Искусственный интеллект

2022/23 учебный год

Заключительный этап

Предметный тур

Информатика. 8–11 класс

Задача VI.1.1.1. Покупка серверов (10 баллов)

Условие

Для обучения искусственного интеллекта (ИИ) требуются вычислительные мощности. Компания, в которой вы работаете, поручила вам купить K серверов, на которых будут обучаться нейросети.

На выбор есть N различных серверов, про каждый известна его цена C_i и его мощность B_i. Выберите K наилучших серверов по соотношению мощность/цена. Можно купить не более одного сервера каждого вида.

Формат входных данных

В первой строке даны 2 числа — N и K, про которые известно, что $(1 \leqslant N \leqslant \leqslant 100000)$, $(1 \leqslant K \leqslant N)$.

В следующей строке даны N чисел B_i — мощности серверов, про которые известно, что $(1 \leqslant B_i \leqslant 10000)$.

В третьей строке даны N чисел C_i — цены серверов, про которые известно, что $(1 \leqslant C_i \leqslant 10000)$.

Формат выходных данных

Выведите **по возрастанию** K различных чисел из диапазона $[1...N]$ — номера купленных серверов. Если оптимальных ответов несколько, вывести любой.

Примеры

Пример №1

Стандартный ввод
5 4 1 2 3 3 2 2 4 6 9 1

Стандартный вывод
1 2 3 5

Решение

Отсортируем компьютеры по соотношению мощность/цена и выберем k наилучших.

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
#include <bits/stdc++.h>

using namespace std;

bool cmp(vector<int>& l, vector<int>& r) {
    int left = l[1] * r[2];
    int right = l[2] * r[1];
    return left < right;
}

int main() {
    int n, k;
    cin >> n >> k;
    vector<int> powers(n);
    for (int i = 0; i < n; ++i)
        cin >> powers[i];
    vector<int> costs(n);
    for (int i = 0; i < n; ++i)
        cin >> costs[i];

    vector<vector<int>> servers(n);
    for (int i = 0; i < n; ++i) {
        servers[i].resize(3);
        servers[i][0] = i + 1;
        servers[i][1] = powers[i];
        servers[i][2] = costs[i];
    }
    sort(servers.begin(), servers.end(), cmp);
    reverse(servers.begin(), servers.end());

    vector<int> ans(k);
    for (int i = 0; i < k; ++i)
        ans[i] = servers[i][0];
    sort(ans.begin(), ans.end());
    for (int i = 0; i < k; ++i)
        cout << ans[i] << " ";
    cout << '\n';
}
```

Задача VI.1.1.2. Подбор картинки (15 баллов)

Условие

Ученые загадочной страны Берляндии разработали новую модель ИИ, предсказывающую насколько картинка понравится пользователю. Согласно разработанной теории, картинка — это набор пикселей на плоскости, а удовольствие от ее просмотра прямо пропорционально количеству одноцветных тупоугольных треугольников.

Помогите ученым в написании этой модели. Ваша задача состоит в том, чтобы посчитать, сколько одноцветных тупоугольных треугольников есть в изображении.

В данной задаче мы считаем, что пиксели настолько малы, что они не воспринимаются как точки на плоскости. Считаем координатами точки-пикселя номер строки и столбца, в которых он находится.

Треугольник называется одноцветным, если во всех трех его вершинах стоят одинаковые цвета.

Треугольник называется тупоугольным, если один из его углов строго больше 90°.

Треугольник называется вырожденным, если все три его вершины расположены на одной прямой.

Вырожденные треугольники считать не нужно.

Формат входных данных

В первой строке даны 2 числа — N и M, про которые известно, что $(1 \leqslant N \leqslant 10)$, $(1 \leqslant M \leqslant 10)$.

В следующих N строках дано по M чисел через пробел — цвета соответствующих пикселей на картинке.

Каждый пиксель — это число X, про которое известно, что $(0 \leqslant X \leqslant 256^3)$.

Формат выходных данных

Выведите одно число — количество тупоугольных треугольников на картинке. Если их нет — вывести 0.

Примеры

Пример №1

Стандартный ввод
2 3 2 2 1 2 3 2
Стандартный вывод
1

Пояснения к примеру

На картинке всего один пиксель цвета 1 и один пиксель цвета 3, треугольников они не образуют.

Пиксели цвета 2 имеют координаты:

(0, 0) — левый верхний (1).

(1, 0) — левый нижний (2).

(0, 1) — средний столбцей верхний (3).

(1, 2) — правый нижний (4).

Направление осей: ось X направлена вниз, ось Y направлена вправо. Если записать их в двумерный массив, то пиксель с координатами (x, y) можно получить как `matrix[x][y]`. Всего есть 4 треугольника — (1, 2, 3), (1, 2, 4), (2, 3, 4) и (1, 3, 4). Первые 3 являются прямоугольными, последний — тупоугольный.

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
#include <bits/stdc++.h>

using namespace std;

int len(int x1, int y1, int x2, int y2) {
    int dx = x1 - x2;
    int dy = y1 - y2;
    return dx * dx + dy * dy;
}

bool check_obtuce_triangle(int x1, int y1, int x2, int y2, int x3, int y3) {
    int a = len(x1, y1, x2, y2);
    int b = len(x1, y1, x3, y3);
    int c = len(x2, y2, x3, y3);
    vector<int> tmp = {a, b, c};
    sort(tmp.begin(), tmp.end());
    a = tmp[0]; b = tmp[1]; c = tmp[2];
    if ((x2 - x1) * (y3 - y1) - (x3 - x1) * (y2 - y1) == 0)
        return false; // вырожденный треугольник
    if (a + b < c)
        return true;
    return false;
}

int main() {
    int n, m;
    cin >> n >> m;
    vector<vector<int>> data(n, vector<int>(m));
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < m; ++j)
            cin >> data[i][j];
    int ans = 0;
    for (int x1 = 0; x1 < n; ++x1)
        for (int y1 = 0; y1 < m; ++y1)
            for (int x2 = 0; x2 < n; ++x2)
                for (int y2 = 0; y2 < m; ++y2)
                    for (int x3 = 0; x3 < n; ++x3)
                        for (int y3 = 0; y3 < m; ++y3)
                            if (data[x1][y1] == data[x2][y2] && data[x1][y1] ==
                            ↪  data[x3][y3])
                                if (check_obtuce_triangle(x1, y1, x2, y2, x3, y3))
                                    ++ans;
    assert(ans % 6 == 0);
    cout << ans / 6 << '\n';
    return 0;
}
```

Задача VI.1.1.3. Голосовой помощник (20 баллов)

Условие

В современных городах очень легко заблудиться без навигатора. Развязки, ремонты, стройки — все это надо учитывать при выборе маршрута. Вы работаете в компании, которая поддерживает текущее состояние дорог и тротуаров. На сервере хранится карта города, разбитая на маленькие прямоугольники. Про каждый из них известно, есть через него проход или нет.

