Отраслевая олимпиада школьников
«Газпром»
профиль «Информационные и коммуникационные технологии»

10-11 классы

Задание 1. Обработка данных

10 баллов – Программа выдаёт корректный результат для четырех задач, для каждой задачи реализована отдельная функция. Программа содержит не более двух синтаксических ошибок.

7 баллов – Программа выдаёт корректный результат для трех задач, для каждой задачи реализована отдельная функция. Программа содержит не более двух синтаксических или одной содержательной ошибок.

5 баллов – Программа выдаёт корректный результат для двух задач, для каждой задачи реализована отдельная функция. Программа содержит не более двух синтаксических ошибок.

3 балла – Программа выдаёт корректный результат для одной задачи, содержит не более двух синтаксических или одной содержательной ошибок.

1 балл – В комментариях присутствует хотя бы частичное описание логики программы, однако сама она не реализована или выдает неверный результат.

Вам поступили данные результатах буровых работ на нефтяных месторождениях. Данные представлены в виде списка (массива), где каждая запись содержит название месторождения и количество добытой нефти в баррелях:

Определите:

1. Среднее значение добычи нефти по всем месторождениям (результат – вещественное число с точностью до двух знаков после запятой).
2. Месторождение с наименьшим средним объёмом добычи, а также выведите соответствующий суммарный объём.
3. Для каждого месторождения размах (разница между максимальным и минимальным значением). Полученные данные сформировать в словарь (dict, map, Dictionary).
4. Медиану количества добытой нефти по всем месторождениям.

Формат ввода	[("Поле A", 2200), ("Поле B", 2500), ("Поле A", 2700), ("Поле C", 1600), ("Поле B", 2400), ("Поле A", 2600)]
Формат вывода	Среднее количество нефти: 2333.33 Месторождение с максимальным объёмом добычи: Поле C (1600 баррелей) Поле A: 500 Поле B: 100 Поле C: 0 Медиана добычи нефти: 2450.00

Решение задания. Необходимо реализовать 4 функции для решения 4 задач. В них должно быть:

- Вычисление среднего количества добытой нефти по всем месторождениям:
- Собираем все значения добычи нефти из списка. Затем суммируем их и делим полученную сумму на общее количество записей. Полученное число округляется до двух знаков после запятой.
    - Определение месторождения с наименьшим средним объёмом добычи и его суммарного объёма:
    - Сначала группируем записи по названию месторождения (например, «Поле А», «Поле В», «Поле С»).
    - Для каждой группы рассчитываем среднее значение добычи, суммируя все показатели для данного месторождения и деля на количество его записей.
    - Сравниваем средние значения всех групп, выбираем то месторождение, у которого среднее меньше всего.
    - Выводим название выбранного месторождения вместе с суммой добычи (в данном наборе данных для «Поля С» имеется всего одна запись, поэтому её среднее и сумма совпадают).
- Вычисление объема добычи для каждого месторождения:
- Для каждой группы (каждого месторождения) находим максимальное и минимальное значение добычи и их разность ними. Если в группе только одно значение, то разность равна нулю. Результаты собираются в словарь, где ключ – название месторождения, а значение – разность.
- Определение медианы добычи нефти по всем месторождениям:
    - Извлекаем все значения добычи и сортируем их по возрастанию.
    - Если общее число значений чётное, медианой считается среднее арифметическое двух центральных чисел. Если нечётное – выбирается центральное число.
    - Полученное значение округляется до двух знаков после запятой.

Решение на языке Python представлено ниже.

```python
from collections import defaultdict

def calculate_average(data):
    """
    Вычисляет среднее значение добычи нефти по всем месторождениям
    """
    total_oil = sum(production for _, production in data)
    count = len(data)
    return total_oil / count

def field_with_min_average(data):
    """
    Возвращает: название месторождения, среднее значение добычи
    """
    fields = defaultdict(list)
    for field, production in data:
        fields[field].append(production)

    min_avg_field = None
    min_avg_value = float('inf')
    for field, productions in fields.items():
        avg = sum(productions) / len(productions)
        if avg < min_avg_value:
            min_avg_value = avg
            min_avg_field = field
    return min_avg_field, min_avg_value

def calculate_field_ranges(data):
    """
    Вычисляет объем добычи для месторождений в формате: {название: объем добычи}
    """
    fields = defaultdict(list)
    for field, production in data:
        fields[field].append(production)

    field_ranges = {}
    for field, productions in fields.items():
        field_ranges[field] = max(productions) - min(productions)
    return field_ranges

def calculate_median(data):
    """
    Вычисляет медиану добычи нефти по всем месторождениям
    """
    productions = sorted(production for _, production in data)
    n = len(productions)
    if n % 2 == 0:
        median = (productions[n // 2 - 1] + productions[n // 2]) / 2
    else:
        median = productions[n // 2]
    return median
```

```
data = [
    ("Поле A", 2200),
    ("Поле B", 2500),
    ("Поле A", 2700),
    ("Поле C", 1600),
    ("Поле B", 2400),
    ("Поле A", 2600)
]

# Вызов функций для проведения расчётов
average = calculate_average(data)
min_field, min_field_avg = field_with_min_average(data)
ranges = calculate_field_ranges(data)
median = calculate_median(data)

# Вывод результатов
print(f"Среднее количество нефти: {average:.2f}")
print(f"Месторождение с наименьшим средним объёмом добычи: {min_field}
({int(min_field_avg)} баррелей)")
print("Размах добычи для каждого месторождения:")
for field in sorted(ranges.keys()):
    print(f"{field}: {ranges[field]}")
print(f"Медиана добычи нефти: {median:.2f}")
```

Задание 2. Теория чисел

10 баллов – Получен верный ответ, дано верное обоснование решения.

