1. ПЕРВЫЙ ОТБОРОЧНЫЙ ЭТАП

Первый отборочный этап олимпиады НТИ проводится индивидуально в формате онлайн. Результаты оцениваются автоматически средствами системы тестирования. Для каждой возрастной категории (9 класс или 10-11 класс) предлагается набор задач по математике. Задачи по информатике являются общими для всех участников. Решения принимаются в виде программы на любом языке программирования, допущенном к использованию в ЕГЭ.

Для прохождения первого этапа участникам предоставляются три временных слота по два дня каждый. Это время отведено для решения задач по предметам, причём верное решение задачи дает определенное количество баллов. За каждую следующую попытку сдать решение баллы начисляются по формуле $\frac{P}{N}$, где P — максимальное количество очков, которые можно получить за решение задачи, N — количество попыток. Всего на каждую задачу математике давалось по две попытки, на каждую задачу по информатике — четыре попытки. Финальная оцека за отборочный этап является совокупной по математике и информатике — суммарно от 0 до 30 баллов.

Задачи, непосредственно связанные с машинным обучением, довольно сложны. Поэтому первый отборочный тур содержал задачи по математике и информатике повышенной сложности на стандартные школьные темы [1]. Это позволило отобрать школьников с необходимыми компетенциями по данным предетам.

Первая попытка

2.1. Задачи по математике (9 класс)

Задача 2.1.1. (2 балла)

Найдите сумму всех полных квадратов, которые при делении на 11 в частном дают простое число и в остатке 4.

Решение

$n^2 = 11p + 4 \Longrightarrow 11p = (n-2)(n+2)$. $n - 2 = 1$ не подходит, значит $n - 2 = 11$ и $n + 2 = p$ или $n - 2 = p$ и $n + 2 = 11$. В первом случае $p = 15$ не простое, во втором случае $n = 9$. (в наших рассуждениях n натуральное, p простое число)

Ответ: 81

[1] Отборочный этап олимпиады НТИ 2017/18 был составлен командой университета Иннополис

Задача 2.1.2. (3 балла)

Вася и Петя живут в одном доме у прямой автодороги. Чтобы сесть в школьный автобус, они одновременно вышли из дома, Петя пошел в сторону остановки А навстречу автобусу со скоростью 6 км/час, а Вася в сторону остановки В по направлению движения автобуса со скоростью 4 км/час. Они оба подошли к остановкам как раз к моменту прибытия автобуса. Найдите отношение расстояния пройденного Петей к расстоянию пройденного Васей, если скорость автобуса 60 км/час. Результат округлите до сотых.

Решение

Пусть Петя шел к своей остановке t часов и прошел расстояние $6t$ км. В момент, когда автобус подобрал его в остановке В, Вася находился на расстоянии $10t$. Автобус догоняет Васю со скоростью 56км/час и догонит через $\dfrac{10t}{56}$ часов. Вася к этому моменту пройдет $4t + \dfrac{4 \cdot 10t}{56} = \dfrac{66}{14}t$ километров. Отношение расстояний тогда равно $6t : \dfrac{66}{14}t = \dfrac{14}{11} = 1,(27)$.

Ответ: 1,27

Задача 2.1.3. (3 балла)

В автомат для размена денег загружены монеты в 2, 5 и 10 рублей. Сколько всего способов разменять 100 рублевую купюру в этом автомате?

Решение

Пусть x, y, z количество монет в 2, 5 и 10 рублей соответственно. Тогда $2x + 5y + 10z = 100$. Видим, что $x = 5k$, $y = 2n$, иначе уравнение не имеет решения в целых числах. Осталось найти количество неотрицательных целых решений уравнения $k + n + z = 10$. А это число сочетаний с повторениями $\overline{C}_3^{10} = C_{12}^{10} = 66$.

Ответ: 66

Задача 2.1.4. (3 балла)

На доске были написаны n первых натуральных чисел $(1, 2, \ldots n)$. Вася стер одно число. Петя заметил, что среднее арифметическое оставшихся стало $\dfrac{45}{4}$. Какое число стер Вася?

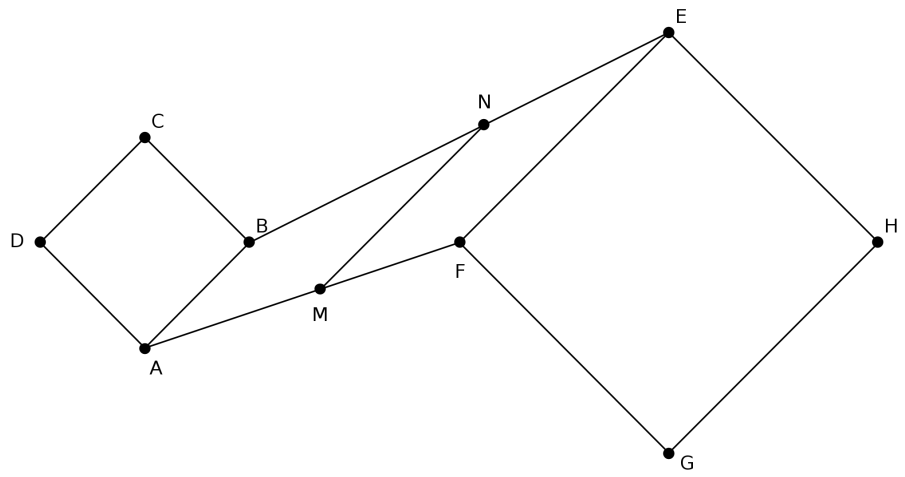
Решение

Из условия ясно, что $n > 1$. Пусть стерли число x. Тогда $x = \frac{n(n+1)}{2} - \frac{45}{4}(n - 1) = \frac{2n^2 - 43n + 45}{4} = \frac{(2n-41)(n-1)}{4} + 1$. По условию $1 \leqslant x \leqslant n$. Отсюда $0 \leqslant \frac{(2n-41)(n-1)}{4} \leqslant n - 1$, $0 \leqslant 2n - 41 \leqslant 4$. Решением неравенства являются только $n = 21$ или $n = 22$. При $n = 22$ число x не является целым, при $n = 21$ $x = 6$.

Ответ: 6

Задача 2.1.5. (4 балла)

На рисунке внизу $ABCD$ и $EFGH$ квадраты, их площади равны 32 см^2 и 66 см^2 соответственно. Отрезок MN параллелен основаниям трапеции $ABEF$ и делит её площадь пополам. Найдите длину отрезка MN.



Решение

Пусть прямые AB и EF пересекаются в точке K. Тогда $2S_{KMN} = S_{KBE} + SKAF$. Отсюда $2 = \frac{S_{KBE}}{S_{KMN}} + \frac{S_{KAF}}{S_{KMN}} = \frac{BE^2}{MN^2} + \frac{AF^2}{MN^2} = \frac{98}{MN^2}$.

Ответ: 7

2.2. Задачи по математике (10-11 класс)

Задача 2.2.1. (2 балла)

На плоскости отмечены 17 точек, не лежащих на одной прямой (т.е. найдутся хотя бы 3 точки, не лежащие на одной прямой). Через каждые две точки провели прямую. Какое наименьшее количество различных прямых могло получиться?

Решение

Пусть на одной прямой x точек, а на другой — $k-x$ точек (не считая общую). Прямые, проходящие через точки на различных прямых, не совпадают. Поэтому их $x(k-x)$. Эта величина уменьшается когда x стремится к 0 или k. Если точки перенесем на одну из отмеченных прямых, то количество прямых будет наименьшим. Все точки, кроме одной, лежат на одной прямой. Тогда количество различных прямых равно 17.