Вам поручили разработать логику голосового помощника, который будет строить маршрут по данным карты и возвращать текст для озвучки пользователю.

Формат входных данных

В первой строке даны 2 числа — N и M, про которые известно, что $(1 \leqslant N \leqslant 100)$, $(1 \leqslant M \leqslant 100)$.

Затем вводится N строк, в каждой из которых M символов, описывающих возможность прохода через ту или иную клетку. Каждый символ является либо точкой («.»), обозначающей наличие прохода, либо решеткой («#»), обозначающий преграду. Отдельными символами обозначается стартовая клетка («S») и финишная («F»). Необходимо проложить и озвучить кратчайший маршрут из точки S в точку F. длиной маршрута считается число шагов. Исходим из того, что повороты происходят моментально. Гарантируется, что на карте есть ровно один символ S и ровно один символ F.

Формат выходных данных

Выведите не более 1000 действий следующего вида:

«MOVE RIGHT» — поворот направо,

«MOVE LEFT» — поворот налево,

«GO» — движение прямо,

«FINISH» — сообщение пользователю о том, что он добрался до точки назначения.

Изначально вы смотрите вниз. Если первым шагом сделать GO, пользователь опустится на строку ниже в лабиринте.

Поле по периметру окружено решетками, т. е. случайно выйти за пределы поля не получится.

Если во время движения вы дойдете до стены и попытаетесь продолжить движение в стену, то получите вердикт «Wrong answer».

Правильным считается любой маршрут из начальной точки в конечную, использующий минимально возможное количество команд GO. Всего в маршруте должно быть не более 1000 команд.

Примеры

Пример №1

Стандартный ввод
5 6 ###### #S...# ##.#.# ###F.# ######

Стандартный вывод
MOVE LEFT GO GO GO MOVE RIGHT GO GO MOVE RIGHT GO FINISH

Решение

Воспользуемся поиском в ширину.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n, m = map(int, input().split())

pole = []
for i in range(n):
    pole.append(input().strip())

start_x = -1
start_y = -1
for i in range(n):
    for j in range(m):
        if pole[i][j] == "S":
            start_x = i
            start_y = j

finish_x = -1
finish_y = -1
for i in range(n):
    for j in range(m):
        if pole[i][j] == "F":
            finish_x = i
            finish_y = j

dist = []
```

```
from1 = []
for i in range(n):
    ln = []
    from_line = []
    for j in range(m):
        ln.append(100000)
        from_line.append([-1, -1])
    dist.append(ln)
    from1.append(from_line)

q = [[start_x, start_y]]
dist[start_x][start_y] = 0
add = [[0, 1], [0, -1], [1, 0], [-1, 0]]
while len(q):
    now_x = q[0][0]
    now_y = q[0][1]
    q.pop(0)
    for add_x_y in add:
        new_x = now_x + add_x_y[0]
        new_y = now_y + add_x_y[1]
        if pole[new_x][new_y] != "#":
            if dist[new_x][new_y] > dist[now_x][now_y] + 1:
                q.append([new_x, new_y])
                dist[new_x][new_y] = dist[now_x][now_y] + 1
                from1[new_x][new_y] = [now_x, now_y]
path = [[finish_x, finish_y]]
while path[-1][0] != start_x or path[-1][1] != start_y:
    prev = from1[path[-1][0]][path[-1][1]]
    path.append(prev)
path.reverse()

napr = 0
add = [[1, 0], [0, -1], [-1, 0], [0, 1]]
ans = []
for i in range(1, len(path)):
    nw = path[i]
    pr = path[i - 1]
    dx = nw[0] - pr[0]
    dy = nw[1] - pr[1]

    mv = [dx, dy]
    if mv == add[(napr + 1) % 4]:
        ans.append("MOVE RIGHT")
        napr = (napr + 1) % 4
    else:
        while mv != add[napr]:
            ans.append("MOVE LEFT")
            napr = (napr + 3) % 4
    ans.append("GO")
ans.append("FINISH")