5 баллов – Получен верный ответ, дано обоснование логики, допущено не более одной логической ошибки.

2 балла – Рассуждения частично верны, однако ответ получен неверный.

Компания «НефтегазПром» эксплуатирует 7 скважин, каждая из которых за один день добывает определенный объем нефти, измеряемое в тысячах баррелей. На каждой скважине данный объем задается множеством:

{3, 5, 11, 4, 8, 6, 12}.

Для оптимизации транспортировки и переработки нефти партия нефти должна иметь общий объём, кратный 5 тысячам баррелей. Определите какое максимальное количество скважин позволит сформировать партию, если суммарный объём добычи выбранных скважин должен делиться на 5 без остатка.

Решение задания. Чтобы найти максимально возможное число скважин с суммарной добычей, делящейся на 5 без остатка, рассмотрим каждое значение по модулю 5 (mod – оператор деления по модулю, который возвращает остаток от деления одного числа на другое):

Исходные объёмы (тыс. баррелей): 3, 5, 11, 4, 8, 6, 12

Их остатки при делении на 5:

$3 \rightarrow 3 \pmod 5$

5 → 0

11 → 1

4 → 4

8 → 3

6 → 1

12 → 2

Сумма всех 7 = 3 + 5 + 11 + 4 + 8 + 6 + 12 = 49.

Остаток 49 mod 5 = 4, то есть не делится на 5.

Чтобы исправить это на 0 mod 5, нужно «убрать» из суммы число с остатком 4 (т. е. объём 4).

Если исключить только «4», сумма по остальным шести будет 49 – 4 = 45, что делится на 5. Таким образом, мы можем оставить 6 скважин, добившись кратности 5.

Ответ: максимальное количество скважин, дающее суммарный объём, кратный 5, равно 6.

Задание 3. Логические операции

10 баллов – Определено оптимальное логическое выражение и составлена логическая схема для минимизированного выражения функции.

5 баллов – Составлена логическая схема для минимизированного выражения функции, но не определено оптимальное логическое выражение.

1 балл – Есть попытка составить логическую схему, но решение неверное.

Вам поручили разработать систему автоматического контроля добычи нефти и газа. Логика системы мониторинга добычи нефти и газа описана функцией F(A, B, C, D), где:

A – датчик давления в нефтяной скважине

B – датчик температуры газа на входе в установку

C – сигнал уровня жидкости в резервуаре

D – датчик наличия газа в линии

Функция F(A, B, C, D) определяет, нужно ли автоматически остановить процесс добычи (1 – остановить, 0 – продолжить), она задаётся таблицей истинности:

A	B	C	D	F(A, B, C, D)
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Необходимо выполнить следующие задачи:

1. Определите выражение для функции F(A, B, C, D), требующее минимального количества логических элементов. Допустимым является использование только логических И, ИЛИ, НЕ.

2. Составьте логическую схему для минимизированного выражения функции F(A, B, C, D), используя следующие логические элементы: И, ИЛИ, НЕ). Укажите количество элементов для реализации схемы.

Решение задания. Функция F(A, B, C, D) принимает 4 входа (A, B, C, D) и определяет, нужно ли остановить добычу (1 – остановить, 0 – продолжить). Запишем наборы, при которых F = 1, т. е. процесс должен быть остановлен:

A	B	C	D	F(A, B, C, D)
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1
1	1	1	1	1

Нам нужно минимизировать выражение для F(A, B, C, D).

Дизъюнктивная нормальная форма (ДНФ) — это «сумма произведений» (OR из AND). Для функции F(A,B,C,D) вы сначала выписываете все комбинации, где F=1, и для каждой такой строки составляете конъюнкцию из переменных (A или $\overline{A}$, B или $\overline{B}$ и т. д.). Объединяя эти конъюнкции по ИЛИ, получаете полную ДНФ:

$F(A,B,C,D)=\overline{A}\cdot\overline{B}\cdot\overline{C}\cdot\overline{D} + \overline{A}\cdot\overline{B}\cdot C\cdot\overline{D} +\overline{A}\cdot B\cdot\overline{C}\cdot\overline{D}+\overline{A}\cdot B\cdot\overline{C}\cdot D+\overline{A}\cdot B\cdot C\cdot D+A\cdot\overline{B}\cdot\overline{C}\cdot D+$
$+A\cdot\overline{B}\cdot C\cdot\overline{D}+A\cdot B\cdot\overline{C}\cdot\overline{D}+A\cdot B\cdot C\cdot\overline{D}+A\cdot B\cdot C\cdot D$

