# Технологии виртуальной реальности

2022/23 учебный год

# Заключительный этап

# Предметный тур

## Информатика. 8–11 класс

### *Задача VI.1.1.1. Поиск трапеции (100 баллов)*

**Имя входного файла**: стандартный ввод.

**Имя выходного файла**: стандартный вывод.

**Ограничение по времени выполнения программы**: 1 с.

**Ограничение по памяти**: 512 Мбайт.

#### *Условие*

Ваня купил себе VR-гарнитуру и решил поиграть. Для начала ему необходимо разметить VR-зону в комнате. Для этого Ваня хочет использовать изоленту. У него уже есть $N$ отрезанных кусков длиной $a_i$. VR-зона должна иметь форму *прямоугольной трапеции*. Каждая сторона трапеции должна быть образована ровно одним куском изоленты.

#### *Формат входных данных*

Первая строка содержит единственное число $N$. Следующие $N$ строк содержат целые числа — длины отрезков $a_i$.

#### *Формат выходных данных*

Выведите 4 индекса отрезков в порядке возрастания или $-1$, если невозможно получить прямоугольную трапецию. Индексация начинается с нуля.

Если существует несколько ответов, выведите трапецию с максимальной площадью, а среди таких — с минимальным первым индексом.

#### *Ограничения*

$1 \leqslant N \leqslant 40$, $1 \leqslant a_i \leqslant 100$.

### *Примеры*

*Пример №1*

| Стандартный ввод |
| --- |
| 5<br>12<br>10<br>3<br>7<br>23 |
| **Стандартный вывод** |
| -1 |

*Пример №2*

| Стандартный ввод |
| --- |
| 6<br>14<br>12<br>16<br>21<br>15<br>25 |
| **Стандартный вывод** |
| 1 2 4 5 |

### *Решение*

Пусть дана прямоугольная трапеция с основаниями $a$ и $b$ и боковыми сторонами $c$ и $d$, причём $a \leqslant b$ и $c \leqslant d$. Тогда должно выполняться условие $(b-a)^2 + c^2 = d^2$. Поскольку $N$ невелико, можно перебрать все упорядоченные комбинации данных отрезков и для каждой проверить выполнение условия.

Если условие выполнено для нескольких наборов, требуется определить трапецию с минимальной площадью (площадь трапеции равна $1/2(a+b)c$, поскольку более короткая сторона в данном случае равна высоте), а среди таких — с минимальным индексом стороны.

Алгоритмическая сложность составляет $O(N^4)$.

Данную задачу можно решить и более эффективно, однако при указанных ограничениях этого не требуется.

### *Пример программы-решения*

Ниже представлено решение на языке C++.

```
1  #include <iostream>
2  #include <cmath>
3  #include <set>
4  #include <map>
```

```cpp
#include <unordered_map>
#include <unordered_set>
#include <vector>
#include <iomanip>
#include <algorithm>
#include <queue>
#include <string>

#define MOD 1000000007
#define INF 1000000008
#define all(arr) arr.begin(), arr.end()
#define rall(arr) arr.rbegin(), arr.rend()

#define ll long long
#define ull unsigned long long
#define double long double

#define fast ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0)
#define files(input, output) freopen(input, "r", stdin); freopen(output, "w", stdout)

using namespace std;

template<class T>
istream &operator>>(istream &in, vector<T> &arr) {
    for (T &i: arr)
        in >> i;
    return in;
}
template<class T>
ostream &operator<<(ostream &out, const vector<T> &arr) {
    for (const T &i: arr)
        out << i << ' ';
    return out;
}

bool check_trap(const vector<ll>& trap){
    ll tmp = trap[0] - trap[1];
    return tmp * tmp == trap[3] * trap[3] - trap[2] * trap[2];
}

ll area(const vector<ll> & trap){
    return (trap[0] + trap[1]) * trap[2];
}

void solve() {
    int n;
    cin >> n;
    vector<ll> segments(n);
    cin >> segments;
    ll s = 0;
    vector<ll> res(4, INF), ntrap(4), nres(4);
    for (int i = 0; i < n; ++i){
        for (int j = 0; j < n; ++j){
            if (i == j) continue;
            for (int k = 0; k < n; ++k){
                if (i == k || j == k) continue;
                for (int m = 0 ; m < n; ++m){
                    if (m == i || m == j || m == k) continue;
                    ntrap[0] = segments[i];
```

```
                        ntrap[1] = segments[j];
                        ntrap[2] = segments[k];
                        ntrap[3] = segments[m];
                        if (check_trap(ntrap)){
                            ll ns = area(ntrap);
                            nres[0] = i;
                            nres[1] = j;
                            nres[2] = k;
                            nres[3] = m;
                            sort(nres.begin(), nres.end());
                            if (ns > s) {
                                s = ns;
                                res = nres;
                            }
                            else if (ns == s){
                                bool change = false;
                                for (int l = 0; l < 4; ++l){
                                    if (res[l] == nres[l]) continue;
                                    if (res[l] < nres[l]){ change = false; break;}
                                    change = true; break;
                                }
                                if (change){
                                    res = nres;
                                }
                            }
                        }
                    }
                }
            }
        }
        if (s == 0){
            cout << -1;
        }
        else{
            cout << res[0] << ' ' << res[1] << ' ' << res[2] << ' ' << res[3];
        }
    }

    signed main() {
        fast;
        cout.precision(20);
        int t = 1;
        while (t--)
            solve();
    }
```

## *Задача VI.1.1.2. Экзамен (100 баллов)*

**Имя входного файла**: стандартный ввод.

**Имя выходного файла**: стандартный вывод.

**Ограничение по времени выполнения программы**: 1 с.

**Ограничение по памяти**: 256 Мбайт.

### *Условие*

Артему предстоит сложный экзамен по алгебре, геометрии и физике. Преподаватель на лекции сообщил, что студентам на выбор будет предложено $n$ билетов и скинул всем их на электронную почту. Артем оценил каждый билет по трем параметрам: сложность его по алгебре, геометрии и физике. Сам Артем оценил свои силы следующим образом: если сложность билета по алгебре, геометрии и физике меньше чем $a$, $b$ и $c$ соответственно, он способен к нему подготовится и сдать.

Так как Артем был прилежным студентом, преподаватель готов принять у него экзамен, пропустив один из предметов по его выбору: алгебру, геометрию или физику. Студент не против воспользоваться такой возможностью, поэтому просит у вас узнать, какой предмет ему стоит пропустить, чтобы можно было решить как можно больше билетов.

### *Формат входных данных*

В первой строке записано три целых числа $a$, $b$ и $c$ — предельные пороги сложности по алгебре, геометрии и физике соответственно.