print('\n'.join(ans))
```

Задача VI.1.1.4. Работа с вероятностью (25 баллов)

Условие

В моделях машинного обучения для характеристик различных сущностей применяют теорию вероятности и многомерные векторы.

Например {1, 2} — двумерный вектор, у которого число 1 — первая коорданата, {4, 3, 2, 5, 4} — пятимерный вектор, первая координата которо равняется 4.

Вы работете с некой сложной моделью ИИ, в которой координаты векторов — случайные целые числа из интервала $[0, K]$. K — некое целое число, которое заранее известно. Разработчики алгоритма попросили написать программу, которая вычислит важный коэффициент, необходимый для работы нейросети.

Его значение определяется как вероятность события, при котором среднее арифметическое первых координат N векторов будет равно дроби вида M/N, где M — некое число, значение которого вы знаете. Выведите ответ ровно с 6 знаками после запятой.

Напишите программу, которая рассчитает его значение.

Формат входных данных

В единственной строке даны 3 числа через пробел — N, M, K, про которые известно, что $(2 \leqslant N \leqslant 100), (0 \leqslant M \leqslant 1000), (1 \leqslant k \leqslant 100)$.

Формат выходных данных

Выведите единственное число — значение коэффициента ровно с 6 знаками после запятой.

Примеры

Пример №1

Стандартный ввод
`2 2 2`

Стандартный вывод
`0.333333`

Пример №2

Стандартный ввод
`3 5 4`

Стандартный вывод
`0.144000`

Пояснения к примеру

Пример №1

У первого вектора координата — случайное число из набора $\{0, 1, 2\}$. У второго также. Итого есть 9 случаев — $\{0, 0\}$, $\{0, 1\}$, $\{0, 2\}$, $\{1, 0\}$, $\{1, 1\}$, $\{1, 2\}$, $\{2, 0\}$, $\{2, 1\}$, $\{2, 2\}$. Вероятность каждого случая — $\frac{1}{9}$. Всего у нас 3 случая, когда сумма равняется двум — $\{0, 2\}$, $\{1, 1\}$, $\{2, 0\}$. Поэтому искомая вероятность равняется $\frac{1}{9} + \frac{1}{9} + \frac{1}{9} = \frac{1}{3}$.

Пример №2

Координата — случайное число из набора $\{0, 1, 2, 3, 4\}$. Всего есть 125 вариантов — $\{0, 0, 0\}$, $\{0, 0, 1\}$, ..., $\{4, 4, 3\}$, $\{4, 4, 4\}$. Вероятность каждого случая — $\frac{1}{125}$. Из них 18 имеют сумму 5 — $\{0, 1, 4\}$, $\{0, 2, 3\}$, $\{0, 3, 2\}$, $\{0, 4, 1\}$, $\{1, 0, 4\}$, $\{1, 1, 3\}$, $\{1, 2, 2\}$, $\{1, 3, 1\}$, $\{1, 4, 0\}$, $\{2, 0, 3\}$, $\{2, 1, 2\}$, $\{2, 2, 1\}$, $\{2, 3, 0\}$, $\{3, 0, 2\}$, $\{3, 1, 1\}$, $\{3, 2, 0\}$, $\{4, 0, 1\}$, $\{4, 1, 0\}$. Поэтому искомая вероятность равняется $\frac{18}{125}$.

Решение

Применим метод динамического программирования — посчитаем вероятность выпадения суммы j из первых i векторов.

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
#include <bits/stdc++.h>

using namespace std;

using ld = long double;

int main() {
    size_t n, m, k; cin >> n >> m >> k;
    ld factor = (ld)1.0 / (ld)(k + 1);

    vector<vector<ld>> dp(n + 1, vector<ld>(m + 1));
    dp[0][0] = 1;

    for (size_t i = 1; i <= n; ++i) {
        for (size_t j = 0; j <= m; ++j) {

            for (int x = 0; x <= k && x <= j; ++x) {
                dp[i][j] += dp[i - 1][j - x] * factor;
            }
        }
    }

    cout << fixed << showpoint << setprecision(6) << dp[n][m] << endl;
}
```

Задача VI.1.1.5. Подбор музыки (30 баллов)

Условие

Одной из сфер применения ИИ является подбор музыки в музыкальных сервисах.

В данной задаче известна мелодия и музыкальные фрагменты, которые нравятся пользователю. Мелодия представляет собой числовую последовательность. Фрагменты — также числовые последовательности. Если фрагмент встречается единожды, он приносит 1^2 единицу удовольствия, дважды — 2^2 удовольствия. .... N раз встречающийся фрагмент приносит N^2 единиц удовольствия. Посчитайте, сколько единиц удовольствия принесет данная песня конкретному пользователю.

Формат входных данных

В первой строке дано 2 числа N, M – размер последовательности и число музыкальных фрагментов, $(1 \leqslant N \leqslant 3 \cdot 10^5), (1 \leqslant M \leqslant 10^5)$.

В следующей строке задано N чисел B_i через пробел — частоты проигрываемой мелодии $(1 \leqslant B_i \leqslant 50)$. Частоты усреднены, чтобы повысить шанс совпадения мелодии и вкусов пользователя.

Следующие M строк описывают фрагменты, которые нравятся пользователю. В начале каждой строки записано число $S_i, (1 \leqslant S_i \leqslant 10^5)$ — длина очередного фрагмента. Затем в этой же строке идут S_i чисел — частоты фрагмента T_{ij}, $(1 \leqslant T_{ij} \leqslant \leqslant 50)$. Известно, что в базе на одного пользователя отводится не очень много места, из-за чего суммарная длина всех фрагментов также не превышает $3 \cdot 10^5$ символов. Формально говоря $S_1 + S_2 + ... + S_N \leqslant 3 \cdot 10^5$.

Один и тот же фрагмент может встретиться несколько раз.

Формат выходных данных

Выведите одно число — сколько единиц удовольствия принесет данная песня конкретному пользователю.

Примеры

Пример №1

Стандартный ввод
10 5 2 9 24 28 24 21 7 11 24 21 3 27 1 39 3 37 4 47 3 20 5 18 4 24 21 7 11 4 2 9 24 28

Стандартный вывод
2

Пояснения к примеру

Начало мелодии (первые 4 частоты) [2, 9, 24, 28] совпадают с фрагментом 5.

Также отрезок мелодии с 5-го по 8-й ноту [24, 21, 7, 11] совпадает с фрагментом 4.

Можно убедиться, что других полных совпадений фрагментов с мелодией нет.

Решение

Давайте разделим строки на легкие и тяжелые. Легкими будем считать те, длина которых меньше $\sqrt{M}$.

Длинными — те, длина которых более $\sqrt{M}$. Первым шагом разобьем строки на 2 массива, и решим для каждых задачу разными способами.

Решим задачу для длинных строк. Сколько их может быть? Если длина хотя бы $\sqrt{M}$, а суммарная длина — M, то их количество меньше или равно $\sqrt{M}$. Воспользуемся префикс-функцией для поиска всех вхождений строки в текст для каждой строки — $O(\text{длина строки} + N)$. Итого решение длинных строк — $O(M + \sqrt{M} \cdot N)$.

Для решения коротких строк воспользуемся хешированием. Посчитаем хеши всех префиксов. Используя их, выпишем хеши всех подстрок длины 1, 2, ..., $\sqrt{M}$ в словарь так, чтобы ключом было значением хеша, значенем ключа — сколько раз ключ встречался в строке. Затем считаем хеш каждого фрагмента и из словаря, соответствующего длине фрагмента, узнаем число вхождений в мелодию. Итого решение за $O(N \cdot \sqrt{M} + M)$, где первое слагаемое — подсчет хешей отрезков, M — суммарное время вычисления хешей фрагментов. Сложив ответы для длинных и коротких строк, получим ответ на задачу. Итоговая сложность решения — $O(N \cdot \sqrt{M} + M)$.

Такой подход к решению задач — отдельно разбираться с большими, отдельно с маленьким называется корневой декомпозицией.