Далее минимизируем выражение с помощью карт Карно. Там вы «склеиваете» соседние единицы (2, 4, 8 и т. д.), чтобы сократить количество переменных в каждом блоке. Каждый блок даёт упрощённый терм, а все такие блоки вместе образуют минимизированную ДНФ:

$F(A,B,C,D) = \overline{B}\cdot\overline{D} + \overline{A}\cdot B + A\cdot C$

Разбор оптимизированного выражения:

- $\overline{B}\cdot\overline{D}\rightarrow$ Остановка происходит, если B = 0 и D = 0 (низкое давление газа и отсутствие газа в линии).
- $\overline{A}\cdot B\rightarrow$ Остановка, если A = 0 и B = 1 (низкое давление в скважине и высокая температура газа).
- $A\cdot C\rightarrow$ Остановка, если A = 1 и C = 1 (высокое давление в скважине и высокий уровень жидкости).

Для реализации логической схемы используем нашу полученную минимизированную ДНФ (три логических элемента И, два элемента ИЛИ и три НЕ):

1. НЕ (инверторы):
   - Получаем $\overline{A}$, $\overline{B}$ и $\overline{D}$.
2. И (конъюнкции):
   - $\overline{B}\cdot\overline{D}$ (два входа).

- $\overline{A}\cdot B$ (два входа).
- $A\cdot C$ (два входа).

3. ИЛИ (дизъюнкции):
   - Первая ИЛИ: объединяет $\overline{B}\cdot\overline{D}$ и $\overline{A}\cdot B$.
   - Вторая ИЛИ: объединяет первый результат с $A\cdot C$.

Минимизированное выражение позволяет уменьшить количество логических элементов и сделать систему более эффективной.

Ответ. $F(A, B, C, D) = \overline{B} \cdot \overline{D} + \overline{A} \cdot B + A \cdot C$

Количество элементов для реализации схемы:

- 3 элемента НЕ.
- 3 элемента И.
- 2 элемента ИЛИ.

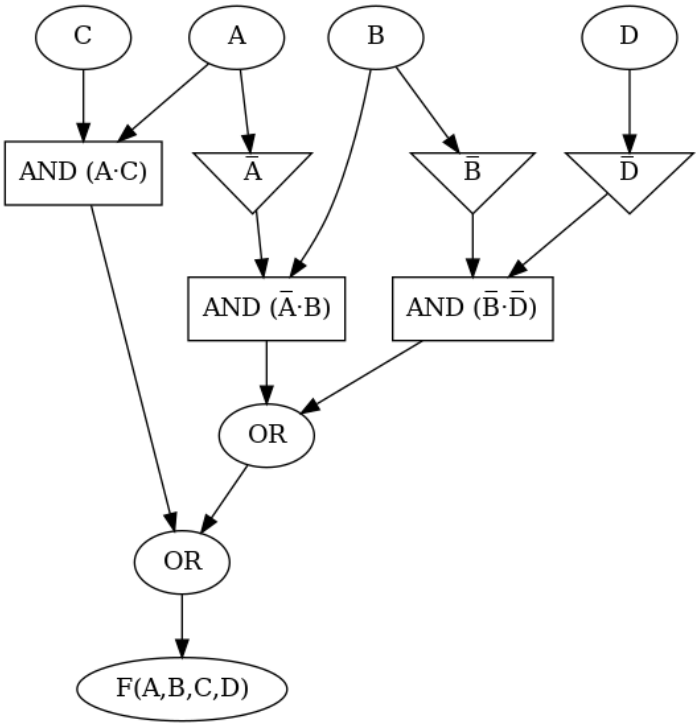

Рисунок 1 - Схема минимизированного выражения

Задание 4. Комбинаторика

10 баллов – Разработана функция, возвращающая корректный результат на всех входных данных.

5 балла – Разработан алгоритм, корректно работающий для всех входных данных, однако он не оформлен в виде функции. Допущено не более двух синтаксических или одной содержательной ошибки.

1 балл – В комментариях присутствует хотя бы частичное описание логики программы, однако сама она не реализована или выдает неверный результат.

На предприятии имеется два станка, которые можно задействовать для производства деталей. Разработайте программу, которая определит, каков

максимум деталей, которые можно произвести за смену продолжительностью K минут.

Условия:

- На запуск первого станка требуется **A** минут. После запуска он производит X деталей в минуту.
- На запуск второго станка требуется **B** минут. После запуска он производит Y деталей в минуту.
- Станки находятся в законсервированном состоянии, поэтому их запуск требует присутствия инженера, и оба станка нельзя запускать одновременно.
- После запуска одного из станков, другой станок можно запускать независимо, а оба станка могут производить детали одновременно.
- Каждый станок может работать не более $\mathbf{P_1}$ и $\mathbf{P_2}$ минут соответственно без необходимости охлаждения, которое занимает **Q** минут.