Во второй строке целое число $n$ — количество билетов на экзамене.

Во следующих $n$ строках записано по три целых числа $a_i$, $g_i$, $p_i$ — сложность билета по алгебре,геометрии и физике соответственно.

### *Формат выходных данных*

Выведите `Algebra`, `Geometry` или `Physics` в соответствии с тем, какой экзамен выгоднее пропустить. Если существует несколько вариантов ответа, выведите любой.

### *Ограничения*

$1 \leqslant n \leqslant 10^5$.

$1 \leqslant a, b, c, a_i, b_i, c_i \leqslant 10^9$.

### *Примеры*

*Пример №1*

| **Стандартный ввод** |
|---|
| 3 3 3<br>3<br>2 4 5<br>2 2 5<br>1 1 4 |

| **Стандартный вывод** |
|---|
| Physics |

*Пример №2*

| Стандартный ввод |
| --- |
| 5 2 4<br>5<br>1 1 1<br>2 2 2<br>3 3 3<br>4 4 4<br>5 5 5 |

| Стандартный вывод |
| --- |
| Geometry |

### *Решение*

Данная задача является тривиальной и имеет линейное решение $O(n)$.

Посчитаем для каждого предмета, сколько билетов можно сдать успешно, пропустив выбранный предмет. Подсчет будем осуществлять следующим образом: если пропускаем а (алгебра), считаем количество билетов, для которых $b > b_i$ (геометрия) и $c > c_i$ (физика). Для остальных предметов делается также.

Далее мы должны найти и вывести тот предмет, при котором получилось максимальное количество сданных билетов.

### *Пример программы-решения*

Ниже представлено решение на языке C++.

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

struct exam {
    int a;
    int b;
    int c;
};

int main() {
    int a, b, c, n, resA = 0, resB = 0, resC = 0;
    cin >> a >> b >> c >> n;
    vector<exam> arr(n);
    for(int i = 0; i < n; i++)
        cin >> arr[i].a >> arr[i].b >> arr[i].c;
    for(int i = 0; i < n; i++) {
        if(b >= arr[i].b && c >= arr[i].c)  resA++;
        if(a >= arr[i].a && c >= arr[i].c)  resB++;
        if(a >= arr[i].a && b >= arr[i].b)  resC++;
    }
    int maxValue = max(resA, max(resB, resC));
    cout << "Possible answers: ";
    if(maxValue == resA)
```

```
27          cout << "Algebra ";
28      if(maxValue == resB)
29          cout << "Geometry ";
30      if(maxValue == resC)
31          cout << "Physics ";
32  }
```

## *Задача VI.1.1.3. Радостные студенты (100 баллов)*

**Имя входного файла**: стандартный ввод.

**Имя выходного файла**: стандартный вывод.

**Ограничение по времени выполнения программы**: 1 с.

**Ограничение по памяти**: 512 Мбайт.

### *Условие*

На лекции по высшей математике в ДВФУ преподаватель собрал $n$ студентов в ряд и задал простой вопрос: «*Кто сейчас грустит, поднимите руку*». На это предложение несколько (возможно, ноль) студентов подняли руки.

После этого он решил выбрать из этой последовательности студентов некоторый отрезок `[left, right]`, на котором он добавит грустным студентам по 5 баллов к экзамену просто так. При этом он понимает, что студенты, которые были радостные на этом отрезке, поменяют своё настроение. Если ни один студент не является грустным, преподаватель не будет выбирать никакой отрезок. Он хочет получить наибольшее количество радостных студентов на лекции, поэтому просит вас написать программу, которая рассчитает максимальное их количество после применения ранее описанной операции.

### *Формат входных данных*

В первой строке записано целое число $n$ — количество студентов.

Во второй строке записано $n$ цифр 0 и 1, где 0 — грустный студент, а 1 — радостный.

### *Формат выходных данных*

Выведите максимально возможное количество радостный студентов.

### *Ограничения*

$1 \leqslant n \leqslant 2 \cdot 10^5$.

### *Критерии оценивания*

Баллы начисляются за каждый тест независимо. Тесты поделены по подзадачам, описанным ниже.

| Подзадача | Количество тестов | Баллы | Дополнительные ограничения | Информация о проверке |
|---|---|---|---|---|
| | | | n | |
| 1 | 5 тестов | 5 баллов за тест | $1 \leqslant n \leqslant 200$ | полная |
| 2 | 5 тестов | 5 баллов за тест | $1 \leqslant n \leqslant 2000$ | полная |
| 3 | 10 тестов | 5 баллов за тест | $1 \leqslant n \leqslant 2 \cdot 10^5$ | полная |

***Примеры***

*Пример №1*

| **Стандартный ввод** |
|---|
| 7<br>1 1 1 0 0 1 0 |
| **Стандартный вывод** |
| 6 |

*Пример №2*

| **Стандартный ввод** |
|---|
| 5<br>1 1 1 1 1 |
| **Стандартный вывод** |
| 5 |

***Решение***

Идея решения данного задания — линейный поиск максимальной суммы подотрезка.

Посчитаем первоначальное количество радостных студентов. Следующим шагом (или параллельно) создадим массив, содержащий изменения настроения студентов, если они войдут в подотрезок профессора. Радостных студентов обозначим как $-1$ (настроение станет негативным), а грустных как 1 (настроение станет позитивным).

Следующим шагом создадим две переменные, в которых будут хранится сумма изменений студентов на некотором отрезке $[1, i]$ и сумма изменений настроений на некотором отрезке $[1, j]$, $j < i \leqslant n$. Параметр $j$ будет на каждом шагу выбираться так, чтобы сумма отрезка $[1, j]$ являлась наименьшей на отрезке $[1, i]$.

Отрезок минимальной суммы будет убираться из отрезка общей суммы $sum[1, i] - sum[1, j]$, тем самым мы сможем получить наибольшую сумму изменений настроений студентов. Так как мы изначально сделали радостных студентов $-1$, а грустных 1, процесс вычисления будет стараться избегать случаем, когда изменится настроение у позитивных студентов, но при этом будет искать наибольшее количество грустных студентов и компенсировать настроение новоиспеченных грустных студентов.

Среди всех подотрезков выберем тот, у которого наибольшее изменение настроений, добавим к уже ранее высчитанным радостным студентам, и выведем полученную сумму. Данный результат будет верным, ведь если изменится настроение у

радостного студента, наш подотрезок будет это учитывать. Например, если на подотрезке 3 грустных студента и 1 радостный, то сумма настроений на подотрезке будет 2 (двое станут новыми радостными, а оставшиеся два студента, радостный и грустный, компенсируют свои настроения).

### *Пример программы-решения*

Ниже представлено решение на языке C++.

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main() {
    int n;
    cin >> n;
    vector<int> arr (n);
    int res = 0;
    for(auto &i : arr) {
        cin >> i;
        if(i == 0) {
            i = 1;
        } else {
            i = -1;
            res++;
        }
    }
    int ans = 0;
    int sum = 0;
    int minSum = 0;
    for(int i = 0; i < n; i++) {
        sum += arr[i];
        ans = max(ans, sum - minSum);
        minSum = min(minSum, sum);
    }
    cout << res + ans;
}
```

