом проверки).

Отметка «3» ставится, если:

- допущены более одной ошибки или более двух-трех недочетов в выкладках, чертежах или графиках, но учащийся владеет обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что учащийся не владеет обязательными умениями по данной теме в полной мере.

## Практическое занятие № 6. Применение булевой алгебры в программировании: реализация логических операций в коде

**Цель работы:** освоение методов реализации основных логических операций средствами программирования и развитие практических навыков использования булевых выражений в условиях решения реальных прикладных задач, связанных с искусственным интеллектом.

**Выполнив работу, Вы будете:**

***уметь:***

- анализировать логику высказываний и утверждений;
- реализовывать базовые логические операции (AND, OR, NOT) на выбранном языке программирования;
- формулировать задачи логического характера и применять средства математической логики для их решения.

**Материальное обеспечение:**

Раздаточный материал (карточки с заданиями).

Технические средства обучения: компьютер с лицензионным программным обеспечением и мультимедиа проектор.

**Задание:**

Внимательно прочитайте теоретические сведения.

Разберите решение задач, приведенных ниже, и запишите их в тетрадь.

Решите задачи из раздела для самостоятельного решения.

**Краткие теоретические сведения**

Перед выполнением задания рекомендуется повторить следующие темы дисциплины:

Основные законы булевой алгебры.

Понятие таблицы истинности.

Минимизация логических функций методами карт Карно и алгоритма Квайна-МакКласки.

Реализация логических операторов в языках программирования Python, JavaScript, C++ и др.

**Порядок выполнения работы:**

**Часть I. Базовые логические операции**

Реализовать программы, демонстрирующие выполнение базовых логических операций:

**Простые логические выражения:** Напишите программу, которая принимает два булевых значения и выводит результат логического сложения, умножения и отрицания каждого аргумента.

```
# Примеры простых логических операций
a = True
b = False
print("a AND b:", a and b)
print("a OR b:", a or b)
print("NOT a:", not a)
```

**Таблицы истинности:** Создать таблицу истинности для заданной сложной логической формулы вида (A ∧ B) → ¬C. Таблица должна содержать столбцы всех переменных и результирующее значение.

```
from itertools import product

# Функция для вычисления результата формулы
def formula(A, B, C):
    return ((A and B) <= (not C))

# Создание таблицы истинности
for A, B, C in product([True, False], repeat=3):
    result = formula(A, B, C)
```

```
    print(f"A={A}, B={B}, C={C}: {result}")
```

**Часть II. Минимизация логических функций**

Используя метод Квайна-МакКласки минимизировать следующую формулу:F(x,y,z) = x'yz + xy'z + xyz

Разбейте задание на этапы:

Найдите минимальное покрытие импликантами.

Составьте каноническое выражение минимальной формы.

Реализуйте минимизированную функцию в виде программного кода.

Пример программы для проверки результатов минимизации:

```
def minimized_function(x, y, z):
    return (x and y) or (y and z)

print(minimized_function(True, True, False))  # Проверка значений
```

**Часть III. Решение практической задачи**

Создайте простую экспертную систему, основанную на булевом выводе. Например, система диагностики неисправности компьютера:

| **Признак** | **Значение** |
|---|---|
| Нет звука | true |
| Черный экран | false |
| Перезагрузка циклическая | true |

Определите правила вывода ошибок (например, перегрев процессора), используя булевы выражения и минимальные логические схемы.

**Контрольные вопросы:**

Какие операторы используются для реализации логических операций в Python?

Что такое таблица истинности и зачем она применяется?

Какое правило де Моргана используется для упрощения логических выражений?

Приведите пример применения метода Квайна-МакКласки для минимизации логической функции.

Объясните роль булевых функций в системах принятия решений.

**Порядок выполнения работы:**

1. Решить задания в тетради.

2. Получить у преподавателя задания для самостоятельной работы и решить их в тетради.

**Форма представления результата:**

Представить выполненные задания в тетради для практических работ преподавателю.

**Критерии оценки:**

Решение задачи считается безупречным, если правильно выбран способ решения, само решение сопровождается необходимыми объяснениями, верно выполнены нужные вычисления и преобразования, получен верный ответ, последовательно и аккуратно записано решение.

Отметка «5» ставится, если:

- работа выполнена полностью;

- в логических рассуждениях и обосновании решения нет пробелов и ошибок;

- в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);

- допущена одна ошибка или два-три недочета в выкладках, рисунках, чертежах или графиках (если эти виды работы не являлись специальным объектом проверки).

Отметка «3» ставится, если:

- допущены более одной ошибки или более двух-трех недочетов в выкладках, чертежах или графиках, но учащийся владеет обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что учащийся не владеет обязательными умениями по данной теме в полной мере.

## Тема 2.1. Основные понятия алгоритмов

### Практическое занятие № 7. Оценка времени выполнения алгоритмов: вычисление сложности O(n)

**Цель работы:** овладеть основными методами оценки временной сложности алгоритмов с использованием нотации Big-O, научиться определять сложность линейных алгоритмов и сравнивать эффективность различных подходов.

**Выполнив работу, Вы будете:**

***уметь:***

- самостоятельно оценивать временную сложность простейших алгоритмов типа O(n);
- понять разницу между алгоритмами различной сложности и уметь выбирать наиболее эффективный вариант;
- сравнивать производительности разных реализаций одного и того же алгоритма.

**Материальное обеспечение:**

Раздаточный материал (карточки с заданиями).

Технические средства обучения: компьютер с лицензионным программным обеспечением и мультимедиа проектор.

**Задание:**

Внимательно прочитайте теоретические сведения.

Разберите решение задач, приведенных ниже, и запишите их в тетрадь.

Решите задачи из раздела для самостоятельного решения.

**Краткие теоретические сведения**

Для успешного выполнения лабораторной работы студентам необходимо ознакомиться с такими понятиями и вопросами:

Определение нотации Big-O.

Временная сложность алгоритмов: оценка порядков роста.

Линейные алгоритмы (O(n)).

Методы измерения скорости исполнения программ.

**Ход работы:**

**Задание 1. Оценка линейной сложности**

Напишите простой алгоритм поиска элемента в массиве целых чисел. Используйте один цикл for и оцените его временную сложность.

**Алгоритм поиска элемента в массиве:**

```
def linear_search(arr, target):
    """
    Поиск целевого элемента в массиве.

    :param arr: Массив целых чисел
    :param target: Целевое число
    :return: Индекс первого найденного элемента или None, если элемент отсутствует
```

```
    """
    for i in range(len(arr)):
        if arr[i] == target:
            return i
    return None
```

Определите временную сложность данного алгоритма.

Подтвердите оценку экспериментально, измеряя среднее время выполнения для разных размеров массива.

**Задание 2. Оптимизация алгоритма**

Предложите оптимизацию для предыдущего алгоритма, позволяющую уменьшить количество итераций цикла. Рассчитайте новую временную сложность вашего улучшенного варианта.

**Оптимизированный алгоритм:**

```
def optimized_linear_search(arr, target):
    """
    Улучшенный поиск элемента в отсортированном массиве.

    :param arr: Отсортированный массив целых чисел
    :param target: Целевое число
    :return: Индекс первого найденного элемента или None, если элемент отсутствует
    """
    left, right = 0, len(arr)-1
    while left <= right:
        mid = (left+right)//2
        if arr[mid] >= target:
            right = mid - 1
        else:
            left = mid + 1
    if left < len(arr) and arr[left] == target:
        return left
    return None
```

Чем отличается временная сложность нового алгоритма от исходного?

Почему новый алгоритм работает быстрее на больших входных данных?

**Задание 3. Экспериментальное сравнение**

Проведите сравнительный тест обоих алгоритмов на массивах разного размера (например, от 1000 до 10 млн элементов). Запустите каждый алгоритм по крайней мере 10 раз и вычислите среднее время выполнения.

**Пример теста производительности:**

```
import timeit
import random

sizes = [1000, 10000, 100000, 1000000]
results = []

for size in sizes:
    data = sorted(random.sample(range(size*10), size))
    results.append((size,
                    timeit.timeit(lambda: linear_search(data, data[-1]), number=10),
                    timeit.timeit(lambda: optimized_linear_search(data, data[-1]), number=10)))

print(results)
```

**Контрольные вопросы:**

Что означает запись O(n)?

Какой порядок роста имеет линейный алгоритм?

Для чего нужны тесты производительности?

Когда использование сортировки перед поиском оправдано?

Может ли изменение структуры данных повлиять на производительность алгоритма?

**Порядок выполнения работы:**

1. Решить задания в тетради.

2. Получить у преподавателя задания для самостоятельной работы и решить их в тетради.

**Форма представления результата:**

По итогам занятия студенты предоставляют отчет, включающий:

Описание разработанных алгоритмов.

Расчеты временной сложности для каждого случая.

График зависимости времени выполнения от количества элементов в массиве.

Выводы относительно преимуществ и недостатков предложенных вариантов.

**Критерии оценки:**

Решение задачи считается безупречным, если правильно выбран способ решения, само решение сопровождается необходимыми объяснениями, верно выполнены нужные вычисления и преобразования, получен верный ответ, последовательно и аккуратно записано решение.

Отметка «5» ставится, если:

- работа выполнена полностью;

- в логических рассуждениях и обосновании решения нет пробелов и ошибок;

- в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);

- допущена одна ошибка или два-три недочета в выкладках, рисунках, чертежах или графиках (если эти виды работы не являлись специальным объектом проверки).

Отметка «3» ставится, если:

- допущены более одной ошибки или более двух-трех недочетов в выкладках, чертежах или графиках, но учащийся владеет обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что учащийся не владеет обязательными умениями по данной теме в полной мере.

## Практическое занятие № 8. Реализация и анализ базовых структур данных: массивы, списки, очереди, деревья

**Цель работы:** приобрести практические навыки реализации основных структур данных (массивы, списки, очереди, деревья) и освоить основы их анализа эффективности (сложность операций вставки, удаления, поиска).

**Выполнив работу, Вы будете:**

***уметь:***

- работать с базовыми структурами данных и способами их представления в памяти;

- реализовывать динамические списки, очереди и деревья;

- провести анализ временных характеристик операций над этими структурами;

- правильно выбирать структуру данных исходя из решаемой задачи.

**Материальное обеспечение:**

Раздаточный материал (карточки с заданиями).

Технические средства обучения: компьютер с лицензионным программным обеспечением и мультимедиа проектор.

**Задание:**

Внимательно прочитайте теоретические сведения.

Разберите решение задач, приведенных ниже, и запишите их в тетрадь.

Решите задачи из раздела для самостоятельного решения.

**Краткие теоретические сведения**

Способы хранения данных в оперативной памяти.

Основы организации структур данных (списки, очереди, стеки, бинарные деревья).

Алгоритмы обхода и манипуляции деревьями.

Анализ сложности операций в терминах Big-O.

**Ход работы**

**Часть 1. Реализация и тестирование структур данных**

Реализуйте вручную основные структуры данных и проведите первичное тестирование на небольших примерах.

**Задача 1. Массивы**

Реализуйте класс ArrayList, поддерживающий базовые операции:

Добавление элемента в конец списка (append).

Удаление последнего элемента (pop).

Получение элемента по индексу (getitem).

Установка элемента по индексу (setitem).

**Пример класса:**

```
class ArrayList:
    def __init__(self):
        self.data = []

    def append(self, value):
        """Добавляет элемент в конец"""
        self.data.append(value)

    def pop(self):
        """Удаляет последний элемент"""
        return self.data.pop()

    def getitem(self, index):
        """Получает элемент по индексу"""
        return self.data[index]

    def setitem(self, index, value):
        """Устанавливает элемент по индексу"""
        self.data[index] = value
```

**Задача 2. Связанные списки**

Реализуйте односвязанный список LinkedList с операциями добавления, удаления узлов и поиска элемента.

**Пример класса узла:**

```
class Node:
    def __init__(self, data=None):
        self.data = data
```

```
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def insert_at_beginning(self, data):
        new_node = Node(data)
        new_node.next = self.head
        self.head = new_node

    def remove_first(self):
        if self.head is not None:
            temp = self.head
            self.head = temp.next
            del temp
```

**Задача 3. Очереди**

Реализуйте очередь Queue на основе связанного списка или массива с поддержкой операций enqueue() и dequeue().