Формат входных данных
K – количество минут в смене ($1 \le K \le 10^6$), A – время запуска первого станка ($1 \le A \le k$), X – количество деталей, производимых первым станком за минуту ($1 \le X \le 10^6$), B – время запуска второго станка ($1 \le B \le K$), Y – количество деталей, производимых вторым станком за минуту ($1 \le Y \le 10^6$), P_1 — максимальное время работы первого станка без охлаждения ($1 \le P_1 \le K$), P_2 — максимальное время работы второго станка без охлаждения ($1 \le P_2 \le K$), Q — длительность охлаждения ($1 \le Q \le K$).

Решение задания. Для начала определим стратегии запуска, так как станки нельзя запускать одновременно, возможны два варианта:

1. Сначала запускаем первый станок, потом второй.
2. Сначала запускаем второй станок, потом первый.

Для каждого варианта мы считаем, сколько деталей успеет произвести каждый станок.

Далее проведем расчет производства деталей:

1. После запуска станка мы вычисляем, сколько времени у него остается на работу: $\text{Время работы} = \text{К} - \text{время запуска}$
2. Рабочие циклы состоят из:
   - Работы (до P_1 или P_2 минут).
   - Охлаждения (длится Q минут).
   - Вычисляем полное количество таких циклов:
     $$\text{количество} = \frac{\text{Время работы}}{P + Q}$$
   - Оставшееся время (если есть) используется без охлаждения.
3. Считаем общее число произведенных деталей.

Решение на языке Python представлено ниже.

```python
def max_parts_produced(total_time, start_delay1, rate1, start_delay2, rate2,
max_work1, max_work2, cooldown):
    """
    Вычисляет общее количество произведённых деталей двумя станками.

    Параметры функции:
        total_time: общее время работы
        start_delay1: время старта первого станка
        rate1: скорость производства первого станка (детали/единица времени)
        start_delay2: задержка второго станка относительно первого
        rate2: скорость производства второго станка (детали/единица времени)
        max_work1: максимальное время работы первого станка до перерыва
        max_work2: максимальное время работы второго станка до перерыва
        cooldown: время перерыва (охлаждения) после работы

    Возврат функции: общее количество произведённых деталей
    """

    def produced_parts(start, rate, work_limit):
        # Вычисляем доступное время после старта станка
        available_time = total_time - start
        if available_time <= 0:
            return 0

        # Полный цикл работы: рабочее время + охлаждение
        cycle_time = work_limit + cooldown

        # Считаем количество полных циклов
        full_cycles = available_time // cycle_time
        # Остаток времени, когда станок может работать без охлаждения
        extra_time = available_time % cycle_time

        # Производство за полные циклы плюс неполный цикл (ограничено work_limit)
        return full_cycles * (work_limit * rate) + min(extra_time, work_limit) * rate
```

Задание 5. Побитовые операции

15 баллов – Разработана функция, возвращающая верное значение и использующая алгоритм с асимптотической временной сложностью O(n).

10 баллов – Разработана функция, возвращающая верное значение, но асимптотическая временная сложность больше O(n). Допущено не более двух синтаксических ошибок.

5 баллов – Разработана программа без использования функции, возвращающая верное значение, но асимптотическая временная сложность больше O(n). Допущено не более одной содержательной.

1 балл – В комментариях присутствует хотя бы частичное описание логики программы, однако сама она не реализована или выдает неверный результат.

В будущем нефть будет транспортироваться не только по обычным трубопроводам, но и через новые насосные системы, которые используют цифровые датчики для контроля потока нефти. Эти датчики передают информацию о состоянии потока в двоичном виде. Ввиду ограниченности вычислительных ресурсов микроконтроллеров в них не заложена возможность работы со строковым типом данных. Однако инженеры столкнулись с проблемой: некоторые комбинации битов приводят к сбоям в работе насосов и системы управления, из-за чего прокачка нефти по трубам останавливается.

Учёные выяснили, что последовательность битов "101" вызывает сбои в работе насосов. Чтобы устранить эту проблему, система автоматически заменяет первую найденную последовательность "101" на "0110". Если после замены в числе всё ещё остаются такие последовательности, процесс повторяется до тех пор, пока их не будет.

Разработайте программу, которая выполнит все необходимые преобразования и устранит все сбои в системе.

Входные данные:

Целое число N ($1 \leq N \leq 10^9$) – числовое представление состояния потока.

Выходные данные:

Одно целое число – результат преобразований, представляющий новое состояние потока, без сбойных последовательностей.

Решение задания. Решение сводится к преобразованию двоичного представления числа, в котором мы поочередно заменяем все вхождения "101" → "0110" до полного их исчезновения. Решение на Python представлено ниже.

```python
def transform_flow_state(N: int) -> int:
    bits = []
    while N > 0:
        bits.append(N & 1)
        N >>= 1

    result = []

    for bit in bits:
        result.append(bit)
        while len(result) >= 3 and result[-3:] == [1, 0, 1]:
            result.pop()
            result.pop()
            result.pop()
            result.extend([0, 1, 1, 0])

    new_state = 0
    power_of_two = 1
    for b in result:
        if b == 1:
            new_state += power_of_two
        power_of_two <<= 1

    return new_state
```

Задание 6. Сжатие данных

10 баллов – Верно рассчитана эффективность сжатия, расписан ход решения, вывод правильно обоснован.