## *Задача VI.1.1.4. Подготовка к ЕГЭ (100 баллов)*

**Имя входного файла**: стандартный ввод.

**Имя выходного файла**: стандартный вывод.

**Ограничение по времени выполнения программы**: 1 с.

**Ограничение по памяти**: 512 Мбайт.

### *Условие*

Мальчик Миша готовится к экзаменам. На это у него осталось $N$ дней. В $i$-ый день у Миши вдохновение решить $a_i$ задач. Но он не сверхчеловек, поэтому ему необходимо спать. В $i$-ый день у Миши есть выбор:

- Поспать, не решив ни одной задачки.

- Выпить чай с лимонником и не спать, решив все задачи.
- Не спать, решив все задачи.

Не спать он может только в том случае, если у него достаточно сил, то есть если в предыдущий день он поспал.

Также у него есть замечательный напиток — чай с лимонником, который даст ему сил не спать. Но при этом Миша знает, что избыток чая вреден для здоровья, поэтому он не станет его пить, если делал это в предыдущий день.

### *Формат входных данных*

В первой строке записано целое число $N$.

Во второй строке находится $N$ целых чисел $a_i$.

### *Формат выходных данных*

Выведите одно целое число: максимальное количество задач, которые может решить Миша.

### *Ограничения*

$1 \leqslant N \leqslant 2 \cdot 10^5$.

$1 \leqslant a_i \leqslant 10^6$.

### *Критерии оценивания*

Баллы начисляются за каждый тест независимо. Тесты поделены по подзадачам, описанным ниже.

| **Подзадача** | **Количество тестов** | **Баллы** | **Дополнительные ограничения** | **Информация о проверке** |
|---|---|---|---|---|
| | | | **N** | |
| 1 | 5 тестов | 4 балла за тест | $1 \leqslant N \leqslant 20$ | полная |
| 2 | 5 тестов | 4 балла за тест | $1 \leqslant N \leqslant 5 \cdot 10^4$ | полная |
| 3 | 10 тестов | 4 балла за тест | $1 \leqslant N \leqslant 2 \cdot 10^5$ | полная |

### *Примеры*

*Пример №1*

| **Стандартный ввод** |
|---|
| 3<br>1 2 3 |

| **Стандартный вывод** |
|---|
| 5 |

*Пример №2*

| **Стандартный ввод** |
| --- |
| 5<br>1 1 1 1 1 |
| **Стандартный вывод** |
| 4 |

### *Решение*

Решим задачу при помощи динамического программирования. Создадим 3 массива длины $N$, где в первом на $i$-ой позиции будет храниться максимальное количество задач, решенное к $i$-ому дню включительно, если в этот день Миша решит поспать. Во втором — если решит попить чай. А в третьем — если решит не спать.

База динамики: в первом массиве на первой позиции будет стоять 0, так как Миша будет спать и не решит задач. Во втором и третьем массивах на первой позиции будет стоять $a_1$.

Далее для всех $i$ от 2 до $N$ на $i$-ой позиции:

- первого массива — максимум из значений всех массивов на позиции $i - 1$;
- второго массива — сумма $a_i$ и максимума из значений первого и третьего массивов на позиции $i - 1$;
- третьего массива — сумма значения первого массива на $i - 1$ позиции и $a_i$.

Ответом будет максимум из элементов на $N$-ой позиции всех трёх массивов.

### *Пример программы-решения*

Ниже представлено решение на языке C++.

```cpp
#include "iostream"
#include "vector"

using namespace std;

int main() {
    int N;
    cin >> N;
    vector<long long> a(N);
    for (int i = 0; i < N; ++i) {
        cin >> a[i];
    }
    vector<vector<long long>> dp(N, vector<long long>(3, 0)); //спать, чай, не спать
    dp[0][0] = 0;
    dp[0][1] = a[0];
    dp[0][2] = a[0];
    for (int i = 1; i < N; ++i) {
        dp[i][0] = max(max(dp[i - 1][0], dp[i - 1][1]), dp[i - 1][2]);
        dp[i][1] = max(dp[i - 1][0], dp[i - 1][2]) + a[i];
        dp[i][2] = dp[i - 1][0] + a[i];
    }
    cout << max(dp[N - 1][0], max(dp[N - 1][1], dp[N - 1][2]));
}
```

## *Задача VI.1.1.5. Путешествие (100 баллов)*

**Имя входного файла**: стандартный ввод.

**Имя выходного файла**: стандартный вывод.

**Ограничение по времени выполнения программы**: 1 с.

**Ограничение по памяти**: 512 Мбайт.

### *Условие*

Утенок Даки только выпустился из университета и устроился на работу разработчиком игр. Ему поручили создание новой игры SpaceWar. Игра заключается в сражениях с космическим флотом противника путём отправки флотов кораблей между планетами.

Даки закончил разработку и решил проверить качество игры самостоятельно. В ходе проверки утенок понял, что в игре играет роль не только мощь флота, но и распределение его между планетами, чтобы вражеский флот не смог захватывать территории. Даки решил опробовать новую тактику и, несмотря на опасность потери планет, не разделять свой флот. Так он сможет с легкостью захватывать планеты. Но такая тактика имеет большой недостаток: во время нападения на новые планеты собственные территории беззащитны. Поэтому утенок хочет рассчитывать время, за которое его флот сможет добраться до планеты.

В этой игре между двумя планетами иногда появляются «кротовые норы» (англ. wormholes), пройдя через которые, флот может переместиться вперед во времени и оказаться у следующей планеты. В то же время может также существовать «обычный» путь между планетами, который флот может преодолеть за некоторое время. Существование кротовых нор и обычного пути не связаны между собой. Кротовые норы появляются не сразу и, если флот окажется у планеты, нора около которой еще не образовалась, он не сможет ею воспользоваться. Воспользоваться норой в обратном направлении невозможно. Даки хочет определить, в какой самый ранний момент времени его флот, находящийся у планеты $A$, может оказаться у планеты $B$.

### *Формат входных данных*

Первая строка содержит 3 целых числа: $N$, $A$, $B$ — количество планет, планету с флотом и планету-цель, соответственно.