**Пример реализации очереди:**

```
class Queue:
    def __init__(self):
        self.items = []

    def enqueue(self, item):
        """Добавляем элемент в хвост очереди"""
        self.items.append(item)

    def dequeue(self):
        """Убираем первый элемент очереди"""
        if len(self.items) > 0:
            return self.items.pop(0)
        return None
```

**Задача 4. Бинарные деревья**

Реализуйте бинарное дерево BinaryTree с функциями вставки и поиска узлов.

**Пример реализации дерева:**

```
class TreeNode:
    def __init__(self, key):
        self.left = None
        self.right = None
        self.val = key

class BinarySearchTree:
    def __init__(self):
        self.root = None

    def insert(self, val):
        if self.root is None:
            self.root = TreeNode(val)
        else:
            current = self.root
```

```
            while True:
                if val < current.val:
                    if current.left is None:
                        current.left = TreeNode(val)
                        break
                    current = current.left
                elif val > current.val:
                    if current.right is None:
                        current.right = TreeNode(val)
                        break
                    current = current.right
                else:
                    break
```

**Часть 2. Анализ сложности операций**

Рассмотрите временные характеристики операций над каждой структурой данных:

Операции добавления, удаления и поиска в списке, очереди и дереве.

Постройте графики зависимости времени выполнения от объема данных.

**Методика эксперимента:**

Генерируйте случайные данные различного объема (например, 100, 1000, 10000 элементов).

Измеряйте время выполнения операций на каждом наборе данных.

Постройте график зависимости времени выполнения от размера набора данных.

**Подсказка:** Используйте модуль timeit для замеров времени выполнения.

```
import timeit
import random

data_sizes = [100, 1000, 10000]
times = {}

for size in data_sizes:
    # Генерируем случайные данные
    data = list(range(size))
    random.shuffle(data)

    # Замеряем время вставки в конец списка
    t_insert_list = timeit.timeit(lambda: arraylist.append(data[-1]), globals=globals(), number=100)

    times[size] = {'insert': t_insert_list}
```

**Часть 3. Заключение**

Сделайте выводы по результатам экспериментов:

Какие структуры данных эффективнее для конкретной операции?

В каких случаях лучше использовать массивы, списки, очереди или деревья?

**Контрольные вопросы:**

Назовите преимущества и недостатки массива перед связанным списком.

Какие структуры данных подходят для быстрого поиска элемента?

Какова средняя сложность вставки и удаления элемента в бинарном дереве поиска?

Чем отличается очередь от стека?

Как определить временную сложность алгоритма?

**Порядок выполнения работы:**

1. Решить задания в тетради.

2. Получить у преподавателя задания для самостоятельной работы и решить их в тетради.

**Форма представления результата:**

По итогам занятия студенты предоставляют отчет, включающий:

Код реализованных классов и тестов.

Таблицу и графики измерений времени выполнения операций.

Выводы по каждому пункту задания.

**Критерии оценки:**

Решение задачи считается безупречным, если правильно выбран способ решения, само решение сопровождается необходимыми объяснениями, верно выполнены нужные вычисления и преобразования, получен верный ответ, последовательно и аккуратно записано решение.

Отметка «5» ставится, если:

- работа выполнена полностью;

- в логических рассуждениях и обосновании решения нет пробелов и ошибок;

- в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);

- допущена одна ошибка или два-три недочета в выкладках, рисунках, чертежах или графиках (если эти виды работы не являлись специальным объектом проверки).

Отметка «3» ставится, если:

- допущены более одной ошибки или более двух-трех недочетов в выкладках, чертежах или графиках, но учащийся владеет обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что учащийся не владеет обязательными умениями по данной теме в полной мере.

## Практическое занятие № 9. Построение и анализ графов в представлении "список смежности"

**Цель работы:** ознакомиться с основами теории графов, освоить способы представления графа в виде списка смежности, овладеть навыками построения и анализа графовых моделей, закрепить умение проводить теоретический и практический анализ графов.

**Выполнив работу, Вы будете:**

***уметь:***

- представлять граф в виде списка смежности;

- освоить процедуры обработки графов: обход вершин, определение степени вершины, поиск путей;

- реализации и анализа графов на примере конкретных задач;

- работать с методами визуализации графов и анализировать различные свойства графов (связность, наличие циклов и другие аспекты).

**Материальное обеспечение:**

Раздаточный материал (карточки с заданиями).

Технические средства обучения: компьютер с лицензионным программным обеспечением и мультимедиа проектор.

**Задание:**

Внимательно прочитайте теоретические сведения.

Разберите решение задач, приведенных ниже, и запишите их в тетрадь.

Решите задачи из раздела для самостоятельного решения.

**Краткие теоретические сведения**

Основные определения теории графов (вершины, рёбра, пути, компоненты связности).

Представление графа различными способами (матрица смежности, список смежности).

Типичные задачи на графах: поиск компонент связности, проверка наличия циклов, подсчёт числа рёбер и степеней вершин.

**Ход работы:**

**Часть 1. Формирование списка смежности**

Задача состоит в формировании списка смежности для заданного ориентированного графа. Студентам предлагается создать граф с вершинами V = {$v_1$ , $v_2$ , ..., v□} и рёбрами E = {(u,v)}.

**Порядок действий:**

Нарисовать схему графа.

Оформить граф в виде списка смежности.

Выполнить проверку правильности составления списка смежности.

**Пример:** Граф с вершинами {A,B,C,D} и рёбрами {(A,B),(B,C),(C,A),(D,A)}.Список смежности выглядит следующим образом:

```
{
  'A': ['B', 'D'],
  'B': ['C'],
  'C': ['A'],
  'D': ['A']
}
```

**Часть 2. Обход графа**

Обойти созданный ранее граф двумя известными методами: DFS (глубинный поиск) и BFS (поиск в ширину).

**Порядок действий:**

Реализовать рекурсивный алгоритм DFS для прохода графа.

Реализовать алгоритм BFS с использованием очереди.

Запустить оба алгоритма и зафиксировать последовательность посещённых вершин.

**Пример реализации DFS:**

```
def dfs(graph, start_vertex, visited=None):
    if visited is None:
        visited = set()
    visited.add(start_vertex)
    print(start_vertex, end=' ')
    for neighbor in graph[start_vertex]:
        if neighbor not in visited:
            dfs(graph, neighbor, visited)
```

**Пример реализации BFS:**

```
from collections import deque

def bfs(graph, start_vertex):
    queue = deque([start_vertex])
    visited = set([start_vertex])
    while queue:
        vertex = queue.popleft()
        print(vertex, end=' ')
        for neighbor in graph[vertex]:
            if neighbor not in visited:
```

```
            visited.add(neighbor)
            queue.append(neighbor)
```

**Часть 3. Нахождение компонент связности**

Выполните анализ графа на предмет наличия компонент связности. Используем алгоритм DFS/BFS для выявления компонент связности графа.

**Порядок действий:**

Программно реализовать процедуру нахождения компонент связности.

Проанализируйте полученный граф на количество компонент связности.

**Пример реализации:**

```
def find_connected_components(graph):
    visited = set()
    components = []
    for vertex in graph.keys():
        if vertex not in visited:
            component = []
            dfs_component(graph, vertex, visited, component)
            components.append(component)
    return components

def dfs_component(graph, vertex, visited, component):
    visited.add(vertex)
    component.append(vertex)
    for neighbor in graph.get(vertex, []):
        if neighbor not in visited:
            dfs_component(graph, neighbor, visited, component)
```

**Часть 4. Визуализация графа**

Используя библиотеку NetworkX или Matplotlib, создайте визуализацию графа на основе полученного списка смежности.

**Порядок действий:**

Установите необходимые библиотеки (NetworkX, Matplotlib).

Преобразуйте список смежности в объект Graph и постройте его визуализацию.

**Пример реализации:**

```
import networkx as nx
import matplotlib.pyplot as plt

graph_data = {
    'A': ['B', 'D'],
    'B': ['C'],
    'C': ['A'],
    'D': ['A']
}

G = nx.DiGraph()
for node, neighbors in graph_data.items():
    for neighbor in neighbors:
        G.add_edge(node, neighbor)

nx.draw(G, with_labels=True, font_weight="bold")
plt.show()
```

**Контрольные вопросы:**

Что такое список смежности?

Как реализуется обход графа с помощью алгоритма DFS?

В чём разница между алгоритмами DFS и BFS?

Как определяются компоненты связности в графе?

Зачем необходима визуализация графа?

**Порядок выполнения работы:**

1. Решить задания в тетради.

2. Получить у преподавателя задания для самостоятельной работы и решить их в тетради.

**Форма представления результата:**

По итогам занятия студенты предоставляют отчет, включающий:

Исходный код реализации графа в виде списка смежности.

Детали обхода графа и выявление компонент связности.

Скриншоты визуализации графа.

Выводы по работе с каждым этапом задания.

**Критерии оценки:**

Решение задачи считается безупречным, если правильно выбран способ решения, само решение сопровождается необходимыми объяснениями, верно выполнены нужные вычисления и преобразования, получен верный ответ, последовательно и аккуратно записано решение.

Отметка «5» ставится, если:

- работа выполнена полностью;

- в логических рассуждениях и обосновании решения нет пробелов и ошибок;

- в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);

- допущена одна ошибка или два-три недочета в выкладках, рисунках, чертежах или графиках (если эти виды работы не являлись специальным объектом проверки).

Отметка «3» ставится, если:

- допущены более одной ошибки или более двух-трех недочетов в выкладках, чертежах или графиках, но учащийся владеет обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что учащийся не владеет обязательными умениями по данной теме в полной мере.

## Тема 2.2. Поиск и сортировка

### Практическое занятие № 10. Реализация алгоритмов сортировки: пузырьковая сортировка, быстрая сортировка, сортировка слиянием

**Цель работы:** научиться эффективно реализовывать классические алгоритмы сортировки («пузырьковую», «быструю», «сортировку слиянием») и понимать различия в их эффективности и применимости в различных ситуациях.

**Выполнив работу, Вы будете:**

***уметь:***

- изучить три классических алгоритма сортировки.

- освоить принцип их работы и особенности реализации.

- исследовать зависимость времени выполнения от размера массива.

- применить полученные знания на практике, выполнив конкретные задания.

**Материальное обеспечение:**

Раздаточный материал (карточки с заданиями).

Технические средства обучения: компьютер с лицензионным программным обеспечением и мультимедиа проектор.

**Задание:**

Внимательно прочитайте теоретические сведения.

Разберите решение задач, приведенных ниже, и запишите их в тетрадь.

Решите задачи из раздела для самостоятельного решения.

**Краткие теоретические сведения**

Понятие алгоритма сортировки.

Критерии оценки качества алгоритмов (временная сложность, устойчивость, дополнительные затраты памяти).

Асимптотическая нотация Big-O для описания порядка роста сложности.

**Порядок выполнения работы:**

**Часть 1. Реализация пузырьковой сортировки**

Разработайте алгоритм пузырьковой сортировки и исследуйте его работу на примерах.

**Постановка задачи:**Создайте функцию bubble_sort(array), которая сортирует элементы массива методом пузырьковой сортировки.