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
#include <bits/stdc++.h>

using namespace std;

vector<int> prefix_function(const vector<int>& s) {
    int n = (int)s.size();
    vector<int> pi(n);
    for (int i = 1; i < n; i++) {
        int j = pi[i-1];
        while (j > 0 && s[i] != s[j])
            j = pi[j-1];
        if (s[i] == s[j])
            j++;
        pi[i] = j;
    }
    return move(pi);
}

int countIn(int sz, const vector<int>& s) {
    vector<int> ans = prefix_function(s);
```

```
    int cnt = 0;
    for (auto x : ans)
        if (x == sz)
            ++cnt;
    return cnt * cnt;
}

long long long_solve(vector<vector<int>>&& long_parts, const vector<int>& text) {
    long long ans = 0;
    for (const auto& line : long_parts) {
        vector<int> s;
        s.reserve(line.size() + 1 + text.size());
        for (auto x : line)
            s.push_back(x);
        s.push_back(-1);
        for (auto x : text)
            s.push_back(x);
        ans += (long long)countIn(line.size(), s);
    }
    return ans;
}

const int MAXN = 300000;
const long long P = 53;
const long long MOD = 1e9 + 7;
long long p[MAXN], pref_hash[MAXN];

void precalc() {
    p[0] = 1;
    for (int i = 1; i < MAXN; ++i)
        p[i] = p[i - 1] * P % MOD;
}

void calc_text_hash(const vector<int>& text) {
    pref_hash[0] = 0;
    for (int i = 0; i < text.size(); ++i) {
        pref_hash[i + 1] = (pref_hash[i] * P + text[i]) % MOD;
    }
}

long long get_hash(int l, int r) {
    return (pref_hash[r] + (MOD - pref_hash[l]) * p[r - l]) % MOD;
}

void recount(const vector<int>& text, unordered_map<int, int>& hashs, int len) {
    hashs.clear();
    for (int i = 0; i + len <= text.size(); ++i)
        ++hashs[get_hash(i, i + len)];
}

int count_hash(const vector<int>& line) {
    long long hash = 0;
    for (auto el : line)
            hash = (hash * P + el) % MOD;
    return (int)hash;
}

long long short_solve(vector<vector<int>>&& short_parts, const vector<int>& text)
↪ {
    sort(short_parts.begin(), short_parts.end(),[] (const vector<int>& lhs, const
    ↪ vector<int>& rhs) {return lhs.size() < rhs.size();});
```