Далее следует строка с 2 целыми числами: $M$ — количество кротовых нор и $K$ — количество обычных путей.

Последующие $M$ строк содержат 4 целых числа: $A_i$, $B_i$ — две планеты, между которыми появляется нора, $t_i$ — время её появления и $dt_i$ — насколько изменится время при перемещении по ней из $A_i$ в $B_i$.

Далее идут $K$ строк в таком формате: $A_j$, $B_j$, $t_j$ — две планеты и время пути между ними.

Считаем, что изначально время равно 0. Все планеты нумеруются от 1 до $N$ включительно. Обычные пути существуют в любое время.

### *Формат выходных данных*

Выведите единственное число — самое раннее время в которое флот сможет оказаться у планеты $B$. Гарантируется, что путь существует.

### *Ограничения*

$1 \leqslant A, B, A_i, B_i, A_j, B_j \leqslant N \leqslant 10^4$

$N \leqslant M + K \leqslant 10^5$

$0 \leqslant t_i, dt_i, t_j \leqslant 10^9$

### *Примеры*

*Пример №1*

| **Стандартный ввод** |
|---|
| 6 3 5<br>3 6<br>6 3 0 0<br>1 3 2 3<br>2 1 0 1<br>3 5 3<br>1 6 2<br>5 1 4<br>3 6 0<br>5 2 1<br>2 4 2 |

| **Стандартный вывод** |
|---|
| 3 |

*Пример №2*

| **Стандартный ввод** |
|---|
| 5 3 2<br>0 8<br>3 2 4<br>1 4 1<br>5 2 2<br>5 3 5<br>1 5 3<br>2 4 1<br>4 1 3<br>4 3 2 |

| **Стандартный вывод** |
|---|
| 4 |

### *Решение*

Для решения задачи можно воспользоваться модифицированным алгоритмом Дейкстры. Длина пути в данном случае эквивалентна самом раннему времени при-

бытия.

Напомним, что в этом алгоритме вершины почерёдно помечаются как обработанные, что означает что текущее найденное расстояние до них является кратчайшим.

На этапе релаксации пути помимо рёбер, исходящих из только что помеченной вершины, следует также рассмотреть новые рёбра, образовавшиеся при появлении «кротовых нор». При этом в качество новой длины пути следует брать сумму времени перемещения по норе и минимума из времени прибытия в помеченной вершину и времени появления норы.

Для повышения эффективности выбора помечаемой вершины можно использовать бинарную кучу.

### *Пример программы-решения*

Ниже представлено решение на языке C++.

```
#include <iostream>
#include <cmath>
#include <set>
#include <map>
#include <unordered_map>
#include <unordered_set>
#include <vector>
#include <iomanip>
#include <algorithm>
#include <queue>
#include <string>

#define MOD 1000000007
#define INF 1000000008
#define all(arr) arr.begin(), arr.end()
#define rall(arr) arr.rbegin(), arr.rend()

#define ll long long
#define ull unsigned long long
#define double long double

#define fast ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0)
#define files(input, output) freopen(input, "r", stdin); freopen(output, "w",
↪  stdout)

using namespace std;

template<class T>
istream &operator>>(istream &in, vector<T> &arr) {
    for (T &i: arr)
        in >> i;
    return in;
}

template<class T1, class T2>
istream &operator>>(istream &in, pair<T1, T2> &p) {
    in >> p.first >> p.second;
    return in;
}
struct hole{
    int from;
```

```
    int to;
    int t;
    int dt;
};
void solve() {
    int n, a, b;
    cin >> n >> a >> b;
    --a; --b;
    vector<vector<pair<int, int>>> ways(n, vector<pair<int, int>>());
    vector<vector<hole>> holes(n, vector<hole>());
    vector<ull> d(n, INF);
    d[a] = 0;
    vector<pair<ull, int>> heap = {{0,a}};
    make_heap(heap.begin(), heap.end(), std::greater<>{});
    int m, k;
    cin >> m >> k;
    for (int i = 0; i < m; ++i){
        hole tmp{};
        cin >> tmp.from>> tmp.to >> tmp.t >> tmp.dt;
        --tmp.from; --tmp.to;
        holes[tmp.from].push_back(tmp);
    }
    for (int j = 0; j < k; ++j){
        int a_j, b_j, tmp;
        cin >> a_j >> b_j >> tmp;
        --a_j; --b_j;
        ways[a_j].emplace_back(b_j, tmp);
        ways[b_j].emplace_back(a_j, tmp);
    }
    while (d[b] == INF) {
        if (heap.empty())
            break;
        pop_heap(heap.begin(), heap.end(), std::greater<>{});
        pair<ull, int> v = heap.back();
        heap.pop_back();
        if (d[v.second] != INF && v.second != a) continue;
        d[v.second] = v.first;
        for (const auto &w: ways[v.second]) {
            heap.emplace_back(d[v.second] + w.second, w.first);
            push_heap(heap.begin(), heap.end());
        }
        for (const auto &h: holes[v.second]) {
            ll tmp;
            if (h.t <= d[v.second]) {
                tmp = d[v.second] + h.dt;
            } else {
                tmp = (ll) h.t + h.dt;
            }
            heap.emplace_back(tmp, h.to);
            push_heap(heap.begin(), heap.end());
        }
    }
    cout << d[b];
}

signed main() {
    fast;
    cout.precision(20);
    int t = 1;
    while (t--)

        solve();
}
```

# Математика. 8–9 классы

## *Задача VI.1.2.1. (15 баллов)*

*Темы: графы, задача на минимум.*

Под застройку жилого массива выделен участок земли, на котором по плану необходимо построить 10 жилых кварталов и 12 общественных мест (поликлиники, школы, места отдыха и т. д.). Также по плану в новом районе не будет перекрёстков: с одной улицы на другую можно будет попасть только через жилой квартал или общественное место. Какое минимальное количество улиц необходимо проложить, чтобы любой житель этого района мог попасть из своего квартала в любое общественное место не посещая других кварталов, и при этом имел возможность сходить в гости в любой из кварталов не посещая общественных мест?

### *Решение*

План нового района можно представить в виде графа, в котором вершинами являются жилые кварталы (назовём их вершинами типа $A$) и общественные места (назовём их вершинами типа $B$), а рёбрами — улицы. Требуется, чтобы из любой вершины типа $A$ можно было попасть в любую вершину типа $B$ без посещения других вершин типа $A$. Это значит, во-первых, что из любой вершины типа $A$ ведёт ребро в подграф из вершин типа $B$ (уже имеем как минимум 10 рёбер в графе). Во-вторых, подграф из вершин типа $B$ — связный, следовательно, имеет как минимум $12 - 1 = 11$ рёбер.