**Пример реализации:**

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        swapped = False
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
                swapped = True
        if not swapped:
            break
    return arr
```

Проверьте работоспособность своего алгоритма на нескольких наборах данных (упорядоченный, частично упорядоченный, случайно перемешанный массив).

**Часть 2. Реализация быстрой сортировки**

Реализуйте быстрый алгоритм сортировки QuickSort.

**Постановка задачи:**Создайте функцию quicksort(array), которая сортирует элементы массива методом быстрой сортировки.

**Пример реализации:**

```
def partition(arr, low, high):
    pivot = arr[(low + high) // 2]
    i = low - 1
    j = high + 1
    while True:
        i += 1
        while arr[i] < pivot:
            i += 1
        j -= 1
        while arr[j] > pivot:
            j -= 1
        if i >= j:
            return j
        arr[i], arr[j] = arr[j], arr[i]
```

```
def quicksort(arr):
    def _quicksort(items, low, high):
        if low < high:
            split_index = partition(items, low, high)
            _quicksort(items, low, split_index)
            _quicksort(items, split_index + 1, high)
    _quicksort(arr, 0, len(arr) - 1)
    return arr
```

Проверьте быстроту вашей реализации на тех же наборах данных, что и в первой задаче.

**Часть 3. Реализация сортировки слиянием**

Реализуйте алгоритм сортировки Merge Sort.

**Постановка задачи:**Создайте функцию merge_sort(array), которая сортирует элементы массива методом сортировки слиянием.

**Пример реализации:**

```
def merge(left, right):
    merged = []
    left_idx, right_idx = 0, 0
    while left_idx < len(left) and right_idx < len(right):
        if left[left_idx] <= right[right_idx]:
            merged.append(left[left_idx])
            left_idx += 1
        else:
            merged.append(right[right_idx])
            right_idx += 1
    merged.extend(left[left_idx:])
    merged.extend(right[right_idx:])
    return merged

def merge_sort(arr):
    if len(arr) <= 1:
        return arr
    middle = len(arr) // 2
    left_half = merge_sort(arr[:middle])
    right_half = merge_sort(arr[middle:])
    return merge(left_half, right_half)
```

Проверить свою реализацию аналогично предыдущим двум задачам.

**Часть 4. Анализ и сравнение**

Исследуйте скорость работы трех созданных вами алгоритмов на одном и том же наборе данных разной длины (например, 100, 1000, 10000 элементов). Постройте график зависимости времени выполнения от размера массива.

**Методика исследования:**

Подготовьте наборы данных (случайно распределённые целые числа).

Зафиксируйте время выполнения каждого алгоритма на соответствующих наборах данных.

Построить график зависимости времени выполнения от размера массива.

**Совет:** Используйте встроенную функцию timeit для точного измерения времени выполнения.

**Контрольные вопросы:**

Какова временная сложность пузырьковой сортировки?

В чем заключается суть алгоритма быстрой сортировки?

Как устроена сортировка слиянием?

В каких случаях предпочтительнее применять быструю сортировку, а в каких — сортировку слиянием?

Почему важно учитывать стабильность сортировок?

**Порядок выполнения работы:**

1. Решить задания в тетради.

2. Получить у преподавателя задания для самостоятельной работы и решить их в тетради.

**Форма представления результата:**

По итогам занятия студенты предоставляют отчет, включающий:

Реализации трёх алгоритмов сортировки.

Таблицы с результатами тестирования и график зависимостей времени выполнения от размера массива.

Анализ полученных результатов и рекомендации по выбору подходящего алгоритма в зависимости от ситуации.

**Критерии оценки:**

Решение задачи считается безупречным, если правильно выбран способ решения, само решение сопровождается необходимыми объяснениями, верно выполнены нужные вычисления и преобразования, получен верный ответ, последовательно и аккуратно записано решение.

Отметка «5» ставится, если:

- работа выполнена полностью;
- в логических рассуждениях и обосновании решения нет пробелов и ошибок;
- в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);
- допущена одна ошибка или два-три недочета в выкладках, рисунках, чертежах или графиках (если эти виды работы не являлись специальным объектом проверки).

Отметка «3» ставится, если:

- допущены более одной ошибки или более двух-трех недочетов в выкладках, чертежах или графиках, но учащийся владеет обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что учащийся не владеет обязательными умениями по данной теме в полной мере.

## Практическое занятие № 11. Сравнение времени выполнения различных алгоритмов сортировки

**Цель работы:** практически сравнить и проанализировать эффективность алгоритмов сортировки (пузырьковая, быстрая, сортировка выбором, сортировка вставкой, сортировка слиянием), оценить их временную сложность на реальных примерах.

**Выполнив работу, Вы будете:**

***уметь:***

- объективно оценивать временную сложность алгоритмов на практике.
- осуществлять выбор оптимального алгоритма сортировки для конкретного сценария.

**Материальное обеспечение:**

Раздаточный материал (карточки с заданиями).

Технические средства обучения: компьютер с лицензионным программным обеспечением и мультимедиа проектор.

**Задание:**

Внимательно прочитайте теоретические сведения.

Разберите решение задач, приведенных ниже, и запишите их в тетрадь.

Решите задачи из раздела для самостоятельного решения.

**Краткие теоретические сведения**

Классические алгоритмы сортировки (bubble sort, selection sort, insertion sort, quicksort, mergesort).

Понятие временной сложности и её обозначение (Big-O notation).

Особенности устойчивости и затрат памяти для различных типов сортировок.

**Порядок выполнения работы:**

**Часть 1. Реализация алгоритмов сортировки**

Реализуйте пять популярных алгоритмов сортировки:

**Bubble Sort (пузырьковая сортировка):**

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr
```

**Selection Sort (сортировка выбором):**

```
def selection_sort(arr):
    n = len(arr)
    for i in range(n):
        min_idx = i
        for j in range(i+1, n):
            if arr[min_idx] > arr[j]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
    return arr
```

**Insertion Sort (сортировка вставкой):**

```
def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i-1
        while j >=0 and key < arr[j] :
            arr[j+1] = arr[j]
            j -= 1
        arr[j+1] = key
    return arr
```

**Quick Sort (быстрая сортировка):**

```
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)
```

**Merge Sort (сортировка слиянием):**

```
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        L = arr[:mid]
        R = arr[mid:]
        merge_sort(L)
        merge_sort(R)
        i = j = k = 0
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i += 1
            else:
                arr[k] = R[j]
                j += 1
            k += 1
        while i < len(L):
            arr[k] = L[i]
            i += 1
            k += 1
        while j < len(R):
            arr[k] = R[j]
            j += 1
            k += 1
    return arr
```

**Часть 2. Тестирование и замер времени выполнения**

Тестируем созданные алгоритмы на массивах разного размера и определяем среднюю скорость выполнения каждого алгоритма.

**Порядок действий:**

Подготовьте случайные массивы размером N = {100, 1000, 10000, 100000} элементов.

Используя модуль timeit, определите среднее время выполнения каждого алгоритма сортировки на указанных массивах.

Повторите эксперименты хотя бы трижды для исключения погрешностей и получите средние показатели.

**Пример запуска тайминга:**

```
import timeit
import random

arrays = [
    random.sample(range(100000), 100),
    random.sample(range(100000), 1000),
    random.sample(range(100000), 10000),
    random.sample(range(100000), 100000)
]

algorithms = ["bubble_sort", "selection_sort", "insertion_sort", "quick_sort", "merge_sort"]

for alg_name in algorithms:
    print(f"\nВремя выполнения {alg_name}:")
    for arr_size in arrays:
```

```
    exec_time = timeit.timeit(
        f"{alg_name}(arr)",
        setup=f"from __main__ import {alg_name}; arr = {arr_size}",
        number=10
    )
    print(f"Размер массива {len(arr_size)}, среднее время: {exec_time:.6f} сек.")
```

**Часть 3. Анализ и интерпретация результатов**

Постройте графики зависимости среднего времени выполнения от размера массива для каждого алгоритма сортировки. Сделайте выводы о целесообразности выбора определенного алгоритма в зависимости от размера обрабатываемых данных.

**Контрольные вопросы:**

В чем преимущество быстрой сортировки перед пузырьковой?

В каком случае сортировка вставкой оказывается эффективнее остальных?

Почему сортировка слиянием эффективна даже для крупных объемов данных?

Можно ли считать одну сортировку универсальной для любых случаев?

Как влияет размер массива на эффективность сортировочных алгоритмов?

**Порядок выполнения работы:**

1. Решить задания в тетради.

2. Получить у преподавателя задания для самостоятельной работы и решить их в тетради.

**Форма представления результата:**

По итогам занятия студенты предоставляют отчет, включающий:

Описание разработанных алгоритмов.

Расчеты временной сложности для каждого случая.

График зависимости времени выполнения от количества элементов в массиве.

Выводы относительно преимуществ и недостатков предложенных вариантов.

**Критерии оценки:**

Решение задачи считается безупречным, если правильно выбран способ решения, само решение сопровождается необходимыми объяснениями, верно выполнены нужные вычисления и преобразования, получен верный ответ, последовательно и аккуратно записано решение.

Отметка «5» ставится, если:

- работа выполнена полностью;

- в логических рассуждениях и обосновании решения нет пробелов и ошибок;

- в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);

- допущена одна ошибка или два-три недочета в выкладках, рисунках, чертежах или графиках (если эти виды работы не являлись специальным объектом проверки).

Отметка «3» ставится, если:

- допущены более одной ошибки или более двух-трех недочетов в выкладках, чертежах или графиках, но учащийся владеет обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что учащийся не владеет обязательными умениями по данной теме в полной мере.

## Практическое занятие № 12. Реализация и анализ линейного и бинарного поиска в массивах

**Цель работы:** ознакомить студентов с алгоритмами линейного и бинарного поиска, их реализацией на языке программирования, а также научить анализировать эффективность этих алгоритмов.

**Выполнив работу, Вы будете:**

***уметь:***

- самостоятельно оценивать временную сложность простейших алгоритмов типа O(n);
- понять разницу между алгоритмами различной сложности и уметь выбирать наиболее эффективный вариант;
- сравнивать производительности разных реализаций одного и того же алгоритма.

**Материальное обеспечение:**

Раздаточный материал (карточки с заданиями).

Технические средства обучения: компьютер с лицензионным программным обеспечением и мультимедиа проектор.

**Задание:**

Внимательно прочитайте теоретические сведения.

Разберите решение задач, приведенных ниже, и запишите их в тетрадь.

Решите задачи из раздела для самостоятельного решения.

**Краткие теоретические сведения**

Понимание разницы между линейным и бинарным поиском.

Применение бинарного поиска исключительно на отсортированных массивах.

Оценка временной сложности каждого алгоритма (линейный поиск O(N), бинарный поиск O(log N)).

**Порядок выполнения работы:**

**Часть 1. Реализация алгоритмов поиска**

**1. Линейный поиск**

Задача: разработать функцию, выполняющую последовательный перебор элементов массива для поиска заданного ключа.

**Пример реализации линейного поиска:**

```
def linear_search(arr, target):
    for idx, element in enumerate(arr):
        if element == target:
            return idx
    return -1
```

**2. Бинарный поиск**

Задача: разработать функцию, осуществляющую двоичный поиск на отсортированном массиве.

**Пример реализации бинарного поиска:**

```
def binary_search(arr, target):
    low, high = 0, len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return -1
```

**Часть 2. Анализ эффективности поиска**

Экспериментальным путем исследовать поведение двух алгоритмов поиска на массивах разного размера и содержания.

**Ход эксперимента:**

Подготовьте случайные массивы (размерностью 100, 1000, 10000, 100000 элементов).

Для бинарного поиска сначала отсортируйте массивы.

Выберите случайные ключи для поиска внутри массивов.

Произведите замер времени выполнения каждого алгоритма поиска.

**Использование модуля timeit:**

```
import timeit
import random

# Случайные массивы
arrays = [
    random.sample(range(100000), 100),
    random.sample(range(100000), 1000),
    random.sample(range(100000), 10000),
    random.sample(range(100000), 100000)
]

search_keys = [random.choice(arr) for arr in arrays]

# Время выполнения линейного поиска
linear_times = []
for arr, key in zip(arrays, search_keys):
    lin_exec_time = timeit.timeit(lambda: linear_search(arr, key), number=100)
    linear_times.append(lin_exec_time)

# Время выполнения бинарного поиска
binary_times = []
for arr, key in zip(sorted(arrays), search_keys):
    bin_exec_time = timeit.timeit(lambda: binary_search(arr, key), number=100)
    binary_times.append(bin_exec_time)
```

**Часть 3. Интерпретация результатов**

Проанализируйте результаты замеров времени и сделайте выводы о различиях между линейным и бинарным поиском. Обратите внимание на масштабируемость алгоритмов при увеличении размера массива.

**Контрольные вопросы:**

В каких случаях целесообразно использовать линейный поиск?

Какие условия необходимы для эффективного применения бинарного поиска?

Почему бинарный поиск обладает лучшей производительностью на больших объемах данных?

Всегда ли бинарный поиск превосходит линейный по времени выполнения?

Насколько сильно зависит эффективность поиска от размера массива?

**Порядок выполнения работы:**

1. Решить задания в тетради.

2. Получить у преподавателя задания для самостоятельной работы и решить их в тетради.

**Форма представления результата:**

По итогам занятия студенты предоставляют отчет, включающий:

Полные тексты ваших реализаций алгоритмов поиска.

Таблицы с результатами замеров времени выполнения для линейного и бинарного поиска.

Графики зависимости времени выполнения от размера массива.