5 баллов – Ход решения и вывод верный, однако допущена ошибка в расчетах.

1 балл – Решение не верно, однако рассуждения частично верны.

Компания необходимо обрабатывать большие объёмы текстовой информации: отчёты, техническую документацию и исследования. Данные тексты содержат много повторяющихся терминов. Чтобы уменьшить объём передаваемых данных и ускорить их обработку, компания решает использовать алгоритм для сжатия текста.

Дан текст, состоящий из нескольких терминов, и частоты их появления в этом тексте.

Таблица частот	
Термин	Частота
«буровая установка»	13
«скважина»	22
«нефть»	26
«природный газ»	20
«трубопровод»	14
«бурение»	9

Необходимо выполнить следующие задачи:

1. Осуществите сжатие заданного текста с использованием выбранного Вами метода.
2. Рассчитайте и запишите отношение исходного размера данных к сжатому в процентах. Укажите, к какому классу сложности алгоритмов (например, линейно-логарифмическому) можно отнести использованный метод сжатия и почему.

Ограничения:

1. Сжатие должно производиться без потерь.
2. Временная сложность кодирования должна быть не хуже $O(n\log n)$.

Решение задания. Решение сводится к применению оптимального алгоритма сжатия текста для хранения и передачи терминов с их частотами. Учитывая ограничение на временную сложность не хуже $O(n\log n)$, необходимо выбрать метод, который обеспечивает эффективное кодирование без потерь, минимизируя объем данных.

Существует несколько алгоритмов сжатия без потерь, которые можно было бы рассмотреть:

1. Кодирование Хаффмана (Huffman Coding) – $O(n\log n)$:
   - Основан на частотном анализе: термины с большей частотой получают более короткие коды.

- Использует жадный алгоритм построения двоичного дерева, обеспечивающий оптимальную длину кодов.
- Эффективность: хорошо подходит для сжатия небольших словарей терминов, где частоты заранее известны.

2. Арифметическое кодирование – O(n), но сложнее в реализации:
   - Позволяет достичь почти оптимальной степени сжатия.
   - Минус: более сложная реализация и возможные ошибки округления.
   - Применяется в более сложных системах сжатия (например, JPEG, MPEG), но для частотных словарей из небольшого количества терминов не дает значительного преимущества перед Хаффманом.
3. Lempel-Ziv (LZ77, LZ78, LZW) – O(n), но неэффективен для небольших словарей:
   - Работает на основе поиска повторяющихся фрагментов, а не частотного анализа.
   - Лучше всего показывает себя на больших текстах, но в нашем случае, где всего 6 терминов, он неэффективен.
4. Golomb Coding – эффективен, но требует параметров:
   - Используется в ситуациях, где есть геометрическое распределение частот (например, сжатие картинок).
   - Не подходит, так как предполагает специфическое распределение данных, а не частотный анализ.

Так как данные содержат повторяющиеся термины, естественным выбором является Код Хаффмана – эффективный метод префиксного кодирования, который обеспечивает оптимальную длину кодирования с учетом частот терминов.

Как работает код Хаффмана?

- Алгоритм создает оптимальное двоичное дерево (*Huffman Tree*), где более часто используемые термины получают более короткие коды, а менее распространенные – более длинные.
- Используется жадный алгоритм для построения кодов, имеющий сложность O($n\log n$), что удовлетворяет требованиям задачи.

На основе таблицы частот терминов строится дерево Хаффмана:

Термин	Частота	Двоичный код
«нефть»	26	0
«скважина»	22	10
«природный газ»	20	110
«трубопровод»	14	1110
«буровая установка»	13	11110
«бурение»	9	11111

Таким образом, более частые термины получают более короткие коды, что значительно уменьшает объем данных.

Далее сделаем подсчет размера сжатого текста:

1. Исходный размер данных можно считать, как:

$$S_{original} = \sum_{\text{все термины}} (\text{длина термина} * \text{частота})$$

$S_{original}$ = (16×13)+(8×22)+(5×26)+(13×20)+(11×14)+(7×9) = 1004 бит

2. Сжатый размер рассчитывается как:

$$S_{compressed} = \sum_{\text{все термины}} (\text{длина кода Хаффмана} * \text{частота})$$

$S_{compressed}$ = (1×26)+(2×22)+(3×20)+(4×14)+(5×13)+(5×9)=296 бит

3. Отношение сжатия:

$$R = \frac{S_{original}}{S_{compressed}} * 100\%$$

$$R = \frac{1004}{296} * 100\%$$

$$R = 339.19\%$$

Проанализируем сложность алгоритма:

- Код Хаффмана строится за O($n$log$n$), так как использует очередь с приоритетом**.** Очередь с приоритетом используется в алгоритме Хаффмана, потому что она позволяет быстро извлекать два узла с наименьшей частотой, что является ключевой частью построения оптимального двоичного дерева кодирования.
- Кодирование текста выполняется за O(n), так как:
  - Каждое слово заменяется на свой двоичный код.
  - Это происходит за один линейный проход по исходному тексту.
  - Никаких вложенных операций (циклов), кроме простой подстановки, не выполняется.
- Расшифровка текста занимает O(n), так как:
  - Каждое прохождение по двоичному коду требует O(1) времени.
  - Общий размер двоичного кода пропорционален количеству терминов в тексте.
  - Следовательно, весь процесс декодирования занимает O(n).