```
80      long long ans = 0;
81      int len_counted = 0;
82      unordered_map<int, int> cnt_by_hash;
83      for (const auto& line : short_parts) {
84              if (line.size() != len_counted) {
85                      recount(text, cnt_by_hash, line.size());
86                      len_counted = line.size();
87              }
88              long long cnt_in = (long long)cnt_by_hash[count_hash(line)];
89              ans += cnt_in * cnt_in;
90      }
91      return ans;
92  }
93
94  int main() {
95      precalc();
96      cin.tie(nullptr);
97      cout.tie(nullptr);
98      ios_base::sync_with_stdio(false);
99      int n, m;
100     cin >> n >> m;
101     vector<int> text(n);
102     for (int i = 0; i < n; ++i)
103             cin >> text[i];
104     calc_text_hash(text);
105     vector<vector<int>> long_parts, short_parts;
106     const int MID = 300;
107     for (int i = 0; i < m; ++i) {
108             int sz_line;
109             cin >> sz_line;
110             (sz_line > MID) ? long_parts.push_back({}) :
                ↪  short_parts.push_back({});
111             vector<int>& new_line = (sz_line > MID) ? long_parts.back() :
                ↪  short_parts.back();
112             new_line.resize(sz_line);
113             for (int i = 0; i < sz_line; ++i)
114                     cin >> new_line[i];
115     }
116     long long ans = long_solve(move(long_parts), text) +
        ↪  short_solve(move(short_parts), text);
117     cout << ans << '\n';
118     return 0;
119 }
```

Тестовые наборы для задач представленны по ссылке — https://disk.yandex.ru/d/-LVERknoaUJZCA.

Математика. 8–9 классы

Задача VI.1.2.1. (15 баллов)

Условие

Момент времени будем называть *красивым*, если после выписывания даты в формате «год.месяц.день» и приписывания далее точного времени в 24-часовом формате получается симметричная последовательность цифр, то есть такая, которая слева направо читается так же, как и справа налево. Например, 2020.01.10.02:02. Определите последний в этом тысячелетии красивый момент.

Решение

Так как момент времени должен быть в этом тысячелетии, то первая (а значит, и последняя) цифра равна 2. Отметим, что всего цифр в записи тогда 12, из которых две уже установлены: $2***.**.**.** : *2$. Будем называть искомый в задаче момент *моментом X*.

Предпоследняя цифра является разрядом десятков в минутах, то есть не может быть больше 5. Тогда симметричная ей цифра в разряде сотен лет момента X не может быть более 5 (а 5 может). Максимальная цифра в следующем разряде — это 9, значит число на первых трёх позициях записи момента не может быть более 259, что соответствует последним трём цифрам 9 : 52. Предшестующая им четвёртая с конца цифра представляет разряд десятков в часе момента X. То есть эта цифра может быть 0, 1 или 2. При этом для следующей цифры 9 (в третьем с конца разряде) текущая цифра не может быть 2 (так как количество часов в такой записи не может быть более 23). Значит, максимальная цифра на четвёртой позиции (при условии, что на предыдущих 259) это 1.

Пятая и шестая цифры в записи — месяц момента X. Не может быть более 12. Тогда самое позднее начало из перых шести цифр — это 2591.12. Оно полностью и однозначно определяет весь момент, так как цифры с 7-ой по 12-ю — это зеркальное отражение уже найденных цифр.

Ответ: 2591.12.21.19:52.

Примечание: в случае трактовки тысячелетия от 2001 до 3001 года решение аналогичное и ответ 3000.12.21.00:03.

Критерии оценивания

- Только ответ — 5 баллов.
- Ошибка в рассуждении про один из разрядов (не учитывая симметричный разряд) — не более 10 баллов.
- Ошибки в рассуждениях по поводу двух несимметричных разрядов — не более 5 баллов.

Задача VI.1.2.2. (15 баллов)

Условие

Петя, Вика, Толя, Чулпан, Потап, Шарлотта стоят в указанном порядке по кругу, считая по часовой стрелке, и смотрят в центр круга (то есть по правую руку от Пети стоит Шарлотта, а по левую — Вика). Каждый мальчик загадал число и сообщил его шёпотом обеим соседкам. Каждая девочка умножила число от своего соседа против часовой стрелки на 2, и к этому результату прибавила число от соседа по часовой стрелке. В результате у Вики получилось число 57, у Чулпан — число 53, а у Шарлотты — число 31. Какое число загадал Петя?

Решение

Обозначим число Пети за x, число Толи за y, число Потапа за z. Тогда число Вики равно $y + 2x$, оно же равно 57. Аналогично, число Чулпан равно $z + 2y = 53$, число Шарлотты равно $x + 2z = 31$. Сложим все эти три числа отдельно левые части и отдельно правые. Получим равенство $3x+3y+3z = 141$. То есть $x+y+z = 141/3 = 47$.

Пользуясь установленным равенством и равенством для числа Шарлотты, имеем $(x+2z)-(x+y+z) = 31-47$, то есть $z-y = -16$. Аналогично, пользуясь выражением для числа Чулпан, имеем: $y - x = 53 - 47 = 6$. Тогда можем выразить: $y = x + 6$, $z = y - 16 = x - 10$. Подставляя в выражение с суммой всех, имеем: $x + (x + 6) + (x - 10) = 47$, то есть $3x - 4 = 47$ и $x = 17$.

Ответ: 17.

Критерии оценивания

- Только ответ — 5 баллов.
- Ошибка, связанная с поворотом по или против часовой стрелки — не более 12 баллов (-3 балла).
- Логическая ошибка прм решении системы — не более 10 баллов.
- Две логические ошибки — не более 5 баллов.
- Арифметические ошибки, не влияющие на ход решения — -3 балла.
- Арифметические ошибки, существенно влияющие на ход решения (ведущие к другой линии решения и проч.) — не более 5 баллов.

Задача VI.1.2.3. (20 баллов)

Условие

Площадь трапеции $ABCD$ равна 164 см^2, длина её высоты равна 8 см, боковые стороны AB и CD равны 10 см и 17 см соответственно. Чему равна длина основания BC?

Решение

Опустим из точек A и B перпендикуляры на прямую CD, основания перпендикуляров, соответственно, P и Q. Тогда треугольники ABP и DCQ суть прямоугольные, в которых известны гипотенуза и катет (в обоих случаях один из катетов равен длине перпендикуляра между прямыми AD и BC, то есть 8). Тогда по теореме Пифагора имеем, что $AP = \sqrt{AB^2 - BP^2} = \sqrt{10^2 - 8^2} = 6$, аналогично $DQ = 15$.

Площадь прямоугольника $PBCQ$ равна площади всей трапеции за вычетом площадей треугольников ABP и CDQ. При этом одна сторона прямоугольника равна 8. Значит, другая сторона равна: $(164 - 8 \cdot 6/2 - 8 \cdot 15/2)/8 = (41 - 21)/2 = 10$.

Ответ: 10.

Критерии оценивания

- Только ответ — 5 баллов.
- Найдены площади прямоугольных треугольников отсекаемых высотами — +5 баллов.

Задача VI.1.2.4. (25 баллов)

Условие

Функция $P(x)$ определяется как $P(x) = x^2 - 38x - 80$.