Ответ: 17

Задача 2.2.2. (3 балла)

Для проведения олимпиады преподаватели разбивают 60 школьников следующим образом: список в алфавитном порядке разбивается на 4 части, первая идет в первую аудиторию, вторая — во вторую и т. д. При этом в каждую аудиторию отправляется хотя бы один школьник. Сколькими способами можно произвести распределение?

Решение

В списке надо провести 3 разделительные черты. Эти черты можно проводить в промежутках между фамилиями. Надо выбрать 3 промежутка из 59 и поставить там разделительную черту. Всего способов $C_5 9^3 = \dfrac{59!}{3!56!}$.

Ответ: 32509

Задача 2.2.3. (3 балла)

Имеется таблица 100×100 натуральных чисел, в которой строки пронумерованы сверху вниз числами от 1 до 100. В k-ой строке таблицы слева направо написаны числа, которые являются арифметической прогрессией с первым членом 1 и разностью k. Какое наибольшее число встречается среди элементов диагонали, идущей из левой нижней клетки в правую верхнюю?

Решение

На отмеченной диагонали окажется $(101-k)$-ый член арифметической прогрессии, находящейся на k-ой строке. Тогда $x_{101-k} = 1+(100-k)k = 2501-(x-50)^2 \leqslant 2501$.

Ответ: 2501

Задача 2.2.4. (3 балла)

Бочка в виде прямого кругового цилиндра полностью заполнена водой. Радиус основания 40 см, а высота 120 см; бочка стоит на горизонтальной поверхности. На

какой угол необходимо наклонить эту бочка, чтобы вылить ровно половину воды. В ответ напишите тангенс этого угла, если необходимо, округлите до сотых.

Решение

Любая плоскость, проходящая через центр цилиндра, делит её на 2 равные части. Поэтому бочку надо наклонить так, чтобы плоскость, проходящая через точку на границе верхнего основания и центр цилиндра, стала горизонтальной. Тогда угол наклона станет равным углу между этой плоскостью и основанием. Тангенс этого угла равен отношению высоты бочки на диаметр основания, т.е. $\frac{120}{80}$.

Ответ: 1,5

Задача 2.2.5. (4 балла)

Найдите наибольшее натуральное значение n такое, что 120! делится без остатка на 12^n.

Решение

В 120! простое число p входит в степени $\left[\frac{120}{p}\right] + \left[\frac{120}{p^2}\right] + \left[\frac{120}{p^3}\right] + \ldots$. Таким образом находим степени 2 и 3 входящие в разложение 120! на простые множители:

$$120! = 2^{60+30+15+7+3+1} \cdot 3^{40+13+4+1} \cdot A = 12^{58} \cdot A,$$

где A не делится на 12.

Ответ: 58

2.3. Задачи по информатике

Задача 2.3.1. (1 балл)

Вам дана шахматная доска размером 8×8, в поле a которой находится король. Определите, за какое минимальное количество ходов король может добраться до клетки b.

Шахматный король может переместиться на любое соседнее поле, с которым есть хотя бы одна общая точка.

Формат входных данных

Первая строка входных данных содержит два числа a_v, a_h — координаты, где изначально находится король ($1 \leqslant a_v, a_h \leqslant 8$). Вторая строка также содержит два числа b_v, b_h — координаты клетки b, в таком же формате.

Формат выходных данных

Выведите одно целое число — ответ на задачу.

Примеры

Пример №1

Стандартный ввод
2 3 5 8
Стандартный вывод
5

Решение

Для минимизации количества ходов выгодно перемещать короля по диагонали, пока не дойдем до строки b_v или столбца b_h. Это $min(|b_v - a_v|, |b_h - a_h|)$ ходов. Оставшуюся часть доходим ходами по строке/столбцу, их всего $max(|b_v - a_v|, |b_h - a_h|) - min(|b_v - a_v|, |b_h - a_h|)$. Суммируем два ответа и получаем, что ответ равен $max(|b_v - a_v|, |b_h - a_h|)$.

Пример программы

Ниже представлено решение на языке C++

```cpp
#include <iostream>
#include <cmath>

using namespace std;

int main() {
    int av, ah, bv, bh;
    cin >> av >> ah >> bv >> bh;
    cout << max(abs(av - bv), abs(ah - bh)) << '\n';
}
```

Задача 2.3.2. (2 балла)

Фёдор перешел в 11 класс, и подготовка к ЕГЭ — его приоритетная цель в этом году.

Сегодня он узнал, что такое маска подсети, IP-адрес и адрес сети. Адрес сети получается как применение *поразрядной конъюнкции* (логическая операция И) к IP-адресу и маске подсети.

Поразрядная конъюнкция — применение к соответствующим битам операции логического И. Другими словами, если соответствующие биты равны 1, результирующий двоичный разряд будет равен 1, иначе 0.

Применение поразрядной конъюнкции к маске подсети и IP-адресу выглядит так:

```
11111111.11111111.11111111.00000000 (255.255.255.0)
&
11000000.10101000.00000000.00000001 (192.168.0.1)
-----------------------------------
11000000.10101000.00000000.00000000 (192.168.0.0) <--- Адрес сети
```

Вам дана маска подсети и n IP-адресов. Ваша задача определить сколько различных адресов сети получится.

Формат входных данных

В первой строке вам дана маска сети. Во второй строке дано числа n — количество IP-адресов ($1 \leqslant n \leqslant 1000$). Следующие n строк содержат IP-адреса.

Формат выходных данных

Выведите количество различных адресов сети.

Примеры

Пример №1

Стандартный ввод
255.255.255.0 3 192.168.0.1 192.168.0.28 192.168.1.1
Стандартный вывод
2

Решение

Давайте представим маску и IP-адреса как простые 32 битные числа, чтобы было удобно с ними работать.

Теперь получим все возможные адреса сети путем $\widehat{\text{П}}$оразрядная конъюнкция всех IP-адресов с маской. Дальше нужно убрать одинаковый адреса сети. Давайте отсортируем все получившееся адреса сети. Дальше не трудно посчитать сколько будет различных, так как одинаковые теперь стоять друг за другом.

Пример программы

Ниже представлено решение на языке C++

```
#include <iostream>
#include <vector>
#include <set>

using namespace std;

/*Функция address по 4м числам адреса возвращает 32 битовое число,
соответствующее этому адресу (на каждое из 4х чисел отводится по 8 бит).*/
int address(int p1, int p2, int p3, int p4) {
        int data = p1;
        data <<= 8;
```

```
12          data |= p2;
13          data <<= 8;
14          data |= p3;
15          data <<= 8;
16          data |= p4;
17          return data;
18  }
19
20  /*Функция read_ad считывает адрес в формате m1.m2.m3.m4 и
21  возвращает 32 битовое число, используя address.*/
22  int read_ad() {
23          int m1, m2, m3, m4;
24          char c;
25          cin >> m1 >> c >> m2 >> c >> m3 >> c >> m4;
26          return address(m1, m2, m3, m4);
27  }
28
29  set <int> s;
30
31  int main() {
32          int mask = read_ad();
33          int n;
34          cin >> n;
35          for (int i = 0; i < n; i++) {
36                  int x = read_ad();
37                  s.insert(mask & x);
38          }
39          /*В цикле проходимся по всем IP-адресам, получаем адрес сети
40          (mask & x) и добавляем в s (если такой адрес встречено впервые).*/
41          cout << s.size();
42          return 0;
43  }
```

Задача 2.3.3. (3 балла)

Юный поэт Марк хочет выпустить свой первый сборник стихотворений. Как вы знаете, рифма — одна из важных деталей любого стихотворения. И Марк хочет, чтобы во всех его стихах сохранялась рифма; но из-за недостатка опыта он еще не научился определять, имеет ли его стихотворение рифму. И вам необходимо ему помочь.