По условию требуется также, чтобы подграф из вершин типа $A$ был связным, следовательно, он должен содержать как минимум $10 - 1 = 9$ рёбер. Таким образом, всего должно быть как минимум $10 + 11 + 9 = 30$ рёбер. Несложно привести пример такого графа.

**Ответ:** 30.

### *Критерии оценивания*

Отмечено, что подграфы, соответствующие жилым кварталам и общественным местам, должны быть связными, но неверно найдено количество рёбер в них — 5 баллов.

## *Задача VI.1.2.2. (20 баллов)*

*Темы: теория чисел, остатки.*

На какую цифру оканчивается сумма тридцати произвольно выбранных последовательных натуральных чисел, возведенных в восьмую степень?

### *Решение*

Число, оканчивающееся на цифру $d$, после возведения в натуральную степень $k$ будет оканчиваться на ту же цифру, на которую оканчивается число $d^k$. Найдем последние цифры для восьмых степеней цифр, учитывая, что $x^8 = \left((x^2)^2\right)^2$.

| Цифра $x$ | Последняя цифра в $x^8$ |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 6 |
| 3 | 1 |
| 4 | 6 |
| 5 | 5 |
| 6 | 6 |
| 7 | 1 |
| 8 | 6 |
| 9 | 1 |

Сумма имеет последнюю цифру 3. В ряду из последовательных 30 натуральных чисел каждая цифра встретится 3 раза в качестве последней. Поэтому ответом будет $3 \cdot 3 = 9$.

**Ответ:** 9.

***Критерии оценивания***

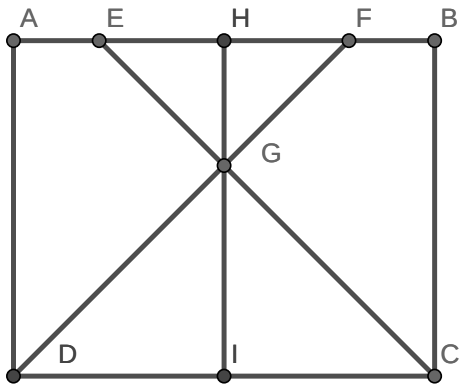
- Предложено рассматривать один десяток — 4 балла.
- Найдены последние цифры одного десятка — 12 баллов.
- Найдена последняя цифра суммы одного десятка — 16 баллов.

## *Задача VI.1.2.3. (20 баллов)*

*Темы: планиметрия, геометрическая вероятность.*

В прямоугольнике $ABCD$ биссектрисы углов $C$ и $D$ пересекают сторону $AB$ в точках $E$ и $F$ соответственно. Найдите вероятность того, что точка, случайным образом брошенная в прямоугольник $ABCD$, попадет в четырехугольник $AEGD$, если $G$ — точка пересечения биссектрис $CE$ и $DF$, а сторона прямоугольника $AB$ в 1,5 раза больше отрезка $EF$.

***Решение***




Через $S_X$ будем обозначать площадь фигуры $X$. Искомая вероятность равна

$$p = \frac{S_{AEGD}}{S_{ABCD}}.$$

Имеем $S_{AEGD} = S_{AFD} - S_{GEF}$.

Так как $\angle FEG = \angle GCD$, $\angle EFG = \angle GDC$, то треугольники $GEF$ и $GDC$ подобны с коэффициентом подобия $k = \dfrac{DC}{EF} = 1{,}5$. Пусть $GH$ и $GI$ — высоты треугольников $GEF$ и $GDC$ соответственно. Положим $h = GH$. Тогда $GI = kh = 1{,}5h$. Поэтому $AD = h + 1{,}5h = 2{,}5h$.

Так как $\Delta DGI = \Delta IGC$, то $DI = IC = AH = HB$. Кроме того, $\Delta EHG = \Delta HGF$, поэтому $EH = HF$. Отсюда, если $EF = x$, то $AF = \dfrac{1{,}5x}{2} + \dfrac{x}{2} = \dfrac{5}{4}x$.

Таким образом,

$$S_{GEF} = \frac{1}{2}hx,$$
$$S_{DAF} = \frac{1}{2}AD \cdot AF = \frac{1}{2} \cdot 2{,}5h \cdot \frac{5}{4}x = \frac{25}{16}hx.$$

Следовательно,

$$p = \frac{S_{DAF} - S_{GEF}}{S_{ABCD}} = \frac{\frac{25}{16}hx - \frac{1}{2}hx}{2{,}5h \cdot 1{,}5x} = \frac{17}{60}.$$

**Ответ:** $\dfrac{17}{60}$.

***Критерии оценивания***

- Площадь треугольника $ADF$ выражена через площадь прямоугольника $ABCD$ — 5 баллов.
- Площадь треугольника $EGF$ выражена через площадь прямоугольника $ABCD$ — 5 баллов.

## *Задача VI.1.2.4. (20 баллов)*

*Темы: алгебра, текстовая задача.*

Для нормальной работы бензопилы требуется заправлять её топливной смесью из масла и бензина в пропорции 1 : 34 (1 л масла на 34 л бензина). Объём бака бензопилы — 729 мл. Перед первым использованием бензопилу полностью заправили такой смесью, после чего израсходовали треть топлива в баке при распилке брёвен. Перед каждым следующим использованием бензопилу заправляли до полного бака чистым бензином, после чего снова расходовали треть бака при распилке. Перед 7-м использованием было решено заправить бензопилу до полного бака так, чтобы восстановить правильную пропорцию масла и бензина. Какой объём масла для этого потребуется?

***Решение***

Обозначим через $V$ объём бака. Пусть $n_j$ — концентрация масла в топливной смеси (отношение объёма масла к объёму смеси) после $j$-й заправки (или, что то же, после $j$-го использования). Имеем

$$n_1 = \frac{1}{35}.$$

Перед 2-м использованием смесь разбавляют чистым бензином: в объёме $V$ после заправки содержится $\frac{2}{3}Vn_1$ л масла. Поэтому после 2-й заправки концентрация масла

$$n_2 = \frac{\frac{2V}{3}n_1}{V} = \frac{2}{3}n_1.$$

Аналогично, после 3-й заправки концентрация масла

$$n_3 = \frac{\frac{2V}{3}n_2}{V} = \frac{2}{3}n_2 = \left(\frac{2}{3}\right)^2 n_1.$$

И так далее. После $j$-й заправки (и после $j$-го использования) концентрация масла

$$n_j = \left(\frac{2}{3}\right)^{j-1} n_1.$$

Тогда перед 7-й заправкой (после 6-го использования) концентрация масла в баке равна $n_6$. Пусть $c$ — концентрация масла в заливаемой топливной смеси перед 7-м использованием. Тогда необходимо

$$\frac{\frac{V}{3}c + \frac{2V}{3}n_6}{V} = n_1.$$

Отсюда $c = 3\left(n_1 - \frac{2}{3}n_6\right) = 3\left(n_1 - \left(\frac{2}{3}\right)^6 n_1\right)$. Следовательно, необходимый объём масла

$$\frac{V}{3}c = Vn_1\left(1 - (2/3)^6\right) = 729 \cdot \frac{1}{35}\left(1 - \frac{2^6}{3^6}\right) = 19 \text{ мл.}$$

**Ответ:** 19 мл.

***Критерии оценивания***

- Получена формула для концентрации $n_j$ — 4 балла.
- Вместо 7-го дан ответ для 6-го или 8-го использования — 12 баллов.
- Допущена арифметическая ошибка, если всё остальное верно — 18 баллов.