Решите уравнение $P(x) = P\left(\dfrac{x+4}{x+1}\right)$.

Решение

Функция $P(x)$ — это квадратичная функция, график которой — это парабола «усами вверх». Тогда для значения y_0 либо не существует решений уравнения $P(x) = y_0$, либо решение одно, если y_0 соответствует вершине данной параболы (для $x = -b/2a$, то есть $38/2 = 19$ в данном случае), либо решения два и они симметричны относительно x-координаты вершины параболы (то есть относительно $-b/2a = 19$ в данной задаче). Причём второй случай принято считать частным случаем третьего случая, просто два корня равны друг другу.

Воспользуемся этим правилом. Тогда если $P(x) = P\left(\dfrac{x+4}{x+1}\right)$ для какого-то x, то получаем, что либо $x = \dfrac{x+4}{x+1}$, либо эти два значения симметричны относительно x-координаты вершины параболы, то есть относительно 19.

Первое условие превращается в квадратное уравнение $x(x+1) = x+4$ после домножения на $x+1$ (и замечания, что x не равно -1). Решения этого уравнения 2 и -2.

Второе условие (симметричности) записывается как $\left(x + \dfrac{x+4}{x+1}\right)/2 = 19$. После домножения на знаменатель также получается квадратное уравнение с корнями $18 - \sqrt{358}$ и $18 + \sqrt{358}$, что завершает доказательство.

Ответ: -2, 2, $18-\sqrt{358}$, $18+\sqrt{358}$.

Критерии оценивания

- Только ответ — 5 баллов.
- Упущены решения в ответе — -2 балла за каждый упущенный.
- Разобран только случай $x=\dfrac{x+4}{x+1}$ — $+7$ баллов.
- Неверные выводы про квадратичную функцию — -5 баллов.

Задача VI.1.2.5. (25 баллов)

Условие

Сферическим кодом будем называть несколько точек в четырёхмерном пространстве (то есть с четырьмя координатами), каждая из координат которых равна либо 0, либо 1, при условии, что все попарные расстояния между этими точками одинаковы. Какое максимальное количество точек может быть в сферическом коде?

Напоминание: расстояние между двумя точками 4-мерного пространства вычисляется по четырёхмерной теореме Пифагора: расстояние между (a,b,c,d) *и* (x,y,z,t) *равно* $\sqrt{(a-x)^2+(b-y)^2+(c-z)^2+(d-t)^2}$.

Решение

Приведём пример сферического кода с 4-мя точками: $(0,0,0,0)$; $(1,1,0,0)$; $(1,0,1,0)$; $(1,0,0,1)$. Отметим, что любые две точки имеют две совпадающие координаты и две отличающиеся, то есть расстояние между любыми двумя из них равно $\sqrt{0^2+0^2+1^2+1^2}=\sqrt{2}$, то есть это — сферический код.

Покажем, что пять и более точек в сферическом коде (для четырёх координат) быть не может. Предположим противное, то есть существуют 5 точек, образующих точки сферического кода. Тогда во-первых, все эти точки имеют одинаковую чётность суммы координат (то есть для каждой точки рассматриваем сумму её координат). Действительно: раз расстояния между каждой парой точек равны, то и квадраты расстояний равны, а это целые числа. Тогда для любых трёх точек из сферического кода имеем следующие сравнения по модулю 2 (исходя из того, что для произвольного n верно $n^2\equiv n \mod 2$): пусть рассматриваются точки (a_1,a_2,a_3,a_4), (b_1,b_2,b_3,b_4) и (c_1,c_2,c_3,c_4). Тогда имеем:

$$(a_1-b_1)+(a_2-b_2)+(a_3-b_3)+(a_4-b_4)\equiv (a_1-b_1)^2+(a_2-b_2)^2+(a_3-b_3)^2+(a_4-b_4)^2\equiv$$

$$\equiv (c_1-b_1)^2+(c_2-b_2)^2+(c_3-b_3)^2+(c_4-b_4)^2\equiv (c_1-b_1)+(c_2-b_2)+(c_3-b_3)+(c_4-b_4)$$

(все сравнения указаны по модулю 2), откуда из первого и последнего равенств в цепочке имеем:

$$a_1+a_2+a_3+a_4\equiv c_1+c_2+c_3+c_4 \mod 2.$$

Всего количество точек с четырьмя координатами 0 и 1 равно $2^4=16$. Тех, у кого сумма координат чётная, столько же, сколько тех, у кого сумма координат нечётная, а именно 8. При этом если в набор одновременно попали две точки, у

которых в каждой из координат стоят разные числа (в одном 0, а в другом 1), то расстояние между такими точками равно $\sqrt{4} = 2$, что возможно только в подобной паре «дополнительных» друг для друга точек. До любой из других точек от любой из этих двух расстояние будет не более $\sqrt{3}$, что меньше 2. То есть если в сферическом коде есть две дополнительные друг другу точки, то других точек в этом сферическом коде нет. Но мы уже знаем, что есть сферический код, в котором больше 2-х точек.

Таким образом, если в сферическом коде более 2-х точек, то в нём нет дополнительных друг другу точек. Но все пары точек с четырьмя координатами, каждая из которых 0 или 1, разбиваются на пары дополнительных, причём суммы координат обеих точек будут иметь одну чётность. Значит, среди точек сферического кода будет не более половины от общего количества точек с суммой координат такой чётности. То есть их не более $8/2 = 4$.

Ответ: 4.

Критерии оценивания

- Ответ — 5 баллов.
- Пример — 5 баллов.
- Доказано, что точки одной чётности — +5 баллов.
- Доказано, что если в наборе есть пара дополнительных друг другу точек, то других точек в наборе нет — +3 балла.

Математика. 10–11 классы

Задача VI.1.3.1. (15 баллов)

Условие

Определите количество решений уравнения $\cos^2 x + (x-1)\sin x = \sin x + 1$ на отрезке $[-\pi, \pi]$.

Решение

Перенесём все слагаемые в левую часть выражения, применим основное тригонометрическое тождество $\cos^2 x + \sin^2 x = 1$. Получим уравнение $-\sin^2 x + (x-2)\sin x = 0$, то есть $\sin x(-\sin x + x - 2) = 0$. Это равнество выполнено, когда нулю равен либо первый множитель ($\sin x$), либо второй множитель ($-\sin x + x - 2$), либо оба одновременно. Первый множитель равен нулю на интервале $[-\pi, \pi]$ в трёх точках: $-\pi, 0$ и π. В других точках нулю он не равен.

Условие равенства второго множителя нулю перепишем в виде $\sin x = x - 2$. Левая часть выражения при всех значениях x принимает значения не менее -1 (так как это стандартная синусоида), а правая часть выражения строго меньше -1 при x меньше или равных 0 (так как это возрастающая линейная функция, которая равна -2 при $x = 0$). Поэтому на интервале $(-\pi, 0]$ рассматриваемое равенство невозможно.

Далее на интервале $(0, 2]$ выражение $\sin x$ принимает положительные значения

(так как $2 < \pi$), а выражение $x - 2$ на этом интервале неположительно. Значит, на этом интервале равенство $\sin x = x - 2$ также невозможно.

На интервале $(2, \pi)$ выражение $\sin x - (x - 2)$ имеет одно решение, так как эта разность непрерывна, строго убывает на данном интервале $\sin x$ строго убывает, $-(x - 2)$ также строго убывает, а значения в концах интервала имеют разные знаки ($\sin 2 - (2 - 2) = \sin 2 > \sin \pi = 0$; $\sin \pi - (\pi - 2) = 2 - \pi < 0$).

Ответ: 4.

Критерии оценивания