Стихотворение Марка представляет из себя n непересекающихся четверостиший. Четверостишие — 4 строки, состоящие из латинских букв и пробелов. Словом считается последовательность букв, не имеющая пробелов. Два слова имеют рифму, если не менее двух их последних букв совпадают. Две строки имеют рифму, если их последние слова рифмуются. Существует 3 способа задания рифмы в четверостишье:

1. Если независимо рифмуются строки номеров (1, 2) и (3, 4).
2. Если независимо рифмуются строки номеров (1, 3) и (2, 4).
3. Если независимо рифмуются строки номеров (1, 4) и (2, 3).

И наконец, стихотворение рифмуется, если все его четверостишья рифмуются одним и тем же способом. Получив стихотворение Марка, вам необходимо ответить, имеет ли оно рифму.

Формат входных данных

Первая строка входных данных содержит целое число $k = 4 * n$ — количество строк в стихотворении Марка ($4 \leqslant k \leqslant 1000$).

Далее следует стихотворение, в виде k строк, состоящих из латинских букв и пробелов (ни одна строка не начинается и не оканчивается пробелом). Длина каждой строки не превышает 1000 символов.

Формат выходных данных

Необходимо вывести *Yes*, если стихотворение рифмуется, а иначе — *No*.

Примеры

Пример №1

Стандартный ввод
4 If bees stay at home Rain will soon come If they fly away Fine will be the day

Стандартный вывод
Yes

Пример №2

Стандартный ввод
4 One two three Let me see I think about u I like u

Стандартный вывод
No

Пример №3

Стандартный ввод
8 I want a tree For shade and rest I want a tree Where birds can nest So I planted A green tree You must see Rest guaranteed

Стандартный вывод
No

Решение

Для того, чтобы понять рифмуются ли две строки, необходимо всего лишь сравнить на равенство по два последних символа строки(не должно быть пробелов). Теперь можно для каждого типа рифмы проверить, поддерживается ли он в каждом четверостишии стихотворения.

Пример программы

Ниже представлено решение на языке Python3

```
k = int(input())
k = k // 4

'''
Функция is_ryf проверяет две строки на рифму:
1)Если какая-то строка короче 2 символов, то рифмы нет
2)Если обе строки больше двух символов, то сравниваем 2 последние буквы
'''
def is_ryf(s1, s2):
        if len(s1[-1]) < 2 or len(s2[-1]) < 2:
                return False

        if (s1[-1][-2:] == s2[-1][-2:]):
                return True
        return False

t1 = True
t2 = True
t3 = True

for i in range(0, k):
        s1 = input().split()
        s2 = input().split()
        s3 = input().split()
        s4 = input().split()

'''
Проверяем стихотворение на наличие одной из рифм.
'''

        if not(is_ryf(s1, s2) and is_ryf(s3, s4) and t1):
                t1 = False
        if not(is_ryf(s1, s3) and is_ryf(s2, s4) and t2):
                t2 = False
        if not(is_ryf(s1, s4) and is_ryf(s2, s3) and t3):
                t3 = False

if t1 or t2 or t3:
        print("Yes")
else:
        print("No")
```

Задача 2.3.4. (4 балла)

Вам дан невырожденный треугольник и точка. Ваша задача состоит в том, чтобы определить периметр видимой части треугольника из данной точки.

Гарантируется, что точка находится строго вне треугольника.

Формат входных данных

В первой строке вам дано 6 чисел: $(x_1; y_1)$, $(x_2; y_2)$ и $(x_3; y_3)$ — точки образующие треугольник соответственно.

На следующей строке дано 2 числа — координаты точки $(x; y)$.

Все координаты точек целые и не превосходят по модулю 10^5.

Формат выходных данных

Вывести одно число: периметр видимой части треугольника из точки $(x; y)$.

Ответ будет засчитан, если он отличается от правильного не более чем на 10^{-6} относительной или абсолютной погрешности.

Примеры

Пример №1

Стандартный ввод
0 1 1 0 1 1 2 2
Стандартный вывод
2.0000000000

Пример №2

Стандартный ввод
0 1 1 0 1 1 2 1
Стандартный вывод
1.0000000000

Пример №3

Стандартный ввод
0 0 0 1 1 0 1 1
Стандартный вывод
1.4142135624

Решение

Если мы зафиксируем какой-нибудь отрезок треугольника, то когда его будет видно? Логично, что его будет видно, если точка, из которой мы смотрим, находится по другую сторону прямой, образованной этим отрезком, для третьей точки треугольника.

Осталось научиться проверять, лежат ли две точки по разные стороны от прямой. Это можно делать с помощью длины векторного произведения. Для удобства введем обозначения: t_1, t_2, t_3 — точки треугольника, p — точка, из которой смотрим. Возьмем длину векторного произведения c_1 для векторов из точки t_1 в точку t_2 и из t_1 в точку p. Аналогично c_2 для векторов из точки t_1 в точку t_2 и из t_1 в точку t_3. Если знаки c_1 и c_2 отличаются, то этот отрезок виден из точки, и нужно прибавить к ответу длину этого отрезка.

Пример программы

Ниже представлено решение на языке C++

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <cmath>

using namespace std;

typedef long long ll;

//структура для точки и подгрузка оператора для нее
struct Point {
        int x, y;
        Point() {}
        Point(int x, int y) : x(x), y(y) {}
};

Point operator-(Point &p1, Point &p2) {
        return Point(p2.x - p1.x, p2.y - p1.y);
}

istream & operator >>(istream &is, Point &p) {
        is >> p.x >> p.y;
        return is;
}

ostream & operator <<(ostream &os, Point &p) {
        os << p.x << ' ' << p.y;
        return os;
}

//Возвращает длину вектроного произведения
ll crossproduct(Point const &p1, Point const &p2) {
        return (ll)p1.x * p2.y - p2.x * p1.y;
}

//Реализует условие, описанное в решении, как две точки  лежат относительно прямой.
int side(Point &p1, Point &p2, Point &p) {
        ll cross = crossproduct(p2 - p1, p - p1);
        if (cross == 0) return 0;
        if (cross > 0) return 1;
        else return -1;
}

//Возвращает расстояние между точками
double dis(Point &p1, Point &p2) {
        return sqrt(((double)p2.x - p1.x) * (p2.x - p1.x) +
```