## *Задача VI.1.2.5. (25 баллов)*

*Темы: комбинаторика, арифметика.*

Из цифр $0, 2, 3, 5$ составляют всевозможные четырёхзначные числа так, что цифры в каждом из них не повторяются. Найдите разность средних арифметических чётных и нечётных чисел.

***Решение***

Будем нумеровать разряды числа справа налево начиная с нуля (то есть нулевой разряд — количество единиц, первый разряд — количество десятков и т. д.). Заметим, что составленные по условию числа имеют в нулевом разряде одну из четырёх цифр: 0, 2, 4, 6.

Все составленные числа можно разбить на две группы: с нулём либо не нулём в конце (в нулевом разряде). Первых будет 7!, а вторых — $3 \cdot 6 \cdot 6!$ (в 0-й разряд можно записать одну из трёх цифр, в 7-й — любую из оставшихся, кроме нуля, а в остальные 6 разрядов — любую из оставшихся шести цифр). Значит, всего составленных чисел будет

$$7! + 3 \cdot 6 \cdot 6! = 25 \cdot 6!.$$

Теперь вычислим сумму всех чисел. Будем действовать так же, как при вычислении суммы «в столбик»: сначала найдём сумму единиц, затем десятков и т. д., после чего сложим полученные результаты, умноженные на соответствующие степени десяти.

Количество чисел с цифрой 2 в нулевом разряде равно $6 \cdot 6!$ (цифра в 7-м разряде отлична от 0). Столько же чисел, оканчивающихся на 4 и на 6. Поэтому сумма единиц равна

$$(2 + 4 + 6) \cdot 6 \cdot 6! = 72 \cdot 6!.$$

Подсчитаем количество десяток. Пусть в 1-м разряде записана цифра 1. В нулевом разряде может быть записан 0 — таких чисел 6!, либо не 0 — таких чисел $3 \cdot 5 \cdot 5!$ (в нулевом разряде одна из трёх цифр 2, 4, 6, в 7-м разряде не 0, в остальных разрядах любые из пяти оставшихся цифр). Тогда всего таких чисел $6! + 3 \cdot 5 \cdot 5! = 21 \cdot 5!$. Столько же чисел, имеющих в 1-м разряде цифры 3, 5 и 7.

Пусть в 1-м разряде записана цифра 2. В нулевом разряде может быть записан 0 — таких чисел 6!, либо не 0 — таких чисел $2 \cdot 5 \cdot 5!$ (в нулевом разряде одна из двух цифр 4, 6, в 7-м разряде не 0, в остальных разрядах любые из пяти оставшихся цифр). Тогда всего таких чисел $6! + 2 \cdot 5 \cdot 5! = 16 \cdot 5!$. Столько же чисел, имеющих в 1-м разряде цифру 4 и 6. Поэтому сумма десятков равна

$$(1 + 3 + 5 + 7) \cdot 21 \cdot 5! + (2 + 4 + 6) \cdot 16 \cdot 5! = 16 \cdot 5! \cdot (21 + 12) = 88 \cdot 6!.$$

Количество сотен, тысяч, ..., миллионов вычисляется так же, как количество десяток.

Подсчитаем количество десятков миллионов. Если в 7-м разряде записана цифра 1, то таких чисел $4 \cdot 6!$ (в 0-м разряде — одна из четырёх цифр 0, 2, 4, 6, в остальных разрядах — любые из шести оставшихся). Столько же чисел, имеющих в 7-м разряде цифры 3, 5 и 7.

Если в 7-м разряде записана цифра 2, то таких чисел $3 \cdot 6!$ (в 0-м разряде — одна из трёх цифр 0, 4, 6, в остальных разрядах — любые из шести оставшихся). Столько же чисел, имеющих в 7-м разряде цифры 4 и 6. Поэтому сумма десятков миллионов равна

$$(1 + 3 + 5 + 7) \cdot 4 \cdot 6! + (2 + 4 + 6) \cdot 3 \cdot 6! = (16 \cdot 4 + 12 \cdot 3) \cdot 6! = 100 \cdot 6!.$$

Таким образом, среднее арифметическое всех составленных чисел

$$\frac{72 \cdot 6! + 88 \cdot 6! \cdot (10 + 10^2 + 10^3 + 10^4 + 10^5 + 10^6) + 100 \cdot 6! \cdot 10^7}{25 \cdot 6!} =$$

$$= \frac{72 + 880 + 88 \cdot 100 \cdot 11111 + 10^9}{25} = \frac{2 + 950 + 88 \cdot 11111 \cdot 100 + 10^9}{25}.$$

Все слагаемые в числителе, кроме первого, делятся на 25, поэтому дробная часть среднего арифметического равна 2/25.

**Ответ:** $\frac{2}{25} = 0{,}08$.

***Критерии оценивания***

- Найдено количество всех чисел: +5 баллов.
- Верно подсчитана сумма всех чисел: +10 баллов.
- Из-за арифметической ошибки получен неправильный ответ при условии, что рассуждения верные — не более 20 баллов.

# Математика. 10–11 классы

## *Задача VI.1.3.1. (15 баллов)*

*Темы: текстовая задача, система уравнений.*

Для создания анимационных роликов используется два компьютера. Рендеринг ролика можно запустить как на одном из компьютеров, так и одновременно на обоих (параллельно). При параллельной работе производительность каждого компьютера уменьшается на 30% по сравнению с одиночным режимом.

Рендеринг ролика начали параллельно, но через 8 часов один из компьютеров забрали для других задач. До завершения рендеринга другой компьютер проработал ещё 12 часов. Если бы весь процесс шёл параллельно до конца, то было бы потрачено 16 часов. За какое время рендеринг ролика проходил бы на каждом из компьютеров в одиночном режиме?