- Только ответ — 5 баллов.
- В целом верно, но из-за арифметических ошибок неверный ответ — не более 12 баллов.
- Логические ошибки, либо более трёх арифметических ошибок — не более 10 баллов.

Задача VI.1.3.2. (15 баллов)

Условие

Каждый день перед сном Коля смотрит юмористические шоу с кружечкой любимого напитка: чай, кофе или какао. Для этого каждый раз он случайно и равновероятно выбирает напиток. Найдите вероятность, что за неделю он выпьет каждого любимого напитка хотя бы один раз.

Решение

Исходя из условий задачи, каждая из последовательностей из 7-ми выборов напитка равновероятна. Поэтому для нахождения искомой вероятности вычислим общее количество исходов и число подходящих исходов. Не подходят последовательности, в которых встречаются только два или только один напиток. Последовательностей, в которых встречаются только чай и кофе, как легко видеть, равно 2^7. Однако при таком подсчёте по каждой из пар окажется, что последовательности ровно с одним напитком будут посчитаны дважды. Тогда общее количество благоприятных исходов равно $3^7 - 3 \cdot 2^7 + 3$, а искомая вероятность $\dfrac{3^7 - 3 \cdot 2^7 + 3}{3^7}$.

Ответ: $\dfrac{602}{729}$.

Критерии оценивания

- Только верный ответ — 5 баллов.
- Не работающие в этой задаче, но верные в целом схемы вычисления — не более 5 баллов.

Задача VI.1.3.3. (20 баллов)

Условие

Дан равносторонний треугольник PQR. Внутри угла PRQ взята точка T. Найдите $\angle PRT$, если $\angle PTR = 20^\circ$ и $\angle QTR = 30^\circ$.

Решение

Рассмотрим поворот треугольника RTQ вокруг R на 60° (в том же направлении, поворот от луча RQ до луча RT). Пусть точка T при этом повороте перейдёт в некоторую точку S. Тогда треугольник TRS равносторонний (так как $RT = RS$ и $\angle TRS = 60^\circ$ по выбору поворота).

При указанном повороте точка Q перейдёт в точку P (так как $QR = PR$, $\angle QRP = 60^\circ$). Значит, треугольник RTQ при данном повороте переходит в треугольник RSP. Тогда верно $\angle RSP = \angle RTQ = 30^\circ$. Но тогда SP — биссектриса равностроннего треугольника RST: $\angle RSP = \angle PST$.

Из симметрии правильного треугольника RST относительно своей биссектрисы SP имеем, что $\angle SRP = \angle STP$. $\angle STP = \angle STR - \angle PTR = 60^\circ - 20^\circ = 40^\circ$. Тогда искомый в задаче угол $\angle PRT = \angle SRT - \angle SRP = 60^\circ - 40^\circ = 20^\circ$.

Ответ: 20°.

Критерии оценивания

- Только ответ — 5 баллов.
- Найдены другие углы, из которых всё ещё неясно, как подобраться к требуемому углу — +0 баллов (не оценивается).
- Есть идея поворота на 60° — +3 балла.
- Указано, что в результате такого поворота получается р/с треугольник(и) — +2 балла (суммируется с предыдущим).
- Нетривиальные продвижения, которые, тем не менее, неясно, как помогут в задаче — не более +3 балла.

Задача VI.1.3.4. (25 баллов)

Условие

Произведением двух матриц размера 2×2, записанных в виде $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ и $\begin{pmatrix} x & y \\ z & t \end{pmatrix}$, называется матрица $\begin{pmatrix} ax + bz & ay + bt \\ cx + dz & cy + dt \end{pmatrix}$. k-ой степенью квадратной матрицы M (размера $m \times m$) называется выражение $M \cdot M \cdot \ldots \cdot M$ с k множителями (умножения в такой ситуации выполняются по очереди). Найдите 20-ю степень матрицы $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$.

Решение

Обозначим $\begin{pmatrix}1 & 1\\ 1 & 0\end{pmatrix}$ как M. Докажем, что M^k имеет вид $\begin{pmatrix}F_{k+1} & F_k\\ F_k & F_{k-1}\end{pmatrix}$, где F_k – числа Фибоначчи, то есть последовательность, заданная условиями $F_1 = F_2 = 1$ и $F_{i+2} = F_i + F_{i+1}$. Доказательство проведём по индукции: база $k = 1$ уже фактически проверена. Пусть утверждение доказано для всех t от 1 до m. Тогда для $M^{m+1} = M^m \cdot E_0 = \begin{pmatrix}F_{m+1} & F_m\\ F_m & F_{m-1}\end{pmatrix} \cdot \begin{pmatrix}1 & 1\\ 1 & 0\end{pmatrix} = \begin{pmatrix}F_{m+1} + F_m & F_{m+1}\\ F_m + F_{m-1} & 0\end{pmatrix}$, что и даёт требуемое.

Остаётся вычислить числа Фибоначчи до соответствующего номера. Для матрицы M^{20} необходимыми будут элементы последовательности с номерами 21, 20 и 19. Это можно делать многими способами, например, заполняя соответствующую таблицу. Выпишем числа Фибоначчи пятёрками: $1, 1, 2, 3, 5$, $8, 13, 21, 34, 55$, $89, 144, 233, 377, 610$, $987, 1597, 2584, 4181, 6765$, $10946, \ldots$.

Таким образом, ответ в задаче: $\begin{pmatrix}10946 & 6765\\ 6765 & 4181\end{pmatrix}$.

Критерии оценивания

- Только ответ — 5 баллов.
- Задача сформулирована в терминах чисел Фибоначчи — +5 баллов.
- Найдена закономерность про матрицы и числа Фибоначчи, либо что-то аналогичное — +5 баллов.

Задача VI.1.3.5. (25 баллов)

Условие

Клетки клетчатой полосы $n \times 1$ изначально покрашены в два цвета. Разрешается выбрать любую клетку вместе с соседними с ней по сторонам, и перекрасить в противоположный цвет каждую из выбранных клеток. Сколько различных покрасок этой полосы можно получить таким образом?

Решение

Проверим, что при $n = 1$ число способов (которое обозначим за Q) равно 2 (первый либо второй цвет). При $n = 2$ число способов Q равно также 2, так как при выборе любой клетки вместе с ней окажется выбрана и вторая клетка. При $n = 3$ имеем такую ситуацию, что при выборе центральной клетки выбранными оказываются все три клетки полосы. При выборе угловой клетки выбраны оказываются две крайние клетки. Отсюда видно, что за два перекрашивания можно добиться того, чтобы перекрасить только угловую клетку (любую из двух). Но тогда и одну центральную клетку перекрасить не составит труда. Значит, каждую из клеток мы научились перекрашивать произвольно и независимо от других. Значит, итоговый ответ при $n = 3$: $Q = 8 = 2^3$.

«Переведём» задачу на язык векторов с координатами 0 и 1: будем обозначать клетку первого цвета на полосе цифрой 1, клетку второго цвета — цифрой 0. Далее каждому перекрашиванию сопоставим вектор из n нулей и единиц, соответствующий

перекрашиваемым клеткам, в котором цифра 1 означает перекрашивание соответствующей клетки, а цифра 0 — сохранение без изменений. Тогда легко видеть, что итог перекрашивания обозначается вектором длины n (всё с тем же соответствием, что 1 обозначает первый цвет, а 0 — второй), равным сумме исходного вектора цветов и вектора перекрашивания, взятому по модулю 2 (по каждой координате независимо).