Выводы по использованию и ограничениям каждого алгоритма.

**Критерии оценки:**

Решение задачи считается безупречным, если правильно выбран способ решения, само решение сопровождается необходимыми объяснениями, верно выполнены нужные вычисления и преобразования, получен верный ответ, последовательно и аккуратно записано решение.

Отметка «5» ставится, если:

- работа выполнена полностью;
- в логических рассуждениях и обосновании решения нет пробелов и ошибок;
- в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);
- допущена одна ошибка или два-три недочета в выкладках, рисунках, чертежах или графиках (если эти виды работы не являлись специальным объектом проверки).

Отметка «3» ставится, если:

- допущены более одной ошибки или более двух-трех недочетов в выкладках, чертежах или графиках, но учащийся владеет обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что учащийся не владеет обязательными умениями по данной теме в полной мере.

## Тема 3.1. Логические высказывания и предикаты

### Практическое занятие № 13. Построение таблиц истинности для логических высказываний

**Цель работы:** формирование умений упрощать логические выражения с помощью законов алгебры логики.

**Выполнив работу, Вы будете:**

***уметь:***

- анализировать логику высказываний и утверждений;
- формулировать задачи логического характера и применять средства математической логики для их решения.

**Материальное обеспечение:**

Раздаточный материал (карточки с заданиями).

Технические средства обучения: компьютер с лицензионным программным обеспечением и мультимедиа проектор.

**Задание:**

Внимательно прочитайте теоретические сведения.

Разберите решение задач, приведенных ниже, и запишите их в тетрадь.

Решите задачи из раздела для самостоятельного решения.

**Краткие теоретические сведения**

Истинность или ложность составных высказываний можно определять чисто формально, не обращаясь к смысловому содержанию высказываний.

Для каждого составного высказывания (логического выражения) можно построить таблицу истинности.

Таблица, показывающая, какие значения принимает логическое выражение при всех сочетаниях (наборах) значений входящих в него простых высказываний (логических переменных), называется *таблицей истинности*.

Для любого логического выражения достаточно просто построить таблицу истинности.

Алгоритм построения таблицы истинности:

1) подсчитать количество переменных $n$ в логическом выражении;
2) определить число строк в таблице $m=2^n$;
3) подсчитать количество логических операций в формуле;
4) установить последовательность выполнения логических операций с учетом скобок и приоритетов;
5) определить количество столбцов в таблице: число переменных плюс число операций;
6) выписать наборы входных переменных следующим образом:

а) разделить колонку значений первой переменной пополам и заполнить верхнюю половину 0, нижнюю половину 1;

б) разделить колонку значений второй переменной на четыре части и заполнить каждую четверть чередующимися группами 0 и 1, начиная с группы 0;

в) продолжать заполнение колонок группами 0 и 1 до тех пор, пока группы 0 и 1 не будут состоять из одного символа;

7) провести заполнение таблицы истинности по столбикам, выполняя логические операции в соответствии с установленной в п.4 последовательностью.

Пример 3. Построить таблицы истинности для следующих логических выражений: а) $F=(A \vee B)\&(\overline{A} \vee \overline{B})$; б) $F=A\&(\overline{B} \vee \overline{B} \& \overline{C})$.

а) Подсчитываем количество переменных: $n=2$;

определяем число строк в таблице: $m=2^2=4$;

подсчитываем количество логических операций в формуле: 5;

устанавливаем последовательность выполнения логических операций с учетом скобок и приоритетов:

$$\begin{array}{ccccccc} & 1 & & 5 & 2 & 4 & 3 \\ (A & \vee & B) & \& & (\overline{A} & \vee & \overline{B}) \end{array}$$

определяем количество столбцов в таблице: $2+5=7$;

строим и заполняем таблицу истинности наборами входных переменных и значениями логических операций:

| $A$ | $B$ | $A \vee B$ | $\overline{A}$ | $\overline{B}$ | $\overline{A} \vee \overline{B}$ | $(A \vee B)\&(\overline{A} \vee \overline{B})$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |

б) Подсчитываем количество переменных: $n=3$;

определяем число строк в таблице: $m=2^3=8$;

подсчитываем количество логических операций в формуле: 5;

устанавливаем последовательность выполнения логических операций с учетом скобок и приоритетов:

$$\begin{array}{ccccccc} & 5 & 1 & 4 & 1 & 3 & 2 \\ A & \& & (\overline{B} & \vee & \overline{B} & \& & \overline{C}) \end{array}$$

определяем количество столбцов в таблице: $3+5=8$;

строим и заполняем таблицу истинности наборами входных переменных и значениями логических операций:

| $A$ | $B$ | $C$ | $\overline{B}$ | $\overline{C}$ | $\overline{B} \& \overline{C}$ | $\overline{B} \vee (\overline{B} \& \overline{C})$ | $A\&(\overline{B} \vee \overline{B} \& \overline{C})$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

Логические выражения, у которых таблицы истинности совпадают, называются *равносильными.*

Для обозначения равносильных логических выражений используется знак «=».

Пример 4. Доказать, что логические выражения $\overline{A}\,\&\,\overline{B}$ и $\overline{A\vee B}$ равносильны.

Построим сначала таблицу истинности для логического выражения $\overline{A}\,\&\,\overline{B}$.

| $A$ | $B$ | $\overline{A}$ | $\overline{B}$ | $\overline{A}\,\&\,\overline{B}$ |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |

Построим теперь таблицу истинности для логического выражения $\overline{A\vee B}$.

| $A$ | $B$ | $A\vee B$ | $\overline{A\vee B}$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 |

Таблицы истинности совпадают, следовательно, логические выражения равносильны: $\overline{A}\,\&\,\overline{B}=\overline{A\vee B}$.

Пример 5. Построить таблицы истинности для следующих логических выражений: а) $F=(A\oplus\overline{B})\&C$; б) $F=\left(\overline{\overline{A}\to B}\right)\to C$.

а) Подсчитываем количество переменных: $n$=___;

определяем число строк в таблице: $m=2^{n}$=___;

подсчитываем количество логических операций в формуле: ___;

устанавливаем последовательность выполнения логических операций с учетом скобок и приоритетов:

$(A\oplus\overline{B})\&C$;

определяем количество столбцов в таблице: _____;

строим и заполняем таблицу истинности наборами входных переменных и значениями логических операций:

| $A$ | $B$ | $C$ | $\overline{B}$ | $A\oplus\overline{B}$ | $(A\oplus\overline{B})\&C$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | | | |
| 0 | 0 | 1 | | | |
| 0 | 1 | 0 | | | |
| 0 | 1 | 1 | | | |
| 1 | 0 | 0 | | | |
| 1 | 0 | 1 | | | |
| 1 | 1 | 0 | | | |
| 1 | 1 | 1 | | | |

б) Подсчитываем количество переменных: $n$=____;

определяем число строк в таблице: $m=2^{n}$=___;

подсчитываем количество логических операций в формуле: ___;

устанавливаем последовательность выполнения логических операций с учетом скобок и приоритетов:

$\left(\overline{\overline{A} \to B}\right) \to C$

определяем количество столбцов в таблице: _____;

строим и заполняем таблицу истинности наборами входных переменных и значениями логических операций:

| $A$ | $B$ | $C$ | $\overline{A}$ | $\overline{A} \to B$ | $\overline{\overline{A} \to B}$ | $\left(\overline{\overline{A} \to B}\right) \to C$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | | | |
| 0 | 0 | 1 | | | | |
| 0 | 1 | 0 | | | | |
| 0 | 1 | 1 | | | | |
| 1 | 0 | 0 | | | | |
| 1 | 0 | 1 | | | | |
| 1 | 1 | 0 | | | | |
| 1 | 1 | 1 | | | | |

**Порядок выполнения работы:**

1. Решить задания в тетради.

2. Получить у преподавателя задания для самостоятельной работы и решить их в тетради.

**Форма представления результата:**

По итогам занятия студенты предоставляют отчет, включающий:

Описание разработанных алгоритмов.

Расчеты временной сложности для каждого случая.

График зависимости времени выполнения от количества элементов в массиве.

Выводы относительно преимуществ и недостатков предложенных вариантов.

**Критерии оценки:**

Решение задачи считается безупречным, если правильно выбран способ решения, само решение сопровождается необходимыми объяснениями, верно выполнены нужные вычисления и преобразования, получен верный ответ, последовательно и аккуратно записано решение.

Отметка «5» ставится, если:

- работа выполнена полностью;

- в логических рассуждениях и обосновании решения нет пробелов и ошибок;

- в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);

- допущена одна ошибка или два-три недочета в выкладках, рисунках, чертежах или графиках (если эти виды работы не являлись специальным объектом проверки).

Отметка «3» ставится, если:

- допущены более одной ошибки или более двух-трех недочетов в выкладках, чертежах или графиках, но учащийся владеет обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что учащийся не владеет обязательными умениями по данной теме в полной мере.

## Практическое занятие № 14. Формализация предикатов для описания условий в задачах анализа данных

## Практическое занятие № 15. Применение предикатов в программировании для обработки данных

**Цель работы:** формирование умений формализовывать предложения с помощью логики предикатов.

**Выполнив работу, Вы будете:**

***уметь:***

- анализировать логику высказываний и утверждений;
- строить отрицания к предикатам;
- формализовывать предложения с помощью логики предикатов.

**Материальное обеспечение:**

Раздаточный материал (карточки с заданиями).

Технические средства обучения: компьютер с лицензионным программным обеспечением и мультимедиа проектор.

**Задания:**

Внимательно прочитайте теоретические сведения.

Разберите решение задач, приведенных ниже, и запишите их в тетрадь.

Решите задачи из раздела для самостоятельного решения.

1. Определите объект, свойство объекта, область значений, функцию отношения для данной области, терм для следующих высказываний:

а) людей под фамилиями Иванов, Петров, Сидоров очень много;

б) две различные точки не совпадают.

2. Введите одноместные предикаты на соответствующих областях и запишите при их помощи следующие высказывания в виде логики предикатов:

а) всякое натуральное число, делящееся на 12, делится на 2,4 и 6;

б) жители Швейцарии обязательно владеют или французским, или итальянским, или немецким языком;

в) функция, непрерывная на отрезке [0,1] сохраняет знак или принимает нулевое значение.

3. Введите предикаты на соответствующих областях и запишите при их помощи следующие высказывания в виде логики предикатов:

а) если а – корень уравнения от одной переменной с вещественными коэффициентами, то $\overline{a}$ также корень этого уравнения;

б) через две различные точки проходит одна единственная прямая;

в) каждый студент выполнил, по крайней мере, одну лабораторную работу;

г) если произведение натуральных чисел делится на простое число, то на него делится, по крайней мере, один из сомножителей.

**Краткие теоретические сведения**

В исчислении высказываний нет предметных переменных, то есть переменных, которые могут принимать нелогические значения, например, числовые. Для того чтобы в логические исчисления могли быть включены нелогические константы и переменные, вводится понятие предиката.

Определение. *N-местным предикатом* на множестве $X$ называется $n$-местная функция из множества $X^n$ во множество $\{0,1\}$.

Примеры. 1. Предикат $A(x) = "x \le 2"$ на множестве $X = R$ – одноместный.

2. Предикат $B(x,y) = "xy > 0"$ на множестве $X = R^2$ – двуместный.

Если $X = \{0,1\}$, то $n$ -местный предикат представляет собой $n$ -местную булеву функцию. Нульместный предикат представляет собой высказывание.

Для каждого предиката $A$ областью истинности называется множество $Y = \{x \in X | A(x) = 1\}$, на котором предикат принимает значение 1.

Примеры. 1. Для предиката $A(x) = "x \le 2"$ на множестве $X = R$ область

истинности $Y = \{x \in R | x \le 2\}$.

2. Для предиката $B(x,y) = "xy > 0"$ на множестве $X = R^2$ область истинности $Y = \{(x,y) \in R^2 | xy > 0\}$.

Поскольку множество значений любого предиката лежит во множестве $\{0,1\}$, то с предикатами можно производить все операции алгебры логики, и все известные свойства логических операций обобщаются для предикатов. Рассмотрим эти свойства (для удобства в свойствах записываются одноместные предикаты):

3. Коммутативность:

$P(x) \vee Q(x) = Q(x) \vee P(x)$, $P(x) \wedge Q(x) = Q(x) \wedge P(x)$.