***Решение***

Пусть в одиночном режиме первый компьютер обрабатывает ролик за $x$ ч, а второй (тот, что забрали для других задач) — за $y$ ч. Тогда их производительности в одиночном режиме равны $1/x$ и $1/y$ соответственно. В параллельном режиме их производительности падают на 30%, поэтому они равны $0{,}7/x$ и $0{,}7/y$ соответственно.

Исходя из условия имеем систему уравнений

$$\begin{cases} \left(\frac{0{,}7}{x}+\frac{0{,}7}{y}\right)\cdot 8+\frac{1}{x}\cdot 12=1, \\ \left(\frac{0{,}7}{x}+\frac{0{,}7}{y}\right)\cdot 16=1. \end{cases}$$

Выражая из второго уравнения совместную производительность (выражение в скобках) и подставляя в первое уравнение, находим

$$\frac{1}{16}\cdot 8+\frac{1}{x}\cdot 12=1,$$

откуда $x=24$.

Подставляя во второе уравнение найденное значение $x$, получаем $y=21$.

**Ответ:** 21 и 24 часа.

***Критерии оценивания***

- По условию задачи составлена система уравнений — 3 балла.

- Найдено время только для одного из компьютеров — 7 баллов.
- Допущена несущественная арифметическая ошибка, при условии, что всё остальное верно — 13 баллов.

## *Задача VI.1.3.2. (20 баллов)*

*Темы: графы, комбинаторика.*

Сеть оросительных каналов берёт начало от полноводной реки и заканчивается большим водоёмом для стока лишней воды во время паводка. Кроме каналов в сети есть 7 шлюзов, направляющих воду в тот или иной канал, связанный с данным шлюзом. Если считать нулевым шлюзом реку, а восьмым шлюзом — водоём, то справедливо следующее утверждение: вода может течь от шлюза $i$ к шлюзу $j$, если и только если $0 < j - i \leqslant 4$. Сколько путей для воды проходит через шлюз 5?

### *Решение*

Оросительную сеть можно представить в виде ориентированного графа, имеющего один вход (река), один выход (водоем) и семь промежуточных вершин (шлюзов). Следующая таблица (матрица смежности) описывает сеть каналов: на пересечении $i$-й строки и $j$-го столбца стоит единица, если вода может течь от шлюза $i$ к шлюзу $j$.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | | | | |
| 1 | | 1 | 1 | 1 | 1 | | | |
| 2 | | | 1 | 1 | 1 | 1 | | |
| 3 | | | | 1 | 1 | 1 | 1 | |
| 4 | | | | | 1 | 1 | 1 | 1 |
| 5 | | | | | | 1 | 1 | 1 |
| 6 | | | | | | | 1 | 1 |
| 7 | | | | | | | | 1 |

Обозначим через $n_{i,j}$ количество способов пройти из вершины $i$ в вершину $j$. По столбцу 5 матрицы смежности видно, что в вершину графа 5 входят дуги из вершин 1, 2, 3, 4. Поэтому количество путей из вершины 0 в вершину 5 равно $n_{0,5} = n_{0,1}+n_{0,2}+n_{0,3}+n_{0,4}$. По строке 5 матрицы смежности видно, что из вершины 5 можно попасть в вершины 6, 7 и 8. Значит, количество путей из вершины 5 в вершину 8 равно $n_{5,8} = n_{6,8} + n_{7,8} + 1$. Таким образом, искомое количество путей

$$n_{0,5}n_{5,8} = (n_{0,1} + n_{0,2} + n_{0,3} + n_{0,4})(n_{6,8} + n_{7,8} + 1).$$

По столбцу 1 находим: $n_{0,1} = 1$.

По столбцу 2 находим: $n_{0,2} = 1 + n_{0,1} = 1 + 1 = 2$.

По столбцу 3 находим: $n_{0,3} = 1 + n_{0,1} + n_{0,2} = 1 + 1 + 2 = 4$.

По столбцу 4 находим: $n_{0,4} = 1 + n_{0,1} + n_{0,2} + n_{0,3} = 1 + 1 + 2 + 4 = 8$.

Из строки 7 следует, что $n_{7,8} = 1$.

Из строки 6 следует, что $n_{6,8} = n_{7,8} + 1 = 1 + 1 = 2$.

Таким образом, общее количество путей, проходящих через 5-ю вершину, равно

$$(1+2+4+8)\cdot(2+1+1)=60.$$

**Ответ:** 60.

***Критерии оценивания***

- Верно подсчитано количество путей из реки в 5-й шлюз — 9 баллов.
- Верно подсчитано количество путей из шлюза 5 в водоём — 9 баллов.

## *Задача VI.1.3.3. (20 баллов)*

*Темы: теория вероятностей, теорема умножения.*

Два игрока поочередно бросают игральный кубик. Тот, у которого первым выпадает число очков кратное трем, делает два независимых выстрела по мишени. Вероятность попасть в мишень при одном выстреле для игрока, бросившего кубик первым, равна $0,4$, а для второго игрока — $0,8$. Выигрывает тот, кто попадет в мишень хотя бы один раз. Если стреляющий промахивается оба раза, то игра заканчивается без победителей. У какого из игроков шанс выиграть выше (вероятность победы больше)?

***Решение***

Пусть событие $A_1$ — выигрыш первого игрока. Имеем

$$P(A_1)=\left(\frac{1}{3}+\frac{2}{3}\cdot\frac{2}{3}\cdot\frac{1}{3}+\ldots\right)\cdot(1-0.6^2)=\frac{3}{5}\cdot(1-0,6^2)=0,384.$$

Пусть событие $A_2$ — выигрыш второго игрока. Тогда

$$P(A_2)=\left(1-\frac{3}{5}\right)\cdot(1-0,2^2)=\frac{2}{5}\cdot(1-0,2^2)=0,384.$$

**Ответ:** шансы равны.

***Критерии оценивания***

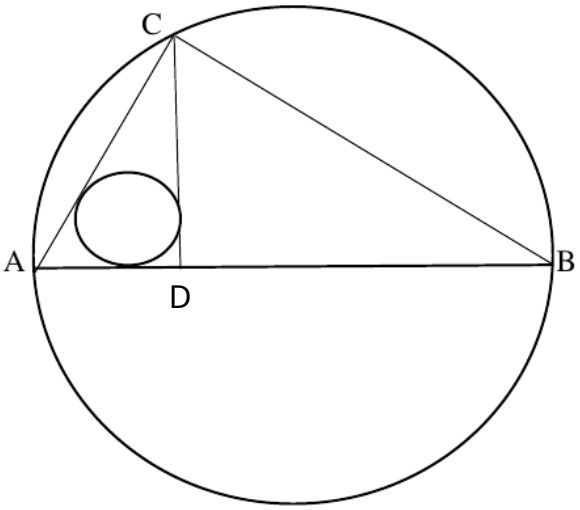
- Верно подсчитан шанс выполнять стрельбу по мишени для каждого игрока — по 4 балла.
- Верно подсчитана вероятность хотя бы одного попадания для каждого игрока — по 4 балла.
- Для вычисления итоговых вероятностей использована вероятность произведения — 4 балла (все баллы суммируются).
- Получен неверный ответ из-за арифметической ошибки — 18 баллов.