```
47                      ((double)p2.y - p1.y) * (p2.y - p1.y));
48  }
49
50  int main() {
51          vector <Point> tri(3);
52          for (auto &i : tri) cin >> i;
53          Point p;
54          cin >> p;
55          double ans = 0;
56          //Делаем првоерку каждой стороны
57          for (int i = 0; i < 3; i++) {
58          if (side(tri[i], tri[(i + 1) % 3], p) *
59              side(tri[i], tri[(i + 1) % 3], tri[(i + 2) % 3]) == -1)
60                  ans += dis(tri[i], tri[(i + 1) % 3]);
61          }
62          cout << fixed << setprecision(15) << ans;
63      return 0;
64  }
```

Задача 2.3.5. (5 баллов)

Программист Искандер уже 14-е сутки подряд пишет программу, и вот, наконец-то, весь проект был готов. Однако, скомпилировав и запустив программу, результат выполнения оказался не таким, как ожидал Искандер...

Искандер — опытный программист, и для удобства он разделил программу на n исходных файлов, каждый из которых мог использовать некоторые из остальных исходных файлов (для простоты он пронумеровал файлы целыми числами от 1 до n). После завершения выполнения программы Искандер тут же осознал, в каких k исходных файлах допущены ошибки. Исправив всё, что нужно, теперь осталась только перекомпиляция, но в повторной компиляции нуждались не только исправленные исходные файлы, но и те файлы, которые непосредственно или через другие файлы используют какие-либо файлы из изменённых.

Искандер — ленивый программист, поэтому он хочет перекомпилировать как можно меньше исходных файлов. Помогите ему посчитать количество таких файлов.

Вспоминая, что Искандер – опытный программист, можно гарантированно сказать, что он не допустил ситуации, когда один из исходных файлов использует самого себя (непосредственно или через другие файлы).

Формат входных данных

В первой строке задано целое число n ($1 \leqslant n \leqslant 5000$).

Далее идут n строк. Каждая следующая i-я строка начинается с числа p – количество исходных файлов, которые используются в i-м файле ($0 \leqslant p \leqslant n - 1$). Затем следуют p различных целых чисел – номера исходных файлов, которые используются в i-м файле. Все числа положительные и не превосходят n. Суммарное количество p по всем строкам не превышает 10^5.

Затем задано целое число k ($1 \leqslant k \leqslant n$). Следующая строка содержит k различных целых чисел – номера файлов, которые исправил Искандер. Все числа положительные и не превосходят n.

Формат выходных данных

Выведите одно число — минимальное количество файлов, требующих перекомпиляции.

Примеры

Пример №1

Стандартный ввод
5 2 2 3 0 1 4 0 0 2 2 4
Стандартный вывод
4

Решение

Представим зависимости файлов между собой как ориентированный граф. Вершина с номером i будет соответствовать файлу с таким же номером. Если в текущем файле i используется файл j, то добавляем ребро из вершины с номером j в вершину с номером i. Теперь рассмотрим, какие из файлов надо перекомпелировать: запустим обход в глубину от каждой из k вершин, соответствующих изменённым файлам, и будем помечать все вершины на пути. Сделаем небольшую оптимизацию: не будем запускать обход в глубину из уже помеченных вершин. Таким образом, каждая вершина будет посещена не более 1 раза. Ответом на задачу является количество помеченных в графе вершин.

Пример программы

Ниже представлено решение на языке Python3

```
used = [];
v = [];

#Реализация поиска в глубину с небольшой модификацией, описанной в решении.
def dfs(x):
    used[x] = True
    for i in range(len(v[x])):
        to = int(v[x][i])
        if used[to] == False:
            dfs(to)

n = int(input())

v = [[] for i in range(n)]
used = [False] * n

```

```
#Считываем информацию о каждом файле и создаем матрицу связности графа.
for i in range(n) :
    p = input().split();
    for j in range(len(p)) :
        p[j] = int(p[j]) - 1
    for j in range(1, len(p)):
        v[p[j]].append(i);

k = int(input())

if k > 0:
    edit = input().split()

    for i in range(k):
        edit[i] = int(edit[i]) - 1

    #Запускаем dfs от каждого исправленного файла.
    for i in range(k):
        dfs(edit[i])

ans = 0;

#Смотрим сколько вершин пометили.
for i in range(n):
    if used[i] == True:
        ans = ans + 1;

print(ans);
```

Вторая попытка

3.1. Задачи по математике (9 класс)

Задача 3.1.1. (2 балла)

С числом можно проделывать 2 типа операций: операция A умножает число на 2, операция B вычитает 3. Как из 11 при помощи этих операций получить 25 за наименьшее количество ходов? В ответ напишите последовательность операций (например последовательность операций AAABBBBBBBBBBBBBBBBBBBB за 24 хода достигает результата).

Решение

Обозначим M_n множество чисел, из которых можно получить 25 за n ходов. Тогда $M_1 = \{28\}$, $M_2 = \{14, 31\}$, $M_3 = \{7, 17, 34\}$, ... На 7-ом шаге встречаем 11.

Ответ: BBABAAB

Задача 3.1.2. (4 балла)

При каком наименьшем значении параметра k корни уравнения

$$x^2 + (k-1)x - k = 0$$

удовлетворяют условию $x_1^2 + x_2^2 = 5$.

Решение

$x_1^2 + x_2^2 = (x_1 + x_2)^2 - 2x_1x_2 = (k-1)^2 + 2k = 5$. Откуда $k = \pm 2$.

Ответ: -2

Задача 3.1.3. (3 балла)

В выпуклом пятиугольнике $ABCDE$ известно, что $AB = AE = 3$, $CD = 2$, $BC = 0,8$, $DE = 1,2$, $\angle ABC = \angle DEA = 90°$. Найдите площадь этого пятиугольника.

Решение

Разделим пятиугольник на 3 треугольника ABC, ADE и ACD. Площади первых двух треугольников находяться по уже известным данным, поскольку треугольники прямоугольные, и известно две их стороны.

Стороные AC и AD треугольников прямоугльных треугольников ABC и ADE находятся также по уже известным данным. Зная все три стороны треугольника ACD, можно найти его площадь.

Останется только сложить площади всех трех треугольников.

Ответ: 6

Задача 3.1.4. (3 балла)

Натуральное число n называется «приятным», если оно удовлетворяет следующим условиям:

- состоит из 4 цифр;
- первая цифра равна третьей;
- вторая цифра равна четвертой;
- n^2 делится на произведение цифр n.

Найдите сумму всех «приятных» чисел.

Решение

$n = \overline{abab} = \overline{ab} \cdot 101$, $n^2 \vdots (a \cdot b \cdot a \cdot b) \Longrightarrow n \vdots ab$. н.о.д.$(101, ab) = 1 \Longrightarrow \overline{ab} \vdots ab$. Отсюда получаем, что $(10a + b)$ делится на a, т.е. $b \vdots a$. Небольшим перебором находим все

значения $\overline{ab}$: 11, 12, 15, 24, 36. Сумма всех «приятных» чисел равна $(11 + 12 + 15 + 24 + 36) \cdot 101$.

Ответ: 9898

Задача 3.1.5. (4 балла)

Найдите количество решений в целых числах уравнения

$$\frac{1}{x} + \frac{1}{y} = \frac{1}{2017}.$$

Решение

Уравнение приводится к виду $y = 2017 + \dfrac{2017^2}{x - 2017}$. Тогда $x - 2017$ является целым делителем 2017^2. Таких значений шесть: ± 1, ± 2017, $\pm 2017^2$. В каждом случае получаем пару целых решений.

Ответ: 6

3.2. Задачи по математике (10-11 класс)

Задача 3.2.1. (2 балла)

Найдите наименьшее число, которое можно представить в виде суммы квадратов натуральных чисел двумя различными способами.

Решение

$65 = 4^2 + 7^2 = 1^2 + 8^2$. Перебирая все суммы квадратов, меньших 64, выясняем минимальность.

Ответ: 65

Задача 3.2.2. (3 балла)

Найдите значение выражения

$$[2\sqrt{1 \cdot 2 \cdot 3 + 2 \cdot 3 \cdot 4 + \ldots + 2016 \cdot 2017 \cdot 2018}].$$

($[x]$ — целая часть числа x.)

Решение

$$1 \cdot 2 \cdot 3 + 2 \cdot 3 \cdot 4 + \ldots + 2016 \cdot 2017 \cdot 2018 = \\ = \frac{1}{4} \cdot 2016 \cdot 2017 \cdot 2018 \cdot 2019 = 4141845698256.$$

Остальное вычисляем на калькуляторе.

Ответ: 4070304

Задача 3.2.3. (3 балла)

На координатной плоскости отмечены точки $A(2;-1)$, $B(3;1)$ и $C(2017;4034)$. Найдите площадь треугольника ABC.

Решение

Пусть $O(0;0)$. Тогда $\overrightarrow{AB}\{1;2\} \uparrow\uparrow \overrightarrow{OC}\{2017;4034\}$. Поэтому площадь $\triangle ABC$ равна площади $\triangle OAB$.

Ответ: 2,5

Задача 3.2.4. (3 балла)

Обозначим $\tau(n)$ количество различных натуральных делителей числа n, а $\sigma(n)$ сумму различных натуральных делителей числа n. Найдите $\tau(\sigma(12)!)$.

Решение

$\sigma(12) = (1+2+4) \cdot (1+3) = 28$, $28! = 2^{25} \cdot 3^{13} \cdot 5^6 \cdot 7^4 \cdot 11^2 \cdot 13^2 \cdot 17 \cdot 19 \cdot 23$, $\tau(28!) = 26 \cdot 14 \cdot 7 \cdot 5 \cdot 3 \cdot 3 \cdot 2 \cdot 2 \cdot 2$.

Ответ: 917280

Задача 3.2.5. (4 балла)

Найдите все значения параметра p, при которых уравнение

$$27x^3 + (26 - 2p)x^2 + 16p^3 = 0$$

имеет три различных корня, образующих арифметическую прогрессию. В ответ напишите сумму полученных значений.

Решение

По теореме Виета для кубического многочлена:

$$\begin{cases} x_1 + x_2 + x_3 = -\dfrac{26 - 2p}{27} \\ x_1x_2 + x_2x_3 + x_3x_1 = 0 \\ x_1x_2x_3 = -\dfrac{16p^3}{27} \end{cases}$$

В эту систему добавляем условие арифметической прогрессии $x_1+x_3 = 2x_2$ и находим единственное решение.

Ответ: -0,5

3.3. Задачи по информатике

Задача 3.3.1. (1 балл)

Никита работает на автостоянке. В его обязанности входит запись номеров въезжающих машин. Это довольно скучное занятие и поэтому Никита решил оптимизировать этот процесс. Он хочет, чтобы компьютер обрабатывал изображение с камеры перед въездом в автостоянку и записывал номера. Никита уже написал софт, который обнаруживает на изображении последовательности из шести символов; осталось только проверять, является ли данная последовательность номером. Но из-за того, что Никита постоянно отвлекается на запись номеров, он просит вас о помощи.

Автомобильный номер – строка из шести символов. Первый символ – заглавная латинская буква, далее следует 3 цифры, и после — две заглавные латинские буквы. Например, строка "P142EQ" является номером. Вам будет дана строка, состоящая из шести символов, необходимо ответить, является ли строка автомобильным номером.

Формат входных данных

В единственной строке находится строка из шести символов, состоящая из цифр и заглавных латинских букв.

Формат выходных данных

Если строка является автомобильным номером, то необходимо вывести *Yes*, в ином случае — *No*.

Примеры

Пример №1

Стандартный ввод
`K040LE`

Стандартный вывод
`Yes`

Пример №2

Стандартный ввод
M3239L

Стандартный вывод
No

Решение

Для того, чтобы решить задачу необходимо проверить, что на первой и двух последних позициях строки находятся заглавные латинские буквы ($'A' \leqslant c \leqslant' Z'$); а между буквами находились три числа ($'0' \leqslant c \leqslant' 9'$).Для того, чтобы решить задачу необходимо проверить, что на первой и двух последних позициях строки находятся заглавные латинские буквы ($'A' \leqslant c \leqslant' Z'$).

Пример программы

Ниже представлено решение на языке C++