Такой алгоритм оптимален для без потерь и соответствует требованиям задачи.

Ответ. После сжатия были сделаны следующие выводы:

- Метод сжатия – Код Хаффмана.
- Сложность алгоритма – O($n$log$n$).
- Сжатие без потерь, так как коды терминов уникальны.

- Отношение сжатия можно рассчитать по формулам выше, и оно будет значительно ниже 100%, так как длинные термины заменяются на короткие коды.

Задание 7. Нормализация данных

15 баллов – Верное приведение к первой, второй и третьей нормальным формам с корректными пояснениями.

11 баллов – Верное приведение к первой, второй и третьей нормальным формам без пояснений или с не корректными пояснениями.

8 баллов – Верное приведение к первой и второй нормальным формам с корректными пояснениями.

6 баллов – Верное приведение к первой и второй нормальным формам без пояснений или с не корректными пояснениями.

3 баллов – Верное приведение к первой нормальной форме с корректными пояснениями.

2 балла – Верное приведение к первой нормальной форме без пояснений или с не корректными пояснениями.

Подрядчики отправляют в корпоративную базу данных Транссиб таблицы о заказах на оборудование и материалы. Однако структура поступающих данных не соответствует правилам организации данных, т. е. нуждаются в нормализации по первым трем нормальным формам. Осуществите поэтапную нормализацию, строя модифицированные таблицы для каждого этапа нормализации, описывая нарушения и исправления.

Номер заказа	Код поставщика	Название поставщика	Наименование оборудования	Коли-чество	Цена за единицу	Общая стоимость
1	545	ГазМехСнаб	Газопровод, Шаровый кран	2, 1	25000, 15000	65000
2	546	НефтеСервис	Газоанализатор, Трубопровод	4, 3	7000, 15000	73000
3	547	ВодоГазСнаб	Фильтр, Газопровод	4, 2	8000, 25000	82000
4	548	ГазЭкспорт	Датчик давления, Гайка	3, 10	1500, 80	4800

Решение задания. Решение сводится к применению поэтапной нормализации данных, чтобы исключить избыточность, аномалии обновления и повысить целостность информации.

В данной таблице нарушены правила нормальных форм (1НФ, 2НФ, 3НФ), поэтому мы будем поэтапно приводить её к нормализованному виду (НФ – нормальная форма).

Исходная таблица содержит множественные нарушения нормализации:

1. Нарушение 1НФ:
   - В одном поле содержатся множественные значения (например, "Газопровод, Шаровый кран").
   - Общее поле "Общая стоимость" можно вычислить, оно избыточно.
2. Нарушение 2НФ:
   - Атрибуты "Название поставщика" зависят только от кода поставщика, но не от заказа.
   - Цена за единицу зависит только от оборудования, а не от конкретного заказа.
3. Нарушение 3НФ:
   - Поле "Общая стоимость" вычисляется из "Количество × Цена за единицу", что приводит к аномалиям при обновлении.

Приведем к 1 Нормальной форме (1НФ).

Требование: Каждое поле должно содержать только одно значение.

Разделяем повторяющиеся значения на отдельные строки.

Номер заказа	Код поставщика	Название поставщика	Наименование оборудования	Коли-чество	Цена за единицу
1	545	ГазМехСнаб	Газопровод	2	25000
1	545	ГазМехСнаб	Шаровый кран	1	15000
2	546	НефтеСервис	Газоанализатор	4	7000
2	546	НефтеСервис	Трубопровод	3	15000
3	547	ВодоГазСнаб	Фильтр	4	8000
3	547	ВодоГазСнаб	Газопровод	2	25000
4	548	ГазЭкспорт	Датчик давления	3	1500
4	548	ГазЭкспорт	Гайка	10	80

Теперь все поля содержат только одно значение, но проблемы 2НФ и 3НФ ещё остаются.

Приведем ко 2 Нормальной форме (2НФ).

Требование: каждый не ключевой атрибут должен зависеть от всего первичного ключа, а не от его части.

Выносим данные о поставщиках в отдельную таблицу "Поставщики" и выносим данные об оборудовании и ценах в отдельную таблицу "Оборудование".