## *Задача VI.1.3.4. (20 баллов)*

*Темы: планиметрия, геометрическая вероятность.*

В окружность радиуса $R$ вписан треугольник, основанием которого является диаметр окружности, а одна из его боковых сторон равна 20. Высота, проведенная к основанию, равна 16 и разбивает треугольник на 2 прямоугольных треугольника, в меньший из которых вписана окружность радиуса $r$. В круг радиуса $R$ случайным образом бросают точку. Найдите вероятность того, что она попадет в круг меньшего радиуса $r$.

***Решение***




Пусть $AB$ — диметр окружности радиуса $R$, $CD$ — высота, опущенная из угла $C$ на сторону $AB$.

1) Вписанный угол $ACB$ опирается на диаметр окружности, следовательно, он является прямым. Угол $CDB$ является прямым по построению. Таким образом, треугольники $BCD$ и $ABC$ — прямоугольные и имеют общий угол $ABC$, следовательно, они являются подобными (по первому признаку). Тогда

$$\frac{CD}{AC} = \frac{BC}{AB}. \tag{VI.1.1}$$

По условию один из катетов треугольника $ABC$ равен 20. Длину второго катета обозначим $x$. Тогда возможны 2 варианта: $AC = 20,\ BC = x$ или $AC = x,\ BC = 20$.

В первом случае соотношение (VI.1.1) принимает вид

$$\frac{16}{20} = \frac{x}{2R} \Longrightarrow\ x = \frac{8R}{5}.$$

Во втором случае соотношение (VI.1.1) принимает вид

$$\frac{16}{x} = \frac{20}{2R} \Longrightarrow\ x = \frac{8R}{5}.$$

Согласно теореме Пифагора, сумма квадратов катетов равна квадрату гипотенузы, поэтому в обоих случаях будем иметь

$$20^2 + x^2 = (2R)^2 \Longrightarrow 20^2 + \left(\frac{8R}{5}\right)^2 = 4R^2 \Longrightarrow R = \frac{50}{3} \Longrightarrow x = \frac{80}{3}.$$

Следовательно, меньший катет равен 20, поэтому $AC = 20$ (согласно приведенному рисунку и введенным обозначениям).

2) Рассмотрим прямоугольный треугольник $ACD$, в который вписана окружность радиуса $r$. В нем $AC = 20, \quad CD = 16 \Longrightarrow \ AD = \sqrt{20^2 - 16^2} = 12$.

Радиус вписанной окружности можно найти различными способами, например, по формуле $r = \dfrac{a + b - c}{2} = \dfrac{12 + 16 - 20}{2} = 4$.

3) Вероятность того, что точка, случайным образом брошенная в круг радиуса $R$, попадет в круг радиуса $r$, будет равна отношению площадей кругов:

$$P(A) = \frac{\pi r^2}{\pi R^2} = \frac{16 \cdot 9}{2500} = \frac{36}{625} = 0,0576$$

**Ответ:** $\dfrac{36}{625} = 0,0576$.

***Критерии оценивания***

- Указано, что вписанный угол, опирающийся на диаметр, является прямым — 4 балла.
- Получено соотношение типа (VI.1.1) — 4 балла.
- Доказано, что меньший катет равен 20 — 4 балла.
- Найден радиус меньшего круга $r$ — 4 балла.

## *Задача VI.1.3.5. (25 баллов)*

*Темы: теория чисел, основная теорема арифметики, остатки.*

Начиная с числа $x$ берётся несколько последовательных чисел, дающих остаток 2 при делении на 3 (каждое следующее число отличается от предыдущего на 3). При каком наименьшем $x$ сумма этих чисел может равняться 2023?

***Решение***

Предположим, что берётся $k$ чисел, которые в сумме дают 2023:

$$2023 = x + (x + 3) + (x + 6) + \ldots + (x + 3(k - 1)).$$

Поскольку $2023 \equiv 1 \pmod 3$, то $k \geqslant 2$.

Пусть $x = 3q + 2$. По формуле для суммы $k$ членов арифметической прогрессии имеем

$$2023 = \frac{x + x + 3(k - 1)}{2} k = \left(3q + \frac{3}{2}k + \frac{1}{2}\right) k.$$

Рассмотрим случай, когда $k$ чётное, то есть $k = 2m$. Тогда

$$2023 = \left(3q + \frac{3}{2} \cdot 2m + \frac{1}{2}\right) \cdot 2m = (3q + 3m + 1/2) \cdot 2m = (6(q + m) + 1)m.$$

Так как $2023 = 7 \cdot 17^2$, $7 \equiv 1 \pmod 6$, $17 \equiv 5 \pmod 6$, то всевозможные делители числа 2023, которые при делении на 6 дают остаток 1 — это числа 1, 7, $17^2 = 289$ и 2023.

Если $6(q+m)+1=1$, $m=2023$, то $q=-2023$.

Если $6(q+m)+1=7$, $m=17^2=289$, то $q=-288$.

Если $6(q+m)+1=289$, $m=7$, то $q=41$.

Если $6(q+m)+1=2023$, $m=1$, то $q=336$.

Рассмотрим случай, когда $k$ нечётное, то есть $k=2m+1$. Тогда

$$2023=(3q+\frac{3}{2}(2m+1)+\frac{1}{2})(2m+1)=(3(q+m)+2)(2m+1)$$

Так как $7\equiv 1 \pmod 3$, $17\equiv 2 \pmod 3$, то всевозможные делители числа 2023, которые при делении на 3 дают остаток 2 — это числа 17 и $7\cdot 17=119$.

Если $3(q+m)+2=17$, $2m+1=119$, то $q=-54$.

Если $3(q+m)+2=119$, $2m+1=17$, то $q=31$.

Среди всех возможных значений $q$ наименьшим является $-2023$. Следовательно, наименьшее возможное значение $x=3\cdot(-2023)+2=-6067$.

**Ответ:** $-6067$.

***Критерии оценивания***

- Применена формула арифметической прогрессии для суммы данных в условии чисел — 5 баллов.
- Установлена необходимость рассмотрения четности-нечетности и найдены некоторые варианты значений $x$ — 15 баллов.
- Рассмотрены все возможные варианты для значений $x$, но при подсчете допущена арифметическая ошибка — 23 балла.