```
#include <iostream>
#include <string>

using namespace std;

//Проверяем символ, является ли он цифрой
bool is_dig(char c) {
        if ('0' <= c && c <= '9') {
                return true;
        }
        return false;
}
//Проверяем символ, является ли он заглавной латинской буквой
bool is_let(char c) {
        if ('A' <= c && c <= 'Z') {
                return true;
        }
        return false;
}

int main() {
        string s;
        cin >> s;
        //Проверяем условия задачи с помощью написанных выше функций.
        if (is_let(s[0]) && is_dig(s[1]) && is_dig(s[2])
           && is_dig(s[3]) && is_let(s[4]) && is_let(s[5])) {
                cout << "Yes\n";
        } else {
                cout << "No\n";
        }
}
```

Задача 3.3.2. (1 балл)

Студент Денис часто прогуливает свои пары в университете. Сегодня, когда Денис пришел в деканат, он узнал, что нужно за каждую пропущенную пару написать

объяснительную.

Денису выдали один бланк для написания объяснительной. Ему нужно сделать еще n копий данного бланка. Для этого в его распоряжении есть два ксерокса. Первый ксерокс тратит на копирование одного листа одну секунду, а второй — две секунды. Копию можно делать как с оригинала, так и с копии. Денис может использовать оба ксерокса одновременно. Определите, какое минимальное время необходимо Денису для получения n копий объяснительной.

Формат входных данных

В единственной строке дано число n — необходимое число копий ($1 \leqslant n \leqslant 1000$).

Формат выходных данных

Выведите одно число — минимальное время в секундах, необходимое для получения n копий.

Примеры

Пример №1

Стандартный ввод
2
Стандартный вывод
2

Пример №2

Стандартный ввод
5
Стандартный вывод
4

Решение

Изначально нужно скопировать один экземпляр, чтобы одновременно копировать на обоих ксероксах. После этого для печати трех экземпляров необходимо 2 секунды. Получается, что для печати $n-1$ копий необходимо затратить $(n-1)//3*2$ секунд в случае, если $n-1$ делится на 3. В случае остатка от деления на 3, равному k, необходимо дополнительно потратить k секунд. Таким образом, итоговая формула выглядит, как $1 + (n-1)//3 * 2 + (n-1)\%3$.

Пример программы

Ниже представлено решение на языке C++

```cpp
#include <iostream>

using namespace std;

int main() {
        int n;
        cin >> n;
        cout << 1 + (n - 1) / 3 * 2 + (n - 1) % 3<< "\n";
}
```

Задача 3.3.3. (3 балла)

Сегодня Ринат на уроке узнал про новую битовую операцию – XOR двух чисел. Напомним, что XOR или Исключающим ИЛИ, называется бинарная операция, которая применяется к каждой паре битов, стоящих на одинаковых позициях в двоичных представлениях чисел так, что, если биты равны, то в результате на этой позиции стоит 0, если же биты различны, то 1. Например, 3 XOR 5 = 6, потому что $3_{10} = 011_2$, $5_{10} = 101_2$, поэтому после применения операции второй и третий бит становятся равными 1, а первый бит – 0, таким образом получается $110_2 = 6_{10}$.

Учитель дал Ринату массив различных чисел и попросил написать массив, в котором все числа различны и нет ни одного числа из массива, который дал учитель. При этом, чтобы XOR всех чисел в массиве Рината был равен XOR всех чисел в массиве учителя.

Формат входных данных

В первой строке дано число n — количество элементов в массиве учителя ($1 \leqslant n \leqslant 10^5$).

Во второй сроке дано n чисел a_i — элементы массива ($0 \leqslant a_i \leqslant 2^{30}$).

Гарантируется, что XOR всех a_i не равен 0.

Формат выходных данных

Выведите элементы вашего массива, чтобы все числа в вашем массиве и массиве учителя отличались.

Все числа вашего массива b_i должны находиться в промежутке ($0 \leqslant b_i \leqslant 2^{30}$).

Количество элементов не должно превосходить 10^5.

Примеры

Пример №1

Стандартный ввод
1 27

Стандартный вывод
21 14

Пример №2

Стандартный ввод
2 5 6
Стандартный вывод
7 4

Пример №3

Стандартный ввод
3 1 2 4
Стандартный вывод
5 3 9 8

Пример №4

Стандартный ввод
3 3 5 1
Стандартный вывод
10 9 6 2

Решение

Стоит подумать о том, какая длина массива нам нужна. Давайте подумаем, можно ли это сделать с одним элементом? Возьмем $x = \mathbf{XOR}\ a_i$. Тогда наш ответ и есть этот x, но он может быть в массиве учителя.

Можно ли это сделать из двух элементов? Давайте переберем число i. Тогда вторым числом должно быть i **XOR** x. Если число i и число i **XOR** x не были в массиве учителя, то мы нашли ответ.

Давайте подумает за сколько итераций завершится цикл, если просто перебирать i увеличивая на 1. А цикл завершится не больше, чем за $2 \cdot n + 1$ итераций. Потому что все числа разбиваются на пары i и i **XOR** x. Всего n элементов могут покрыть не более n пар, поэтому $2 \cdot n$ чисел i могут не подойти.

Пример программы

Ниже представлено решение на языке C++