2. Ассоциативность:

$P(x) \vee (Q(x) \vee R(x)) = (P(x) \vee Q(x)) \vee R(x)$,

$P(x) \wedge (Q(x) \wedge R(x)) = (P(x) \wedge Q(x)) \wedge R(x)$.

3. Дистрибутивность:

$P(x) \vee (Q(x) \wedge R(x)) = (P(x) \vee Q(x)) \wedge (P(x) \vee R(x))$,

$P(x) \wedge (Q(x) \vee R(x)) = (P(x) \wedge Q(x)) \vee (P(x) \wedge R(x))$.

4. Идемпотентность: $P(x) \vee P(x) = P(x)$, $P(x) \wedge P(x) = P(x)$.

5. Закон двойного отрицания: $\neg\neg P(x) = P(x)$.

6. Закон исключения третьего: $P(x) \vee \neg P(x) = 1$.

7. Закон противоречия: $P(x) \wedge \neg P(x) = 0$.

8. Законы де Моргана:

$\neg(P(x) \vee Q(x)) = \neg P(x) \wedge \neg Q(x)$,

$\neg(P(x) \wedge Q(x)) = \neg P(x) \vee \neg Q(x)$.

9. Свойства операций с логическими константами:

$P(x) \vee 1 = 1$, $P(x) \vee 0 = P(x)$, $P(x) \wedge 1 = P(x)$, $P(x) \wedge 0 = 0$.

Здесь $P(x)$, $Q(x)$ и $R(x)$ – любые предикаты.

В то же время, для предикатов определены операции специального вида, которые называются *Кванторами*.

Символ $\forall$ называется *Квантором Всеобщности* (*Общности*).

Символ $\exists$ называется *Квантором существования*.

Пусть дана запись $\forall xA$ (или $\exists xA$). Переменная $x$ называется *Переменной в кванторе*, а $A$ – *Областью действия квантора.*

Имеют место эквивалентности:

$\exists x_i A = \neg\forall x_i \neg A$. $\forall x_i A = \neg\exists x_i \neg A$.

$\neg\exists x_i A = \forall x_i \neg A$. $\neg\forall x_i A = \exists x_i \neg A$.

Предикат называется *Тождественно истинным* (*Тождественно ложным*), если при всех возможных значениях переменных он принимает значение 1(0).

Справедливы эквивалентности:

$\forall x \forall y P(x,y) = \forall y \forall x P(x,y)$, $\exists x \exists y P(x,y) = \exists y \exists x P(x,y)$.

Разноименные кванторы можно переставлять только следующим образом:

$\exists x \forall y P(x,y) \to \forall y \exists x P(x,y)$, $\exists y \forall x P(x,y) \to \forall x \exists y P(x,y)$.

Обратные формулы неверны.

**Пример.** Очевидно, что высказывание $\forall x \exists y (x + y = 0)$ $(X = R)$ истинно. Поменяем кванторы местами. Получим высказывание $\exists y \forall x (x + y = 0)$, которое является ложным.

Выражения с кванторами можно преобразовывать следующим образом:

$\forall x (P(x) \wedge Q(x)) = \forall x P(x) \wedge \forall x Q(x)$, $\exists x (P(x) \vee Q(x)) = \exists x P(x) \vee \exists x Q(x)$.

Справедливы эквивалентности:

$\forall x P(x) \vee \forall x Q(x) = \forall x P(x) \vee \forall y Q(y) = \forall x (P(x) \vee \forall y Q(y)) =$

$= \forall x \forall y (P(x) \vee Q(y))$.

$\exists x P(x) \wedge \exists x Q(x) = \exists x P(x) \wedge \exists y Q(y) = \exists x (P(x) \wedge \exists y Q(y)) =$

$= \exists x \exists y (P(x) \wedge Q(y))$.

Имеют место формулы:

$\forall x (P(x) \wedge C) = \forall x P(x) \wedge C$, $\exists x (P(x) \vee C) = \exists x P(x) \vee C$,

$\forall x (P(x) \vee C) = \forall x P(x) \vee C$, $\exists x (P(x) \wedge C) = \exists x P(x) \wedge C$.

**Задания для самостоятельного решения**

**Задание 1**

Определите, какие из следующих предложений истинные, а какие ложные, считая предметной областью множество действительных чисел R:

1. $\forall x \exists y (x + y = 9)$;
2. $\exists x \exists y (x + y = 9)$;
3. $\exists x \forall y (x + y = 9)$;
4. $\forall x \forall y (x + y = 9)$;
5. $\forall x (((x > 1) \vee (x < 2)) \Leftrightarrow (x = x))$;
6. $\forall x ((x^2 > x) \Leftrightarrow ((x > 1) \vee (x < 0)))$;
7. $\forall a ((\exists x (ax = 1)) \Leftrightarrow (a = 0))$;
8. $\forall a \exists b \forall x (x^2 + ax + b > 0)$;
9. $\exists b \forall a \exists x (x^2 + ax + b = 0)$;
10. $\forall b \exists a \forall x (x^2 + ax + b > 0)$;
11. $\exists a \forall b \exists x (x^2 + ax + b = 0)$.

**Задание 2**

Из следующих предикатов с помощью кванторов постройте всевозможные предложения (как первые четыре предложения предыдущей задачи) и определите их истинностные значения, считая предметной областью множество R :

1. $x^2 + y^2 = 16$;
2. $(x^2 + 1 = 0) \Rightarrow (x = 1)$;
3. $x < y$;
4. $x^2 = 25$.

**Задание 3**

Определите и изобразите на R множества истинности следующих одноместных предикатов:

1. $|x + 4| < 3$;
2. $(x > 1) \wedge (x < 1)$;
3. $\cos(x) > 1$;
4. $(x > 1) \vee (x < 1)$;
5. $x^2 + 9 > 0$;
6. $(x > 1) \Rightarrow (x < 1)$;
7. $(x^2 > 9) \Leftrightarrow (x > 3)$;

8. $(x > 1) \Leftrightarrow x < 1)$.

**Задание 4**

Определите и изобразите на действительной плоскости множества истинности следующих двуместных предикатов:

1. $(x \geq 0) \wedge (y\ 0)$;
2. $(|x| = |y|) \vee (xy > 0)$;
3. $(x \geq 0) \wedge (y \geq 0)$;
4. $(|x| > 2) \Rightarrow (|x| < 3)$;
5. $(x \geq 0) \Rightarrow (y \leq 0)$;
6. $(x2 > 0) \Rightarrow (x2 - 2x - 3 > 0)$;
7. $(x \geq 0) \Leftrightarrow (y \leq 0)$;
8. $(x^2 + y^2 > 1) \Rightarrow (xy < 0)$.

**Задание 5**

Запишите определения следующих предикатов в виде формул, принимая в качестве предметной области множество натуральных чисел и используя предикатный символ =, функциональные символы + и·:

1. d(x, y) $\Leftrightarrow$ x делится на y ;
2. D0 (x, y, z) $\Leftrightarrow$ z является общим делителем x и y ;
3. D(x, y, z) $\Leftrightarrow$ z является наибольшим общим делителем x и y ;
4. P (x) $\Leftrightarrow$ x простое число .

**Задание 6**

Запишите следующие утверждения в виде формул (при условиях предыдущей задачи):

1. если числа x и y делятся на z, то их сумма тоже делится на z ;
2. если x делится на y и y делится на z, то x делится на z ;
3. сумма двух чисел, имеющих разную четность, нечетна ;
4. остаток от деления x на 5 равен 2 ;
5. если произведение двух чисел делится на простое число, то на него делится хотя бы один из сомножителей

**Порядок выполнения работы:**

1. Решить задания в тетради.

2. Получить у преподавателя задания для самостоятельной работы и решить их в тетради.

**Форма представления результата:**

Представить выполненные задания в тетради для практических работ преподавателю.

**Критерии оценки:**

Решение задачи считается безупречным, если правильно выбран способ решения, само решение сопровождается необходимыми объяснениями, верно выполнены нужные вычисления и преобразования, получен верный ответ, последовательно и аккуратно записано решение.

Отметка «5» ставится, если:

- работа выполнена полностью;

- в логических рассуждениях и обосновании решения нет пробелов и ошибок;

- в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);

- допущена одна ошибка или два-три недочета в выкладках, рисунках, чертежах или графиках (если эти виды работы не являлись специальным объектом проверки).

Отметка «3» ставится, если:

- допущены более одной ошибки или более двух-трех недочетов в выкладках, чертежах или графиках, но учащийся владеет обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что учащийся не владеет обязательными умениями по данной теме в полной мере.

## Тема 3.2. Теория графов

### Практическое занятие № 16. Построение графов: ориентированные и неориентированные графы

**Цель работы:** научиться создавать и представлять графы (ориентированные и неориентированные), разобраться в способах их отображения и применить инструменты для визуализации.

**Выполнив работу, Вы будете:**

*уметь:*

- отличия между ориентированными и неориентированными графами.
- техники представления графов (список смежности, матрица смежности).
- строить и интерпретировать такие виды графов визуально.
- попробовать себя в создании интерактивных представлений графов с использованием инструментов Python.

**Материальное обеспечение:**

Раздаточный материал (карточки с заданиями).

Технические средства обучения: компьютер с лицензионным программным обеспечением и мультимедиа проектор.

**Краткие теоретические сведения**

**Графом** G = (V, X) называется пара двух конечных множеств: множество точек V и множество линий X, соединяющих некоторые пары точек.

Точки называются **вершинами**, или **узлами**, графа, линии – **ребрами** графа. Примеры графов приведены на рис.1. Подсчитаем сколько ребер и вершин у каждого из графов, изображенных на рис. 1.

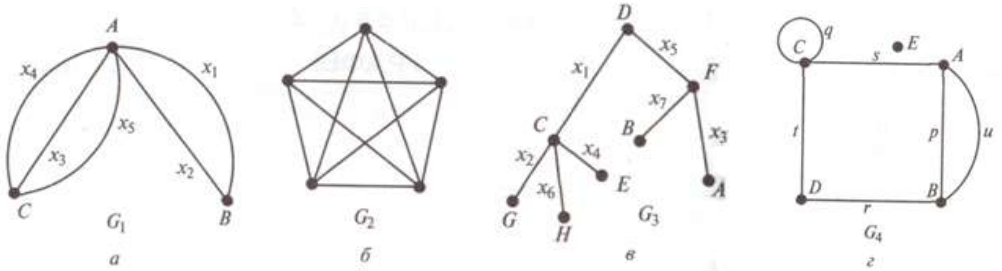



Рис. 1. Примеры графов: *а* – со смежными вершинами; *б* – полный; *в* – со смежными ребрами; *г* – с петлей

Если ребро графа G соединяет две его вершины, то говорят, что это ребро им **инцидентно**. Две вершины графа называются **смежными**, если существует инцидентное им ребро: на рис. 1, *а* смежными являются вершины A и B, A и C. Если граф G имеет ребро, у которого начало и конец совпадают, то это ребро называется **петлей**. На рис. 1, *г* петля — q(C, C). Два ребра называются **смежными**, если они имеют общую вершину. На рис. 1, *в* смежными являются, например, ребра $x_1$ и $x_2$ с общей вершиной C.

Граф может иметь ребра с одинаковыми парами. Такие ребра называются **кратными,** или **параллельными**. На рис. 1, *а* кратными являются, например, ребра $x_1$ (A, B), $x_2$ (A, B). Вершинам A и C инцидентны ребра $x_3$, $x_4$, $x_5$. Количество одинаковых пар называется

**кратностью** ребра. На рис. 1, *а* ребро АС имеет кратность, равную 3, а ребро АВ— кратность, равную 2. Число ребер, инцидентных вершине А, называется **степенью** этой вершины и обозначается deg(A) (от англ. *degree* — степень). Если вершине инцидентна петля, она дает вклад в степень, равный двум, так как оба конца приходят в эту вершину.

На рис. 1, *в* вершина А имеет степень, равную 1, вершина С— 4, вершина D — 2. Записывается это в виде: deg (A) = 1, deg (C) =4, deg (D) = 2. Граф G4 (рис. 1, *г*) содержит пять вершин V= {A, B, C, D, E} и шесть ребер X={p, q, r, s, t, u}.