Таблица "Заказы"

Номер заказа	Код поставщика
1	545
2	546
3	547
4	548

Таблица "Поставщики"

Код поставщика	Название поставщика
545	ГазМехСнаб
546	НефтеСервис
547	ВодоГазСнаб
548	ГазЭкспорт

Таблица "Оборудование"

Код оборудования	Наименование оборудования	Цена за единицу
101	Газопровод	25000
102	Шаровый кран	15000
103	Газоанализатор	7000
104	Трубопровод	15000
105	Фильтр	8000
106	Датчик давления	1500
107	Гайка	80

Таблица "Состав заказа"

Номер заказа	Код оборудования	Количество
1	101	2
1	102	1
2	103	4
2	104	3
3	105	4
3	101	2
4	106	3
4	107	10

Теперь все данные зависят только от полного ключа, но проблема 3НФ остаётся.

Приведем к 3 Нормальной форме (3НФ).Требование: не должно быть транзитивных зависимостей (то есть данные не должны зависеть от не ключевого поля).

Таблица "Заказы"

Номер заказа	Код поставщика
1	545
2	546
3	547
4	548

Таблица "Поставщики"

Код поставщика	Название поставщика
545	ГазМехСнаб
546	НефтеСервис
547	ВодоГазСнаб
548	ГазЭкспорт

Таблица "Оборудование"

Код оборудования	Цена за единицу
101	Газопровод
102	Шаровый кран
103	Газоанализатор
104	Трубопровод
105	Фильтр
106	Датчик давления
107	Гайка

Таблица " Цены на оборудование"

Код оборудования	Цена за единицу
101	25000
102	15000
103	7000
104	15000
105	8000
106	1500
107	80

Таблица "Состав заказа"

Номер заказа	Код оборудования	Количество
1	101	2
1	102	1
2	103	4
2	104	3
3	105	4
3	101	2
4	106	3
4	107	10

Исправление:

- В таблице "Оборудование" теперь хранится только код и цена.
- Общая стоимость теперь не хранится – её можно вычислять при необходимости (Количество × Цена за единицу).

Теперь данные полностью нормализованы, база данных гибкая и защищена от аномалий обновления.

Задание 8. Шифрование

5 баллов – Функция выдаёт корректный результат.

3 балла – Алгоритм работает правильно для всех входных данных, но не оформлен в виде функции. Не более двух синтаксических или одной содержательной ошибки.

1 балл – В комментариях присутствует хотя бы частичное описание логики программы, однако сама она не реализована или выдает неверный результат.

В нефтегазовой компании ведется разработка системы защиты данных, передаваемых между удаленными станциями добычи и центральным офисом. Для обеспечения целостности сообщений требуется метод их проверки.

Разработайте алгоритм, который вычисляет контрольную сумму.

Требования:

- Необходимо преобразовать входное сообщение в числовое значение, характеризующее целостность передаваемых данных.
- Преобразование должно учитывать кодовое представление каждого символа, которое кратное 3 или 2, а также его позицию в сообщении.
- Преобразование должно учитывать, только те символы чьи позиции четные.
- Результирующее значение, полученное путем объединения информации о символах и их позициях, должно быть приведено к числу в диапазоне от 0 до 9999 с использованием операции взятия по модулю 10 000.

Формат входных данных	“Hello”
Формат выходных данных	432

Решение задания. Функция calculate_check_sum вычисляет контрольную сумму строки, обрабатывая символы, находящиеся на четных позициях. Для каждого такого символа определяется его ASCII-код, затем выбирается множитель (3, если код кратен 3; 2, если кратен 2; иначе 1) и символ умножается на этот множитель и на его порядковый номер (i+1); итоговая сумма приводится к диапазону от 0 до 9999 с помощью операции взятия остатка от деления на 10000.

```python
def calculate_check_sum(message):
  checksum = 0

  # Обрабатываем только символы на четных позициях (0, 2, 4, ...)
  for i in range(0, len(message), 2):
    char_code = ord(message[i]) # Получаем ASCII-код символа

    # Определяем множитель
    if char_code % 3 == 0:
      multiplier = 3
    elif char_code % 2 == 0:
      multiplier = 2
    else:
      multiplier = 1 # Если не кратно 2 или 3

    # Учитываем индекс символа в сообщении
    checksum += char_code * multiplier * (i + 1)

  # Приводим значение к диапазону [0, 9999]
  # с помощью целочисленного деления с остатком
  return checksum % 10000
```

Задание 9. Теория игр

10 баллов – Оптимальный первый ход найден, доказано, что он гарантирует победу.

5 баллов – Оптимальный первый ход найден, но без формального доказательства.

1 балл – Логика частично верна, но ход неверный.

На нефтегазовом месторождении имеются два резервуара, в которых хранится ресурс:

- Резервуар 1: 17 единиц нефти,
- Резервуар 2: 11 единиц газа.

Два оператора, Владислав и Пётр, по очереди выполняют операции по извлечению ресурса из резервуаров. Для оптимизации процесса работы начальство установило следующие правила, в рамках которых внедрена система премий:

- Операторы выполняют действия поочередно.
- На каждом ходу оператор может извлечь любое количество ресурса (не менее одной единицы) из одного выбранного резервуара.
- Целью процесса является извлечение последнего ресурса.
- Оператор, который извлечет последний ресурс, получает премию.