```
#include <iostream>
#include <vector>
#include <set>


using namespace std;

```

```cpp

set <int> s;

//Проверка на наличие в s
bool contains(int n) {
    return s.find(n) != s.end();
}

int main() {
        int n;
        cin >> n;
        int now = 0;
        vector <int> v(n);
        for (auto &i : v) {
            cin >> i;
            s.insert(i);
            //делаем xor всех
            now ^= i;
        }
        for (int i = 0; ; i++) {
            //Делаем проверку из условия i != i XOR x, заодно проверяем на наличие в s.
            if (i != (i ^ now) && !contains(i) && !contains(i ^ now)) {
                cout << i << ' ' << (i ^ now) << '\n';
                return 0;
            }
        }
        return 0;
}
```

Задача 3.3.4. (5 баллов)

Азат уже очень давно отдыхает на островах. Сегодня он решил, что хочет перебраться на другой остров. Все острова пронумерованы целыми числами. Какие-то из островов соединены мостами. Но вот незадача, какие-то из островов ушли под воду и попасть на них никак нельзя. Азат сейчас находится на острове с номером v и хочет попасть на остров с номером u. Ваша задача проверить, сможет ли он это сделать.

Формат входных данных

В первой строке содержится 4 числа: n, m, v, и u ($1 \leqslant n \leqslant 1000, 0 \leqslant m \leqslant 5000, 1 \leqslant v, u \leqslant n$). Количество островов, количество мостов, номер острова, на котором находится Азат, и номер острова, на который он хочет попасть.

Следующие m строк описывают мосты. Каждая строка содержит 2 числа: a, b ($1 \leqslant a, b \leqslant n$). Мост между островами с номерами a и b соответственно. По мосту можно попасть с острова a на b, так и с острова b на a.

Следующая строка содержит одно число k ($0 \leqslant k \leqslant n$). Количество островов, которые затонули.

Следующая строка содержит k чисел c_i ($1 \leqslant c_i \leqslant n$) — номера затонувших островов. Гарантируется, что острова u и v не затоплены. Если k равно 0, то данная строка отсутствует.

Формат выходных данных

Если можно добраться от острова v до острова u, не посещая затопленные острова, то необходимо вывести *YES*, в ином случае — *NO*.

Примеры

Пример №1

Стандартный ввод
4 4 1 4 1 2 2 3 3 4 1 3 1 2
Стандартный вывод
YES

Пример №2

Стандартный ввод
5 6 1 5 1 2 2 3 3 4 4 5 1 3 2 4 2 2 3
Стандартный вывод
NO

Решение

Давайте применим известный алгоритм поиска в глубину(DFS). Будем хранить массив $used[i]$, где будем стоять 1, если мы были на острове с номером i, или он затоплен, и 0, если мы там не были.

Изначально пометим все затопленные острова в массиве *used* как 1. Потом запустим DFS от острова, в котором стоит Азат и будем пытаться попасть на остров, с которым мы связаны мостом, но мы не были на нем, и он не затоплен. Если таки образом мы сможем попасть на остров u, то ответ YES, иначе NO.

Пример программы

Ниже представлено решение на языке C++

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

using namespace std;

int const MAX = (int)1e5;

bool used[MAX];

//Стандартная реализация DFS с массивом used
void dfs(int v, vector < vector <int> > &vv) {
        used[v] = true;
        for (auto i : vv[v]) {
                if (used[i]) continue;
                dfs(i, vv);
        }
}

int main() {
        int n, m, a, b;
        cin >> n >> m >> a >> b;
        --a; --b;
        vector < vector <int> > vv(n);

        //Считываем вершину и заполняем "матрицу смежности"
        for (int i = 0; i < m; i++) {
                int f, t;
                cin >> f >> t;
                --f; --t;
                vv[f].push_back(t);
                vv[t].push_back(f);
        }
        int k;
        cin >> k;
        //Заполняем used затопленными островами
        for (int i = 0; i < k; i++) {
                int x;
                cin >> x;
                used[x - 1] = true;
        }
        //Запускаем DFS
        dfs(a, vv);
        //Посетили ли мы нужный остров?
        used[b] ? cout << "YES\n" : cout << "NO\n";
        return 0;
}
```

Задача 3.3.5. (5 баллов)

На прямой отмечено n точек. Нужно соединить некоторые точки отрезками одинаковой длины, чтобы можно было по отрезкам дойти от первой точки до последней, причём можно использовать не более k отрезков. Среди всех таких длин отрезков требуется вывести минимальную.

Формат входных данных

В первой строке заданы целые числа n и k ($2 \leqslant n \leqslant 2500$, $1 \leqslant k \leqslant n - 1$).

На следующей строке в возрастающем порядке заданы n различных неотрицательных целых чисел, каждое из которых не превосходит 100 000.

Формат выходных данных

Выведите одно целое число — ответ на задачу.

Примеры

Пример №1

Стандартный ввод
5 3 1 2 3 4 5
Стандартный вывод
2

Пример №2

Стандартный ввод
9 3 1 3 4 5 7 8 9 10 13
Стандартный вывод
4

Решение

Перебираем все возможные длины отрезка: отрезок длиной равной расстоянию от первой точки до второй, от первой точки до третьей и т.д. Для каждого такого отрезка пробуем соединить некоторые точки: начинаем с первой точки и проверяем, существует ли точка на позиции, равной сумме координаты текущей точки и длины рассматриваемого отрезка. Если такой точки не существует, то соединить точки отрезком текущей длины невозможно, иначе переходим к точке, существование которой мы проверяли. Если таким образом мы смогли дойти до последней точки, и при этом было использовано не более k отрезков, то текущая длина и является ответом.

Пример программы

Ниже представлено решение на языке Python3

```
A = input().split();
n = int(A[0]);
k = int(A[1]);

x = input().split();

for i in range(n) :
    x[i] = int(x[i]);

```

```
#Пробегаем по всем отрезкам от 0..1 до 0..n.



for i in range(1, n) :
    len = int(x[i]) - int(x[0]);
    cnt = 0;
    now = x[0];
    #Пытаемся последовательно добраться такими отрезками до конца
    while (now != x[n - 1]) :
        now += len;
        if now not in x:
            cnt = -1;
            break;
        cnt = cnt + 1;
    #Если на каком-то шаге ответ найден - прирываем цикл.
    if cnt != -1 and cnt <= k :
        print(len);
        break;
```

Третья попытка

4.1. Задачи по математике (9 класс)

Задача 4.1.1. (2 балла)

Имеется 4 попарно различных по весу камней и двухчашечные весы без гирь. За одно взвешивание можно положить по камню на каждую из чаш и узнать какой из камней более тяжелый. За какое наименьшее число взвешиваний можно упорядочить камни по возрастанию?

Решение

Пусть имеются камни весами a, b, c, d. Если взвешивать все пары камней, то получится 6 взвешиваний. Надо предсказать результат одного взвешивания и нам не придется его совершать. Взвешиваем a с b и c с d. Пусть a и c окажутся более тяжелыми. Еще за одно взвешивание из них выберем более тяжелый. Тогда этот камень не надо взвешивать с более легким из другой пары камней (очевидно он тяжелей).

Ответ: 5

Задача 4.1.2. (2 балла)

Чему равен наибольший общий делитель пары чисел 7808809 и 7182391?

Решение

Используем алгоритм Евклида.