Вершина графа, имеющая степень, равную нулю, называется **изолированной**. Граф, состоящий из изолированных вершин, называется **нуль-графом**. Для нуль-графа X=Ø. Вершина графа, имеющая степень, равную 1, называется **висячей**. На рис. 1, *г* вершина Е — изолированная: deg (E) = 0, а вершины A, B, E, G, H на рис. 1, *в* — висячие.

Граф G называется **полным**, если любые две его различные вершины соединены одним и только одним ребром.

Существуют различные способы задания графа: геометрический (рисунки, схемы, диаграммы), простое перечисление вершин и ребер, табличный. Человеку удобно работать с графом-рисунком, т.к. он может легко установить связь между вершинами в наглядном виде с помощью ребер, изображаемых непрерывными линиями. Для машинной обработки удобнее задать граф в алгебраической форме – перечислением (списком) вершин и ребер.

Одним из самых распространенных способов задания графа является **матричный способ.** Пусть дан граф G = (V, X), где V= {$V_1$, $V_2$, …, $V_n$} – вершины, а X={ $X_1$, $X_2$, …, $X_n$} – ребра графа.

**Матрицей инцидентности** B(G) неориентированного графа G с *n* вершинами и *m* ребрами называется матрица размерности $n \times m$, элементы которой определяются следующим образом:

$$b_{ij} = \begin{cases} 1, \text{если вершина } V_i \text{ инцидентна ребру } X_j, \\ 0, \text{в противном случае или если } X_j \text{ - петля.} \end{cases}$$

Очевидно, что в каждом столбце матрицы инцидентности должно быть только два ненулевых числа, т.к. ребро инцидентно двум вершинам. Число ненулевых элементов каждой строки – степень соответствующей вершины.

Но в математике удобнее работать с квадратными матрицами, т.к. для них хорошо разработан соответствующий алгебраический аппарат.

**Матрицей смежности** A(G) неориентированного графа G называется матрица размерности $n \times n$, элементы которой определяются следующим образом:

$$a_{ij} = \begin{cases} 1, \text{если вершины } V_i \text{ и } V_j \text{ смежны}, \\ 0, \text{в противном случае.} \end{cases}$$

Поскольку для неориентированного графа ребра $(V_i, V_j)$, и $(V_j, V_i)$ одновременно принадлежат или не принадлежат графу, т.к. символизируют одно и тоже ребро, то $a_{ij}=a_{ji}$. Матрица смежности неориентированного графа является симметрической и не меняется при транспонировании.

Хотя формально каждая вершина всегда смежна сама с собой, в матрице смежности мы будем ставить $a_{kk} = 0$, если у нее нет петли, и $a_{kk} = 1$, если есть одна петля. Итак, если граф имеет матрицу смежности и не имеет петель, на главной диагонали у него всегда стоят нули.

**Расстоянием** $d(V_i, V_j)$ между вершинами $V_i$ и $V_j$ в неориентированном графе G называется наименьшее число ребер, соединяющих эти вершины. **Условный радиус** графа G относительно вершины $V_i$ определяется формулой:

$$r(V_i) = \max_{V_j \in V(G)} d(V_i, V_j).$$

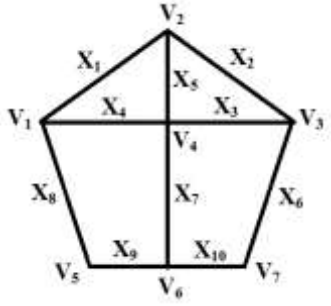
Здесь $V(G)$ – это множество вершин графа $G$.

**Радиус графа** $G$ определяется как наименьший из условных радиусов графа, а **центр графа** составляют вершины, условные радиусы графа относительно которых совпадают с радиусом графа.

**Задание.**

Рассмотрим граф G, изображенный на рисунке. Найдем для него:

1) таблицу степеней вершин;
2) матрицу смежности;
3) матрицу инцидентности;
4) таблицу расстояний в графе;
5) определить радиус и центр графа.




Решение.

1. Таблица степеней вершин данного графа имеет вид:

| $V_i$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
|---|---|---|---|---|---|---|---|
| deg ($V_i$) | 3 | 3 | 3 | 4 | 2 | 3 | 2 |

2. Составим таблицу смежности, а затем по ней матрицу смежности.

| $V_i$ | $V_j$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| $V_1$ | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| $V_2$ | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| $V_3$ | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| $V_4$ | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| $V_5$ | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| $V_6$ | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| $V_7$ | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

$$A(G) = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

3. Составим таблицу инцидентности, а затем по ней матрицу инцидентности.

| $V_i$ | $X_j$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ | $X_7$ | $X_8$ | $X_9$ | $X_{10}$ |
| $V_1$ | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| $V_2$ | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| $V_3$ | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| $V_4$ | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $V_5$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| $V_6$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| $V_7$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

$$B(G) = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

4. Для данного графа таблица расстояний и условных радиусов вершин имеет вид:

| | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ | $r(V_i)$ |
|---|---|---|---|---|---|---|---|---|
| $V_1$ | 0 | 1 | 2 | 1 | 1 | 2 | 3 | 3 |
| $V_2$ | 1 | 0 | 1 | 1 | 2 | 2 | 2 | 2 |
| $V_3$ | 2 | 1 | 0 | 1 | 3 | 2 | 1 | 3 |
| $V_4$ | 1 | 1 | 1 | 0 | 2 | 1 | 2 | 2 |
| $V_5$ | 1 | 2 | 3 | 2 | 0 | 1 | 2 | 3 |
| $V_6$ | 2 | 2 | 2 | 1 | 1 | 0 | 1 | 2 |
| $V_7$ | 3 | 2 | 1 | 2 | 2 | 1 | 0 | 3 |

5. Радиус графа G r(G)=2, следовательно, центр графа – это множество вершин $\{V_2, V_4, V_6\}$.

**Задание для самостоятельной работы**

В таблице для каждого варианта заданы декартовы координаты вершин графа и перечислены ребра графа. Граф неориентирован. Следует построить граф на плоскости *xOy* и найти:

1) таблицу степеней вершин;
2) матрицу смежности;
3) матрицу инцидентности;
4) таблицу расстояний в графе;
5) определить радиус и центр графа.

| | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ | $V_8$ |
|---|---|---|---|---|---|---|---|---|
| Вариант 1 | (1;3) | (3;5) | (6;5) | (2;2) | (3;3) | (1;0) | (3;0) | (6;2) |
| $(V_1;V_2), (V_2;V_5), (V_2;V_3), (V_2;V_4), (V_1;V_6), (V_2;V_7), (V_6;V_7)$ | | | | | | | | |
| Вариант 2 | (4;6) | (2;4) | (4;4) | (6;4) | (2;0) | (4;1) | (6;0) | (9;2) |
| $(V_1;V_2), (V_2;V_5), (V_2;V_3), (V_1;V_4), (V_4;V_7), (V_6;V_7), (V_1;V_3), (V_3;V_4), (V_5;V_6), (V_3;V_6)$ | | | | | | | | |
| Вариант 3 | (2;3) | (2;6) | (3;7) | (3;5) | (5;6) | (5;4) | (6;6) | (4;1) |
| $(V_1;V_2), (V_2;V_3), (V_4;V_6), (V_3;V_4), (V_5;V_6), (V_3;V_5) (V_5;V_7)$ | | | | | | | | |
| Вариант 4 | (1;1) | (2;2) | (2;4) | (2;5) | (3;5) | (5;5) | (3;2) | (5;2) |
| $(V_1;V_2), (V_2;V_3), (V_5;V_6), (V_3;V_5), (V_6;V_8), (V_2;V_7), (V_7;V_8), (V_5;V_7)$ | | | | | | | | |
| Вариант 5 | (1;4) | (3;5) | (5;4) | (1;2) | (5;2) | (1;0) | (5;0) | (7;1) |
| $(V_1;V_2), (V_2;V_4), (V_2;V_5), (V_2;V_3), (V_4;V_5), (V_6;V_7), (V_5;V_7), (V_4;V_6)$ | | | | | | | | |

**Порядок выполнения работы:**

1. Решить задания в тетради.

2. Получить у преподавателя задания для самостоятельной работы и решить их.

**Форма представления результата:**

Представить выполненные задания в тетради для практических работ преподавателю.

**Критерии оценки:**

Решение задачи считается безупречным, если правильно выбран способ решения, само решение сопровождается необходимыми объяснениями, верно выполнены нужные вычисления и преобразования, получен верный ответ, последовательно и аккуратно записано решение.

Отметка «5» ставится, если:

- работа выполнена полностью;
- в логических рассуждениях и обосновании решения нет пробелов и ошибок;
- в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);
- допущена одна ошибка или два-три недочета в выкладках, рисунках, чертежах или графиках (если эти виды работы не являлись специальным объектом проверки).

Отметка «3» ставится, если:

- допущены более одной ошибки или более двух-трех недочетов в выкладках, чертежах или графиках, но учащийся владеет обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что учащийся не владеет обязательными умениями по данной теме в полной мере.

## Практическое занятие № 17. Реализация алгоритмов поиска в глубину (DFS) и поиска в ширину (BFS) на графах

**Цель работы:** освоить методы поиска в графах: поиск в глубину (Depth First Search, DFS) и поиск в ширину (Breadth First Search, BFS); понять их принципиальные различия и области применения.

**Выполнив работу, Вы будете:**

*уметь:*

- писать алгоритмы поиска в глубину и поиске в ширину на графах.
- применять эти алгоритмы для прохождения графов и поиска компонентов связности.
- оценивать эффективность каждого подхода и видеть их достоинства и недостатки.

**Материальное обеспечение:**

Раздаточный материал (карточки с заданиями).

Технические средства обучения: компьютер с лицензионным программным обеспечением и мультимедиа проектор.

**Краткие теоретические сведения**

**Графом** G = (V, X) называется пара двух конечных множеств: множество точек V и множество линий X, соединяющих некоторые пары точек.

Точки называются **вершинами**, или **узлами**, графа, линии – **ребрами** графа. Примеры графов приведены на рис.1. Подсчитаем сколько ребер и вершин у каждого из графов, изображенных на рис. 1.

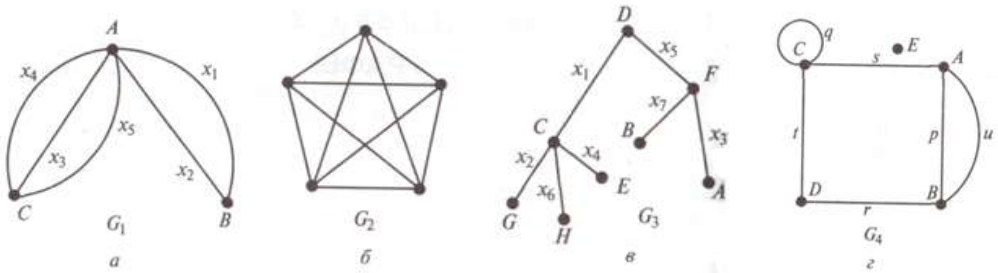



Рис. 1. Примеры графов: *а* – со смежными вершинами; *б* – полный; *в* – со смежными ребрами; *г* – с петлей

**Порядок выполнения работы:**

**Часть 1. Постановка задачи**

Создайте произвольный граф и представьте его в удобной форме (например, в виде списка смежности).

Например, пусть граф представлен таким образом:

```
{
    'A': ['B', 'C'],
    'B': ['A', 'D', 'E'],
    'C': ['A', 'F'],
    'D': ['B'],
    'E': ['B', 'F'],
    'F': ['C', 'E']
}
```

**Часть 2. Реализация поиска в глубину (DFS)**

Напишите функцию для обхода графа в глубину, начиная с определенной вершины.

**Пример реализации DFS:**

```
def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)
    print(start, end=' ')
    for next_vertex in graph[start]:
        if next_vertex not in visited:
            dfs(graph, next_vertex, visited)
```

Запустите функцию с начальной вершиной "A" и посмотрите порядок посещения вершин.

**Часть 3. Реализация поиска в ширину (BFS)**

Напишите функцию для обхода графа в ширину, начиная с определенной вершины.

**Пример реализации BFS:**

```
from collections import deque

def bfs(graph, start):
    visited = set()
    queue = deque([start])
    visited.add(start)
    while queue:
        vertex = queue.popleft()
        print(vertex, end=' ')
        for next_vertex in graph[vertex]:
            if next_vertex not in visited:
                visited.add(next_vertex)
                queue.append(next_vertex)
```

Также запустите функцию с начальной вершиной "A" и проанализируйте порядок посещения вершин.