Для i-ой клетки полосы обозначим соответствующий ей вектор перекрашивания, который в случае $i = 1$ или $i = n$ содержит только две единицы (считаем, что номера координат идут слева направо, то есть в векторе $(1, 1, 0, \ldots, 0)$ первая и вторая координаты равны 1). Поэтому, это вектора $(1, 1, 0, \ldots, 0)$ и $(0, \ldots, 0, 1, 1)$ соответственно. Иначе i-ой клетке соответствует вектор $(0, \ldots, 0, 1, 1, 1, 0, \ldots, 0)$, в котором единицы на $i - 1$-ой, i-ой и $i + 1$-ой позициях.

Утверждение 1. Итоговый результат набора нескольких перекрашиваний не зависит от порядка операций.

Доказательство: утверждение очевидно, так как из сказанного выше следует, что вектор итогового узора равен вектору начального узора плюс сумма векторов перекрашиваний, и всё это по модулю 2 (по каждой координате независимо). Такое выражение не зависит от порядка слагаемых.

Утверждение 2. Итоговое количество возможных различных узоров не зависит от начальной покраски.

Доказательство: прибавив ко всем возможным итоговым узорам вектор начальной покраски (по модулю два покоординатно), получим набор из того же количества различных векторов. При этом легко видеть, что полученные вектора будут соответствовать тем же перекрашиваниям, что и ранее, но только относительно начальной покраски со всеми клетками второго цвета (то есть с вектором покраски, состоящим из всех нулей).

Таким образом, далее будем считать, что изначально полоса покрашена только во второй цвет. На ответ это не повлияет.

Утверждение 3. Для произвольного набора перекрашиваний найдётся набор с тем же итоговым узором, но в котором не будут присутствовать «дубли» — одинаковые перекрашивания два и более раз.

Доказательство: для доказательства достаточно сгруппировать повторяющиеся перекрашивания подряд (согласно утверждению 1 итоговый узор не поменяется), а затем исключать пары одинаковых векторов перекрашиваний, пока такие пары есть в наборе. Очевидно, при удалении любой такой пары итоговый узор не изменится, так как сумма векторов двух одинаковых перекрашиваний равна 0 (по модулю 2).

Таким образом, далее мы будем искать только количество узоров для наборов неповторяющихся перекрашиваний.

Утверждение 4. Если два различных набора перекрашиваний (каждый набор без дублей, наборы отличаются не только порядком входящих в них перекрашиваний) дают одинаковый итоговый узор, то после объединения таких наборов и избавления от дублей, получится непустой набор, дающий нулевой итоговый вектор перекрашивания.

Доказательство: просто рассмотрим этот объединённый набор. С одной стороны, так как итоговый узор равен сумме векторов перекрашиваний, то итоговый вектор перекрашивания объединённого набора — это сумма двух одинаковых векторов по

модулю 2. Но это нулевой вектор. Кроме того, в каждом из наборов нет дублей векторов. Это значит, что в объединённом наборе дубли берутся только из векторов, которые были представлены в единственном экзмепляре в обоих наборах. Но тогда если все вектора разбились на дубли, это значит, что в двух наборах содержатся одинаковые наборы векторов, что противоречит предположению. Утверждение доказано.

Утверждение 5. Пусть набор векторов перекрашиваний не пуст, не содержит дублей и в сумме даёт нулевое итоговое изменение. Тогда n сравнимо с 2 по модулю 3, и в набор входят все вектора перекрашиваний, соответствующие клеткам полосы с номером, не делящимся на 3.

Доказательство: рассмотрим произвольный такой непустой набор перекрашиваний без дублей, который в результате даёт нулевое изменение. Рассмотрим минимальный номер i клетки, для которой вектор перекрашивания входит в набор. Если i не равно 1, то, очевидно, $i-1$-ая клетка будет перекрашена, причём ровно 1 раз. Но значит, в результате эта клетка будет покрашена не как изначально — противоречие с условием. Значит, минимальный номер i равен 1. Следовательно, вектор перекрашивания для клетки 1 входит в набор. Так как эта клетка не должна поменять цвет в итоге, то также в набор должен входить вектор 2. Аналогично имеем, что вектор 3 не входит в набор. Далее аналогично легко установить, что вектора 4 и 5 должны присутствовать, а вектор 6 должен отсутствовать и так далее — все вектора с номерами кратными 3 должны отсутствовать, а остальные — присутствовать.

Остаётся заметить, что эти рассуждения верны и для другого конца полосы. Таким образом, вектора с номерами n и $n-1$ должны присутствовать в наборе, откуда имеем, что это клетки с номерами вида $3k+1$ и $3k+2$. В частности, получаем, что $n = 3k+2$, то есть сравним с 2 по модулю 3.

Тогда получаем, что в случае, когда n несравним с 2 по модулю 3, любой набор перекрашиваний сводится к набору перекрашиваний без дублей (с сохранением итогового узора), а все такие наборы дают различные результаты (иначе, если бы нашли два различных, дающих одинаковый результат, их объединение и удаление дублей дало бы непустой набор перекрашиваний с нулевым результатом (утверждение 4), но такого набора при n, несравнимом с 2, не существует (утверждение 5)). Значит, различных итоговых узоров столько же, сколько наборов перекрашиваний (без дублей) — это 2^n.

Если же n сравнимо с 2 по модулю 3, то аналогично любой набор сводится к набору без дублей, но есть единственный невырожденный «нулевой» набор (то есть дающего нулевой результат). При помощи него перекрашивание, задаваемое крайне правой клеткой, можно заменить на комбинацию нескольких других перекрашиваний. Но значит так, как перекрашивания задаются своими векторами при помощи суммы по модулю 2, то получается, это перекрашивание (задаваемое вектором номер n) можно заменить на эту комбинацию внутри любого набора из перекрашиваний не изменим итоговый результат (и отбросив дубли, если такие появились). Значит, все узоры, которые можно получить в этом случае, можно получить и при помощи первых $n-1$ векторов. Причём среди таких комбинаций уже не может быть пары различных наборов с одинаковым итоговым узором (иначе нулевая комбинация векторов должна была бы существовать и на первых $n-1$ координатах с нулевой последней координатой, но единственная нетривиальная нулевая комбинация имеет единицы на концах, как мы установили в утверждении 5 Таким образом, если n сравнимо с 2 по модулю 3, ответ равен 2^{n-1}.

Критерии оценивания

- Только ответ — 5 баллов.
- Переформулировано в терминах векторов — +2 балла.
- Каждое из сформулированных в доказательстве утверждений +2 балла (даже если не на языке векторов).
- Неправильный подсчёт финальных вариантов, когда уже доказано, что сводится к наборам без дублей — не более 15 баллов.