Ответ: 2677

Задача 4.1.3. (3 балла)

Три генератора тактовой частоты начинают вырабатывать импульсы одновременно. Интервалы между импульсами составляют 4/3 секунды, 5/3 секунды и 2 секунды соответственно. Совпавшие во времени импульсы накладываются (обрабатываются компьютером за один). Сколько тактов будет обработано за 1 минуту? (Первый импульс также считается.)

Решение

Первый генератор выработает 45 импульсов, второй — 36, третий — 30. Но мы посчитали совпадающие несколько раз: попарно совпадающие по два раза, все три совпадения по три раза. У первого и второго генератора совпадение импульсов каждые 20/3 секунды, у первого и третьего — каждые 4 секунды, у второго и третьего — каждые 10 секунд. Тогда попарное совпадение импульсов $9 + 15 + 6$ раз. В эту сумму тройное совпадение вошло 3 раза. Все три импульса совпадают $60 : \frac{60}{3} = 3$ раза. Общее количество импульсов за минуту равно:

$$45 + 36 + 30 - (9 + 15 + 6) + 3 = 84.$$

Ответ: 85

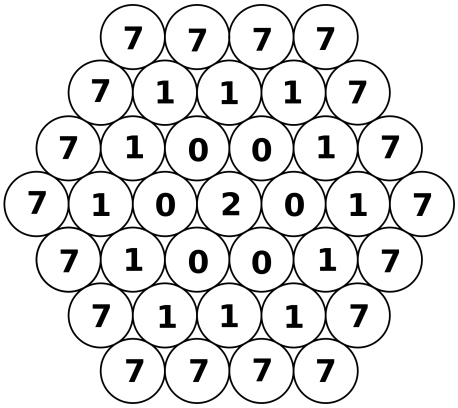
Задача 4.1.4. (4 балла)

Требуется выбрать 4 круга с цифрами 2, 0, 1 и 7 на рисунке ниже так, чтобы круг с цифрой 2 касался круга с 0, а круг с 0 касался круга с 1, круг с 1 касался круга с 7. Сколько способов это сделать?

Решение

Рассмотрим круги с 1. Если брать 1 из углового круга, то к нему от 2 ведет только 1 путь и от него можно пройти к 7 ровно 3 способами. Таких кругов с 1 ровно 6, поэтому всего путей через угловые 1 ровно 18. Если же брать круги с 1 на середине стороны шестиугольника, то к нему от 2 можно попасть через 2 нулика, а от него к 7 можно пройти тоже 2 способами. Таких путей всего $6 \cdot 2 \cdot 2 = 24$.

Ответ: 42



Задача 4.1.5. (4 балла)

Из одной точки окружности проведены две хорды длиной 9 и 17. Найдите радиус окружности, если расстояние между серединами хорд равно 5.

Решение

Расстояние между другими концами этих хорд равно 10 (по теореме о средней линии). Теперь надо выяснить радиус окружности, описанной вокруг треугольника со сторонами 9, 10, 17. Его можно выяснить по формулам $R = \frac{abc}{4S}$, и $S = \sqrt{p(p-a)(p-b)(p-c)}$.

Ответ: 10,625

4.2. Задачи по математике (10-11 класс)

Задача 4.2.1. (2 балла)

Имеется 5 попарно различных по весу камней и двухчашечные весы без гирь. За одно взвешивание можно положить по камню на каждую из чаш и узнать какой из камней более тяжелый. За какое наименьшее число взвешиваний можно упорядочить камни по возрастанию?

Решение

Пусть веса наших камней равны a, b, c, d, e. Первыми двумя взвешиваниями сравниваем пары (a, b) и (c, d). Не умаляя общности можно считать, чтобы $a > b$ и $c > d$. Третьим взвешиванием сравниваем пару (a, c). Не умаляя общности можно считать, что $a > c > d$ и $a > b$. За следующие 2 взвешивания вставляем камень e в цепочку $a > c > d$. А так как мы еще знаем, что $a > b$, то за последние 2 взвешивания вставляем b в нашу полученную цепочку.

Ответ: 7

Задача 4.2.2. (2 балла)

При каком наибольшем n на шахматной доске можно расставить n чёрных и n белых королей так, чтобы чёрные не били белых, а белые — чёрных?

Решение

Группа белых королей от группы черных королей должна быть отделена полосой пустых клеток. Если полоса в 8 пустых клеток, то в одной из частей не более 24 королей. Для полосы в 10 клеток пример строится несложно:

б	б	б	б		ч	ч	ч
б	б	б	б		ч	ч	ч
б	б	б	б		ч	ч	ч
б	б	б			ч	ч	ч
б	б	б			ч	ч	ч
б	б	б		ч	ч	ч	ч
б	б	б		ч	ч	ч	ч
б	б	б		ч	ч	ч	ч

Ответ: 27

Задача 4.2.3. (3 балла)

К числу 2017 припишите слева и справа по одной цифре так, чтобы полученное шестизначное число делилось на 31. Какие числа получились? В ответ напишите сумму таких чисел.

Решение

$\overline{a2017b} = 100000a + 20170 + b \equiv 25a + b - 11 \equiv 0 \pmod{31}$. Откуда $b \equiv 6a + 11 \pmod{31}$. Подставляя все цифры от 1 до 9 вместо a находим b. Нам подойдут $a = 4$, $b = 4$ и $a = 9$, $b = 3$.

Ответ: 1340347

Задача 4.2.4. (4 балла)

Рассматриваются всевозможные расстояния от точки $A(28; 13)$ до точек параболы $y = x^2$. Найдите среди них кратчайшее. В ответ запишите квадрат этого расстояния.

Решение

Пусть $M(x; x^2)$. Введем функцию $f(x) = AM^2 = (x - 28)^2 + (x^2 - 13)^2 = x^4 - 25x^2 - 56x + 28^2 + 13^2$. Требуется найти наименьшее значение $f(x)$. Оно достигается в

точках минимума. $f'(x) = 4x^3 - 50x - 56 = 2(x-4)(2x^2+8x+7)$ имеет единственную точку минимума $x = 4$. $f(4) = 576 + 9 = 585$.

Ответ: 585

Задача 4.2.5. (4 балла)

В окружности проведены хорды AC и BD, пересекающиеся в точке E, причем касательная к окружности, проходящая через точку A, параллельна BD. Известно, что $CD : ED = 4 : 3$, $S_{\triangle ABE} = 72$. Найдите площадь треугольника ABC.

Решение

Прямая, проходящая через A перпендикулярно касательной, проходит через центр окружности. Она же перпендикулярна хорде BD, следовательно, делит её пополам. Треугольник ABD равнобедренный. Используя теорему о вписанном угле выясняем справедливость следующих равенств:

$$\angle ACD = \angle ABD = \angle ADB = \angle ACB.$$

Из этих равенств очевидно следуют подобия треугольников:

$$\triangle ABC \sim \triangle AEB \sim \triangle DCE.$$

Откуда получаем $\dfrac{AB}{AE} = \dfrac{CD}{DE} = \dfrac{4}{3}$.

$$\frac{S_{\triangle ABC}}{S_{\triangle AEB}} = \left(\frac{AB}{AE}\right)^2 = \frac{16}{9}.$$

Ответ: 128

4.3. Задачи по информатике

Задача 4.3.1. (1 балл)

Юному Тимуру нравится рисовать четырехугольники с заданными длинами сторон. Длины для сторон он придумывает сам, и после этого начинает рисовать. Недавно на уроке математики он узнал, что не всегда из придуманных сторон можно получить четырехугольник. И теперь, чтобы не терять времени, Тимур хочет быть уверен, что по его задуманным длинам можно построить четырехугольник. За помощью в этом вопросе он обращается к вам.