**Часть 4. Анализ поведения алгоритмов**

Изучите и сравните поведение DFS и BFS на вашем графе. Попробуйте изменить начальную вершину и увидеть изменения в порядке обхода.

**Часть 5. Интерактивная визуализация (опционально)**

Вы можете попробовать воспользоваться библиотекой NetworkX для визуального отображения шагов алгоритмов поиска.

**Пример скрипта для визуализации:**

```
import networkx as nx
import matplotlib.pyplot as plt

# Наш граф
graph = {
   'A': ['B', 'C'], 'B': ['A', 'D', 'E'], 'C': ['A', 'F'],
   'D': ['B'], 'E': ['B', 'F'], 'F': ['C', 'E']
}

# Конструируем граф
G = nx.Graph()
for u, neighbors in graph.items():
   for v in neighbors:
      G.add_edge(u, v)

# Расположение вершин
pos = nx.spring_layout(G)

# Визуализация графа
nx.draw_networkx_nodes(G, pos, node_color='lightgreen')
nx.draw_networkx_edges(G, pos, edgecolor='grey')
nx.draw_networkx_labels(G, pos, font_size=10, font_family='sans-serif')

plt.title('Наш граф')
plt.axis('off')
plt.show()
```

**Контрольные вопросы:**

В чем принципиальная разница между DFS и BFS?
В каких случаях предпочтителен DFS, а в каких — BFS?
Как влияет порядок соседей на итоговый путь поиска?
Какая временная сложность у каждого из алгоритмов?
Какие существуют альтернативные подходы к поиску в графах?

**Порядок выполнения работы:**

1. Решить задания в тетради.
2. Получить у преподавателя задания для самостоятельной работы и решить их.

**Форма представления результата:**

Ваш собственный граф, используемый для демонстрации алгоритмов.
Функции реализации DFS и BFS с комментариями.
Результаты выполнения обоих алгоритмов на вашем графе.
Анализ достоинств и ограничений каждого подхода.

**Критерии оценки:**

Решение задачи считается безупречным, если правильно выбран способ решения, само решение сопровождается необходимыми объяснениями, верно выполнены нужные вычисления и преобразования, получен верный ответ, последовательно и аккуратно записано решение.

Отметка «5» ставится, если:

- работа выполнена полностью;
- в логических рассуждениях и обосновании решения нет пробелов и ошибок;
- в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);
- допущена одна ошибка или два-три недочета в выкладках, рисунках, чертежах или графиках (если эти виды работы не являлись специальным объектом проверки).

Отметка «3» ставится, если:

- допущены более одной ошибки или более двух-трех недочетов в выкладках, чертежах или графиках, но учащийся владеет обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что учащийся не владеет обязательными умениями по данной теме в полной мере.

## Практическое занятие № 18. Применение графов для моделирования реальных сетей и анализа данных

**Цель работы: н**аучиться применять теорию графов для моделирования реальных социальных, информационных и транспортных сетей, осваивая приемы анализа таких сетевых структур.

**Выполнив работу, Вы будете:**

*уметь:*

- освоить моделирующую силу графов для репрезентации реальных сетей.
- изучить возможности анализа сетей на основе метрик центральных узлов и кластеризации.
- выявлять важные узлы сети и критические связи.
- развить навыки работы с инструментами визуализации и анализа графов (например, библиотека NetworkX).

**Материальное обеспечение:**

Раздаточный материал (карточки с заданиями).

Технические средства обучения: компьютер с лицензионным программным обеспечением и мультимедиа проектор.

**Краткие теоретические сведения**

Модели реальных сетей (социальные сети, транспортные системы, информационные сети).

Центральность узлов (степень центральности, близость, посредничество).

Сообщества и кластеризация в сетях.

Использование графов для анализа распространения информации и эпидемий.

**Порядок выполнения работы:**

**Часть 1. Моделирование социальной сети**

Представьте небольшую социальную сеть, состоящую примерно из 10 человек. Определите связи ("дружба") между ними.

**Пример социального графа:**

```
friends_graph = {
    'Alice': ['Bob', 'Charlie'],
    'Bob': ['Alice', 'David', 'Emma'],
    'Charlie': ['Alice', 'Frank'],
    'David': ['Bob'],
    'Emma': ['Bob', 'George'],
    'Frank': ['Charlie', 'Hannah'],
    'George': ['Emma'],
    'Hannah': ['Frank']
}
```

**Часть 2. Анализ центральной роли участников**

Используя библиотеку NetworkX, рассчитайте степень центральности (degree centrality), близость (closeness centrality) и посредничество (betweenness centrality) для каждого участника сети.

**Пример кода для расчета центральности:**

```
import networkx as nx

# Преобразуем словарь друзей в граф
G = nx.Graph(friends_graph)

# Степенная центральность
deg_centrality = nx.degree_centrality(G)

# Близость
close_centrality = nx.closeness_centrality(G)

# Посредническая центральность
betw_centrality = nx.betweenness_centrality(G)

print("Степень центральности:")
for node, score in deg_centrality.items():
    print(f'{node}: {score}')

print("\nБлизость:")
for node, score in close_centrality.items():
    print(f'{node}: {score}')

print("\nПосредническая центральность:")
for node, score in betw_centrality.items():
    print(f'{node}: {score}')
```

**Часть 3. Визуализация сети**

Попробуйте визуализировать вашу социальную сеть, выделив важных участников разными цветами.

**Пример визуализации:**

```
import matplotlib.pyplot as plt

# Цветовая палитра для ключевых узлов
color_map = []
for node in G.nodes():
    color_map.append('red' if deg_centrality[node] > 0.3 else 'lightblue')
```

```
# Визуализация
pos = nx.spring_layout(G)
nx.draw_networkx_nodes(G, pos, node_color=color_map)
nx.draw_networkx_edges(G, pos, edge_color='gray')
nx.draw_networkx_labels(G, pos, font_size=10, font_family='sans-serif')

plt.title('Социальная сеть')
plt.axis('off')
plt.show()
```

**Часть 4. Анализ распространения влияния**

Предположим, что информация распространяется по вашей сети от одной точки. Посмотрите, кто является главным распространителем новостей и каким маршрутом идет распространение.

**Анализ распространения информации:**

Можно использовать метод моделирования случайного распространения или известный алгоритм симуляции процессов на графах (например, Independent Cascade Model).

**Контрольные вопросы:**

Какие реальные сети могут быть смоделированы с помощью графов?

Как определяется важность узла в сети?

Какие факторы влияют на быстрое распространение информации в социальной сети?

Как графы помогают выявить слабые места инфраструктуры транспортной сети?

Как связаны социальные сети и теория графов?

**Порядок выполнения работы:**

1. Решить задания в тетради.
2. Получить у преподавателя задания для самостоятельной работы и решить их.

**Форма представления результата:**

Самостоятельно составленную модель реальной сети.

Анализ центральной роли участников сети.

График визуализации сети.

Результаты анализа распространения влияния или передачи информации.

**Критерии оценки:**

Решение задачи считается безупречным, если правильно выбран способ решения, само решение сопровождается необходимыми объяснениями, верно выполнены нужные вычисления и преобразования, получен верный ответ, последовательно и аккуратно записано решение.

Отметка «5» ставится, если:

- работа выполнена полностью;
- в логических рассуждениях и обосновании решения нет пробелов и ошибок;
- в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);
- допущена одна ошибка или два-три недочета в выкладках, рисунках, чертежах или графиках (если эти виды работы не являлись специальным объектом проверки).

Отметка «3» ставится, если:

- допущены более одной ошибки или более двух-трех недочетов в выкладках, чертежах или графиках, но учащийся владеет обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что учащийся не владеет обязательными умениями по данной теме в полной мере.

## Тема 4.1. Основы комбинаторики

### Практическое занятие № 19. Решение задач на перестановки, сочетания и размещения
### Практическое занятие № 20. Применение формул комбинаторики для анализа данных.

**Цель:** сформировать умение решать задачи с использованием формул комбинаторики: перестановок, размещений и сочетаний.

**Выполнив работу, Вы будете:**

***уметь:***

- применять стандартные методы и модели к решению комбинаторных задач;
- решать задачи с использованием формул комбинаторики;
- анализировать задачу и выделять её составные части;
- составлять план действия;
- реализовать составленный план;
- оценивать результат и последствия своих действий (самостоятельно или с помощью наставника);
- определять необходимые источники информации;
- структурировать получаемую информацию;
- выделять наиболее значимое в перечне информации;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые);
- участвовать в диалогах на знакомые общие и профессиональные темы.

**Материальное обеспечение:**

Раздаточный материал (карточки с заданиями).

Технические средства обучения: компьютер с лицензионным программным обеспечением и мультимедиа проектор.

**Задание:**

Внимательно прочитайте теоретические сведения.

Разберите решение задач, приведенных ниже, и запишите их в тетрадь.

Решите задачи из раздела для самостоятельного решения.

**Краткие теоретические сведения**

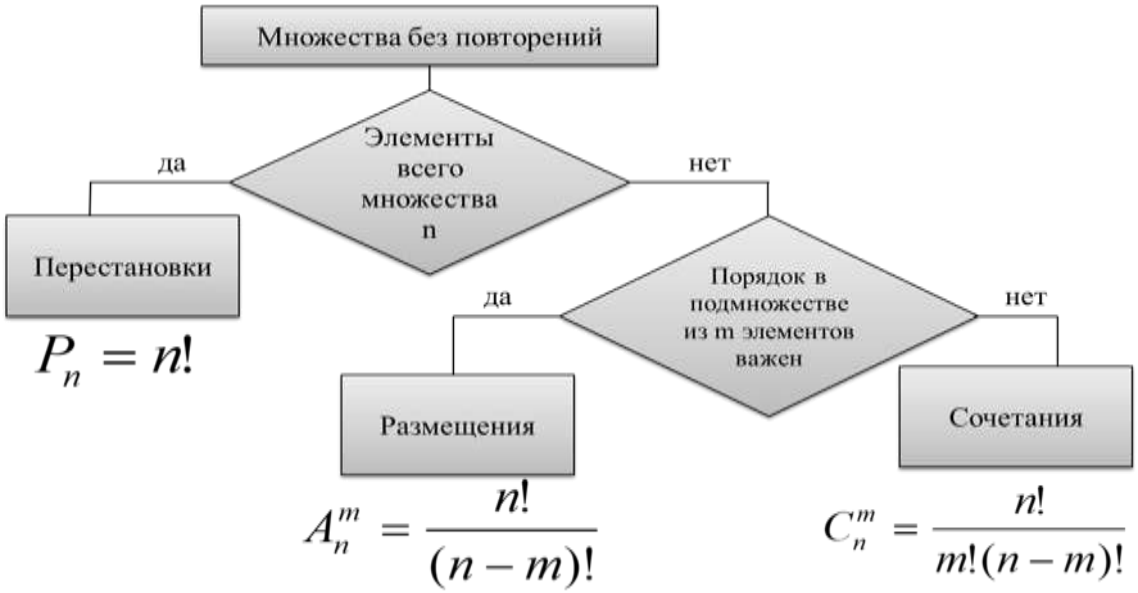
***Комбинаторика*** — один из разделов дискретной математики, изучающий способы подсчета числа элементов в конечных множествах, который приобрел большое значение в связи с использованием его в теории вероятностей, математической логике, теории чисел, вычислительной технике, кибернетике.

**Множества без повторений**

Множества элементов, состоящие из одних и тех же различных элементов и отличающиеся друг от друга только их порядком, называются ***перестановками*** этих элементов: $P_n = n!$, где $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$, $0! = 1$.

***Размещениями*** называют множества, составленные из $n$ различных элементов по $m$ элементов, которые отличаются либо составом элементов, либо их порядком: $A_n^m = \frac{n!}{(n-m)!}$.

***Сочетаниями*** из $n$ различных элементов по $m$ называют множества, содержащие $m$ элементов из числа $n$ заданных, и которые отличаются хотя бы одним элементом: $C_n^m = \frac{n!}{m!(n-m)!}$, где $C_n^0 = 1$, $C_n^n = 1$, $C_n^1 = n$.