Ограничения:

- Извлекать за один ход можно только из одного резервуара.
- Извлекать можно только целые единицы.

- За ход оператор может извлечь не менее 1 единицы и не более, чем осталось в резервуаре.

Необходимо определить первый ход для первого оператора, чтобы гарантировать победу, если они оба играют оптимально[1].

1 - Каждый оператор на каждом ходу выбирает наилучший возможный вариант, который, при наличии такой возможности, гарантирует победу.

Решение задания. Задача является игрой с двумя кучами, где игроки поочередно забирают элементы.

Для определения выигрышной стратегии используем XOR-сумму (ним-сумму) двух чисел: nim-sum = $R_1 \oplus R_2 = 17 \oplus 11$

Вычисляем Nim-сумму:

$17_{(10)} = 10001_{(2)}$

$11_{(10)} = 01011_{(2)}$

$17 \oplus 11 = 10001 \oplus 01011 = 11010_{(2)} = 26_{(10)}$

Так как Nim-сумма = $26 \neq 0$, первый игрок может гарантировать победу. Теперь надо определить первый выигрышный ход.

Чтобы выиграть, первый игрок должен уменьшить одну из куч так, чтобы новая Nim-сумма стала 0.

Ищем, какой ресурс (резервуар) можно уменьшить:

Для каждого возможного хода проверяем, можно ли изменить R1 или R2 так, чтобы результирующая Nim-сумма стала 0.

Проверяем резервуар 1 (нефть, 17 единиц)

Хотим найти X, такое что: $(17 - x) \oplus 11 = 0$

Решаем уравнение: $(17 - x) = 11 \oplus 26 = 21$

Тогда: $x = 17 - 21 = -4$

Отрицательное значение, значит, нельзя выиграть, уменьшая нефть.

Проверяем резервуар 2 (газ, 11 единиц)

Хотим найти Y, такое что: $17 \oplus (11 - y) = 0$

Решаем уравнение: $(11 - y) = 17 \oplus 26 = 9$

Тогда: $y = 11 - 9 = 2$

Если первый игрок забирает 2 единицы газа (из резервуара 2), он гарантирует победу.

Ответ. Итоговый первый ход для победы:

- Первый оператор (Владислав) должен забрать 2 единицы газа из резервуара 2.
- После этого игра переходит в проигрышное состояние для второго игрока (Петра), если оба играют оптимально.
- Владислав гарантированно выигрывает, если продолжит играть правильно.

Задание 10. Графы

5 баллов – Верный ответ с расчётами.
3 балла – Ход решения верный, допущена одна логическая ошибка.
1 балл – Ход решения верный, допущена две логические ошибки.

В группе регионов планируется строительство N месторождений нефти и газа, соединённых двунаправленными трубопроводами. Трубопроводы могут образовывать циклы[1]*, но сеть связная – между любыми двумя месторождениями существует хотя бы один путь. Каждое месторождение принадлежит ровно одному виду добычи. Требуется осуществить расчёты, чтобы выбрать по одному месторождению каждого типа и соединить их трубопроводами так, чтобы суммарная длина этих трубопроводов была минимальной.

Месторождения и трубопроводы	**Типы добычи**
A — B (3[2*]) B — C (5) C — D (7) D — E (4) E — F (6) A — F (8) B — E (10)	Нефтяные: [A, F] Газовые: [B, C] Конденсатные: [D, E]

1 — это путь, начинающийся и заканчивающийся в одной точке.*
2 — расстояние между A и B.*

Решение задания. Решение сводится к построению минимального остовного (минимального) дерева (MST) для соединения по одному месторождению каждого типа с минимальной суммарной длиной трубопроводов.

Для начала отсортируем ребра по возрастанию длины.

Трубопровод	Длина
A — B	3
D — E	4
B — C	5
E — F	6
C — D	7
A — F	8
B — E	10

Формируем MST (Алгоритм Краскала):

- A — B (3) (добавляем, соединяет A и B)
- D — E (4) (добавляем, соединяет D и E)
- B — C (5) (добавляем, соединяет C и B)
- E — F (6) (добавляем, соединяет E и F)
- C — D (7) (добавляем, соединяет C и D)

Строим результирующее минимальное остовное дерево (MST).

Включенный трубопровод	Длина
A — B	3
B — C	5
C — D	7
D — E	4
E — F	6

Суммарная длина MST: 3 + 5 + 7 + 4 + 6 = 25
Выбираем по одному месторождению каждого типа.

Тип добычи	Выбранное месторождение
Нефтяное	A
Газовое	B
Конденсатное	D

Выбираем минимальный путь между A, B и D в MST

Путь	Длина
A — B	3
B — C	5
C — D	7

Суммарная длина соединения месторождений разного типа: 3 + 5 + 7 = 15

Ответ. Получены следующие выводы:

- Выбираем месторождения: A (нефть), B (газ), D (конденсат).
- Соединяем их минимальным путём: A → B → C → D.
- Минимальная суммарная длина трубопроводов = 15.