**Множества с повторениями**

***Перестановки с повторениями***

$$\overline{P_n}(n_1, n_2, \ldots, n_k) = \frac{n!}{n_1! \cdot n_2! \cdot \ldots \cdot n_k!}$$

***Размещения с повторениями***

$$\overline{A_n^m} = n^m$$

***Сочетания с повторениями***

$$\overline{C_n^m} = \frac{(n+m-1)!}{m!(n-1)!} = C_{n+m-1}^m$$

**Правила сложения и умножения**

***Правило сложения.*** Если одно действие можно выполнить *m* способами, а другое *n* способами, и они взаимно исключают друг друга, то выполнить одно любое из этих действий можно *(n + m)* способами.

***Правило произведения.*** Если объект *A* можно выбрать из совокупности объектов *m* способами и после каждого такого выбора объект *B* можно выбрать *n* способами, то пара объектов *(A, B)* в указанном порядке может быть выбрана *m·n* способами.

**Примеры решения задач**

**Размещения без повторений**

**Задача 1.** Сколько можно составить телефонных номеров из 6 цифр каждый, так чтобы все цифры были различны?

**Решение**. Это пример задачи на размещение без повторений. Размещаются здесь 10 цифр по 6. А варианты, при которых одинаковые цифры стоят в разном порядке считаются разными. Значит, ответ на задачу будет:

$$A_{10}^6 = \frac{10!}{(10-6)!} = 151200$$

.

**Задача 2.** Сколькими способами 4 юноши могут пригласить четырех из шести девушек на танец?

**Решение**. Два юноши не могут одновременно пригласить одну и ту же девушку. И варианты, при которых одни и те же девушки танцуют с разными юношами, считаются разными, поэтому:

$$A_6^4 = \frac{6!}{(6-4)!} = \frac{720}{2} = 360$$

Возможно 360 вариантов.

**Перестановки без повторений**

**Задача 3.** Сколько различных шестизначных чисел можно составить из цифр 0, 1, 2, 3, 4,5, если цифры в числе не повторяются?

**Решение.**

1) Найдем количество всех перестановок из этих цифр: $P_6$=6!=720.
2) 0 не может стоять впереди числа, поэтому от этого числа необходимо отнять количество перестановок, при котором 0 стоит впереди. А это $P_5$=5!=120.

$P_6$-$P_5$=720-120=600.

**Сочетания без повторений**

**Задача 4.** Сколько трехкнопочных комбинаций существует на кодовом замке (все три кнопки нажимаются одновременно), если на нем всего 10 цифр.

**Решение**. Так как кнопки нажимаются одновременно, то выбор этих трех кнопок – сочетание. Отсюда возможно $C_{10}^3 = \frac{10!}{(10-3)!\cdot 3!} = \frac{8\cdot 9\cdot 10}{6} = 120$ вариантов.

**Задача 5.** У одного человека 7 книг по математике, а у второго – 9. Сколькими способами они могут обменять друг у друга две книги на две книги.

**Решение:**

Так как порядок следования книг не имеет значения, то выбор двух книг - сочетание. Первый человек может выбрать 2 книги $C_7^2 = \frac{7!}{(7-2)!\cdot 2!} = \frac{6\cdot 7}{2} = 21$ способами. Второй человек может выбрать 2 книги $C_9^2 = \frac{9!}{(9-2)!\cdot 2!} = \frac{8\cdot 9}{2} = 36$ способами. Значит всего по правилу произведения возможно 21·36=756 вариантов.

**Задача 6.** При игре в домино 4 игрока делят поровну 28 костей. Сколькими способами они могут это сделать?

**Решение**: Первый игрок делает выбор из 28 костей. Второй из 28–7=21 кости, третий из 14, а четвертый игрок забирает оставшиеся кости. Следовательно, возможно $C_{28}^7 \cdot C_{21}^7 \cdot C_{14}^7$.

**Размещения и сочетания с повторениями**

**Задача 7.** Сколько трехзначных чисел можно составить из цифр 1, 2, 3, 4, 5?

**Решение**. Так как порядок цифр в числе существенен, цифры могут повторяться, то это будут размещения с повторениями из пяти элементов по три, а их число равно $\overline{A_5^3} = 5^3 = 125$.

**Задача 8.** В кондитерском магазине продавались 4 сорта пирожных: эклеры, песочные, наполеоны и слоеные. Сколькими способами можно купить 7 пирожных.

**Решение**. Покупка не зависит от того, в каком порядке укладывают купленные пирожные в коробку. Покупки будут различными, если они отличаются количеством купленных пирожных хотя бы одного сорта. Следовательно, количество различных покупок равно числу сочетаний четырех видов пирожных по семь: $\overline{C_4^7} = \frac{(7+4-1)!}{7!(4-1)!} = \frac{10!}{7!\cdot 3!} = 120$.

**Задача 9.** Обезьяну посадили за пишущую машинку с 45 клавишами, определить число попыток, необходимых для того, чтобы она наверняка напечатала первую строку романа Л.Н. Толстого «Анна Каренина», если строка содержит 52 знака и повторений не будет?

**Решение**. Порядок букв имеет значение. Буквы могут повторяться. Значит, всего есть $\overline{A_{45}^{52}} = 45^{52}$ вариантов.

**Перестановки с повторениями**

**Задача 10.** Сколькими способами можно переставить буквы слова «ананас»?
**Решение**. Всего букв 6. Из них одинаковы: n1 (а)=3, n2(н)=2, n3 (с)=1. Следовательно, число различных перестановок равно $P_6(3,2,1) = \frac{6!}{3! \cdot 2! \cdot 1!} = 60$.

**Задачи для самостоятельного решения**

1. Сколько перестановок можно сделать из букв слова «Миссисипи».
Ответ: 2520
2. Имеется пять различных стульев и семь рулонов обивочной ткани различных цветов. Сколькими способами можно осуществить обивку стульев.
Ответ: 16807
3. На памятные сувениры в «Поле Чудес» спонсоры предлагают кофеварки, утюги, телефонные аппараты, духи. Сколькими способами 9 участников игры могут получить эти сувениры? Сколькими способами могут быть выбраны 9 предметов для участников игры?
Ответ: $4^9$, 220
4. Сколькими способами можно расставить на шахматной доске 8 ладей так, чтобы на одна из них не могла бить другую?
Ответ: 40320
5. Сколько может быть случая выбора 2 карандашей и 3 ручек из пяти различных карандашей и шести различных ручек?
Ответ:200
6. В течение 30 дней сентября было 12 дождливых дней, 8 ветреных, 4 холодных, 5 дождливых и ветреных, 3 дождливых и холодных, а один день был и дождливым, и ветреным, и холодным. В течение скольких дней в сентябре стояла хорошая погода.
Ответ: 15
7. На ферме есть 20 овец и 24 свиньи. Сколькими способами можно выбрать одну овцу и одну свинью? Если такой выбор уже сделан, сколькими способами можно сделать его еще раз?
Ответ: 480, 437
8. Сколькими способами можно выбрать гласную и согласную буквы из слова «здание»?
Ответ: 9
9. Сколько существует четных пятизначных чисел, начинающихся нечетной цифрой?
Ответ: 25000
10. В книжный магазин поступили романы Ф. Купера «Прерия», «Зверобой», «Шпион», «Пионеры», «Следопыт» по одинаковой цене. Сколькими способами библиотека может закупить 17 книг на выбранный чек?
Ответ: 2985

**Форма представления результата:**

Решение задач, оформленное в тетради.

**Критерии оценки:**

Решение задачи считается безупречным, если правильно выбран способ решения, само решение сопровождается необходимыми объяснениями, верно выполнены нужные вычисления и преобразования, получен верный ответ, последовательно и аккуратно записано решение.

Отметка «5» ставится, если:

- работа выполнена полностью;
- в логических рассуждениях и обосновании решения нет пробелов и ошибок;
- в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);

- допущена одна ошибка или два-три недочета в выкладках, рисунках, чертежах или графиках (если эти виды работы не являлись специальным объектом проверки).

Отметка «3» ставится, если:

- допущены более одной ошибки или более двух-трех недочетов в выкладках, чертежах или графиках, но учащийся владеет обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что учащийся не владеет обязательными умениями по данной теме в полной мере.

## Практическое занятие № 21. Построение деревьев решений с использованием комбинаторных методов

**Цель работы: о**своить создание и исследование деревьев решений для классификации объектов с использованием комбинаторных методов, развивать умения анализа комбинаторных ситуаций и формирования эффективных классификационных правил.

**Выполнив работу, Вы будете:**

*уметь:*

- изучение принципов построения деревьев решений.
- освоение комбинаторных методов оптимизации ветвей дерева.
- применение методов энтропии и информационного выигрыша для выбора лучших признаков разделения.
- овладение навыками проектирования и анализа деревьев решений.

**Материальное обеспечение:**

Раздаточный материал (карточки с заданиями).

Технические средства обучения: компьютер с лицензионным программным обеспечением и мультимедиа проектор.

**Ход работы:**

**Часть 1. Выбор оптимальной последовательности вопросов**

Рассмотрим гипотетическую ситуацию: мы хотим классифицировать животных по нескольким признакам (цвет шерсти, длина хвоста, форма тела и т.п.). Предположим, нам известно конечное число объектов (животные) и признаки, характеризующие каждое животное.

Заполните табличку признаков для небольшого множества объектов (например, кошки, собаки, птицы и рыбы).

**Пример таблицы признаков:**

| Животное | Шерсть | Длина хвоста | Форма тела | Питание |
|---|---|---|---|---|
| Кошка | Да | Средняя | Прямоугольн. | Мясо |
| Собака | Да | Большая | Прямоугольн. | Мясо |
| Птица | Нет | Маленькая | Круглая | Насекомые |
| Рыба | Нет | Очень маленькая | Удлинённая | Водоросли |

Используя метод ID3, выберите наилучший признак для первоначального разделения.

**Часть 2. Построение дерева решений**

Исходя из выбранного признака, продолжите формирование дерева решений, последовательно выбирая последующие признаки для дальнейших делений.

**Шаг 1.** Начнем с корня дерева — лучший признак.

**Шаг 2.** Затем разделим объекты согласно этому признаку и повторим процедуру выбора лучшего признака среди оставшихся.

Продолжайте деление до полного исчерпывания признаков или достижения однозначной классификации объектов.

**Часть 3. Комбинаторный анализ**

Посчитайте количество возможных способов разделения группы объектов, учитывая комбинации признаков.

**Формула для расчёта общего числа комбинаций:**

N=mn*N=mn*

Где:

m*m* — количество признаков,

n*n* — количество объектов.

Примените эту формулу к вашему примеру и найдите общее число возможных комбинаций.

**Часть 4. Оптимизация дерева решений**

Проверьте возможные сокращения дерева решений путём объединения ветвей с одинаковыми результатами или устранения избыточных ветвлений.

**Контрольные вопросы:**

Что такое дерево решений и как оно строится?

Как выбрать оптимальный признак для деления на этапе построения дерева?

Что такое информационная энтропия и как она помогает при выборе признаков?

Какие проблемы возникают при большом количестве признаков и объектов?

В каких областях применяются деревья решений?

**Порядок выполнения работы:**

1. Решить задания в тетради.

2. Получить у преподавателя задания для самостоятельной работы и решить их в тетради.

**Форма представления результата**

В ваш отчет должно войти:

Табличка признаков для рассматриваемых объектов.

Дерево решений с указанием этапов и принятых решений.

Результат комбинаторного анализа возможного пространства выбора признаков.

Предложения по сокращению и улучшению дерева решений.

**Критерии оценки:**

Решение задачи считается безупречным, если правильно выбран способ решения, само решение сопровождается необходимыми объяснениями, верно выполнены нужные вычисления и преобразования, получен верный ответ, последовательно и аккуратно записано решение.

Отметка «5» ставится, если:

- работа выполнена полностью;

- в логических рассуждениях и обосновании решения нет пробелов и ошибок;

- в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала).

Отметка «4» ставится, если:

- работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);

- допущена одна ошибка или два-три недочета в выкладках, рисунках, чертежах или графиках (если эти виды работы не являлись специальным объектом проверки).

Отметка «3» ставится, если:

- допущены более одной ошибки или более двух-трех недочетов в выкладках, чертежах или графиках, но учащийся владеет обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

- допущены существенные ошибки, показавшие, что учащийся не владеет обязательными умениями по данной теме в полной мере.