Тимур даст вам длины четырех сторон. Необходимо определить, возможно ли построит из заданных сторон четырехугольник.

Формат входных данных

В единственной строке входных данных находится четыре целых положительных числа, разделенных пробелом. Каждое число не превосходит 10000.

Формат выходных данных

Если из заданных длин сторон можно построить четырехугольник, то необходимо вывести *Yes*, в ином случае — *No*.

Примеры

Пример №1

Стандартный ввод
2 1 3 2
Стандартный вывод
Yes

Пример №2

Стандартный ввод
4 10 3 2
Стандартный вывод
No

Решение

Необходимое и достаточное условие для построение четырехугольника с заданными сторонами – длина максимальной стороны должна быть строго меньше суммы трех остальных. Например, если a - максимальная сторона, то условие выглядит, как $a < b + c + d$. (b, c, d – остальные стороны четырехугольника).

Пример программы

Ниже представлено решение на языке C++

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

using namespace std;

int main() {
        vector <int> lines(4);
        for (int i = 0; i < 4; i++) {
                cin >> lines[i];
        }
        sort(lines.begin(), lines.end());
        if (lines[0] + lines[1] + lines[2] > lines[3]) {
                cout << "Yes\n";
        } else {
                cout << "No\n";
        }
}
```

Задача 4.3.2. (2 балла)

Из переписки двух друзей:

- Вася, привет! Не мог бы ты мне помочь?
- Привет, Настя! Да, конечно.
- Я готовлюсь к одной олимпиаде по программированию и решила немного подготовиться. Не мог бы ты дать мне какую-нибудь задачку?
- Без проблем! Вот условия: дана строка S_0. Каждая следующая строка получается добавлением в конец к предыдущей строке самого часто встречающего символа этой строки. Надо найти строку S_k. Ответов может быть несколько, поэтому надо вывести лексикографически минимальную строку из всех возможных.
- Хорошо, спасибо большое!

Помогите Насте решить задачу, которую ей дал Вася.

Формат входных данных

В первой строке заданы целые положительные числа n и k ($1 \leqslant n \leqslant 1000$, $1 \leqslant k \leqslant 1000$).

На следующей строке задана строка S_0, состоящая из n строчных букв латинского алфавита.

Формат выходных данных

Выведите лексикографически минимальную строку S_k.

Комментарии

Одна строка лексикографически меньше другой, если существует такая позиция m, что символ первой строки на позиции m меньше символа второй строки на той же позиции, а первые $m - 1$ символы двух строк совпадают.

Примеры

Пример №1

Стандартный ввод
`5 1` `abcdc`
Стандартный вывод
`abcdcc`

Пример №2

Стандартный ввод
`4 2` `baab`
Стандартный вывод
`baabaa`

Решение

Очевидно, что если мы в конец строки добавим самый часто встречающийся символ в ней, то и в следующих строках он также будет самым часто встречающимся. Поэтому ответом является исходная строка S_0, к которой в конец k раз приписан самый часто встречающийся символ в ней. Чтобы получить лексикографически минимальную строку в ответе, нужно среди самых часто встречающихся символов в S_0 выбрать тот символ, который раньше всего идёт в алфавите.

Пример программы

Ниже представлено решение на языке Python3

```
A = input().split();
n = int(A[0]);
k = int(A[1]);

s = input();

cnt = [0] * 26;

#Считаем сколько каждая буква встречается в строке.
for i in range(n) :
    cnt[ord(s[i]) - 97] = cnt[ord(s[i]) - 97] + 1;

mx = 0;

#Выбираем самую часто встречающуюся с минимальным индексом.
for i in range(26) :
    if (cnt[i] > cnt[mx]) :
        mx = i;

#Выводим строку с добавлением этой буквы k раз.
for i in range(k) :
    s = s + chr(97 + mx);

print(s);
```

Задача 4.3.3. (3 балла)

Фермер по имени Артем сегодня решил озеленить свой сад. Для удобства представим, что сад Артема представляет собой декартову плоскость, где на целых координатах он посадил розы.

Чтобы цветок расцвел необходимо его поливать, поэтому он поставил две поливалки, которые поливают все розы в определенном радиусе.

После оказалось, что некоторые розы поливаются с обоих устройств. Артему стало интересно, сколько таких роз, но так как поле очень большое, то он попросил вас помочь ему.

Формат входных данных

Первая строка содержит три целых целых числа $x1$, $y1$, $r1$ ($-10^6 \leqslant x1, y1 \leqslant 10^6, 0 \leqslant r1 \leqslant 10^6$) — координаты центра первой поливалки и её радиус действия.

Во второй в аналогичном формате координаты и радиус второй поливалки.

Формат выходных данных

Выведите количество роз, которые поливаются с двух поливалок.

Примеры

Пример №1

Стандартный ввод
0 0 5 2 2 3
Стандартный вывод
26

Решение

Переберем одну из координат. Найдем целые координаты, которые принадлежат первой окружности на координате, которую мы перебираем. Обозначим их за l_1, r_1. Аналогично сделаем для второй окружности и обозначим за l_2, r_2. Прибавим к ответу количество целых координат в пересечении этих отрезков l_1, r_1 и l_2, r_2.

Чтобы найти координаты можно воспользоваться бинарным поиском или же решить уравнение окружности, где будет одна неизвестная, однако нужно будет хорошо округлить до целых координат. От этого и будет зависеть сложность вашего решения. O(N) — для уравнение и O(NlogN) — для бинпоиска, где N — разность между максимальной и минимальной координатой.

Пример программы

Ниже представлено решение на языке Python3

```
from math import sqrt



x1, y1, r1 = map(int, input().split())
x2, y2, r2 = map(int, input().split())
ans = 0

#делаем полный перебор по x первой окружности
for x in range(x1-r1, x1+r1+1):
    #Если x за пределами второй окружности
    if x > x2 + r2 or x < x2 - r2:
        continue

    #Находим нижний y такой, что (x,y) лежат в первой окр.
    y1l = int(y1 - sqrt(r1 ** 2 - (x1 - x) ** 2)) - 10
    while ((y1l - y1) ** 2 + (x1 - x) ** 2) > r1 ** 2:
        y1l += 1

```

```
    #Находим верхний y такой, что (x,y) лежат в первой окр.
    y1h = int(y1 + sqrt(r1 ** 2 - (x1 - x) ** 2)) + 10
    while ((y1h - y1) ** 2 + (x1 - x) ** 2) > r1 ** 2:
        y1h -= 1

    #Находим нижний y такой, что (x,y) лежат во второй окр.
    y2l = int(y2 - sqrt(r2 ** 2 - (x2 - x) ** 2)) - 10
    while ((y2l - y2) ** 2 + (x2 - x) ** 2) > r2 ** 2:
        y2l += 1

    #Находим верхний y такой, что (x,y) лежат во второй окр.
    y2h = int(y2 + sqrt(r2 ** 2 - (x2 - x) ** 2)) + 10
    while ((y2h - y2) ** 2 + (x2 - x) ** 2) > r2 ** 2:
        y2h -= 1

    #Проверяем в пересечении ли они
    if not (y1h < y2l or y2h < y1l):
        #Добавляем в ответ точки
        ans += abs(max(y1l, y2l) - min(y1h, y2h)) + 1

print(ans)
```

Задача 4.3.4. (4 балла)

Гена очень послушный и умный маленький мальчик. Как и все дети он любит играть в кубики, но часто ему в голову приходят сложные математические формулы или непростые задачи. Вот и в этот раз, когда он раскладывал кубики, то придумал очень интересную задачу.

Он начал выкладывать кубики в ряд длинной n. Причем высота на i-ом месте ряда составляет a_i кубиков. Тут он и задался вопросом: какое минимальное количество кубиков надо переложить, чтобы все высоты стали равны. Кубики можно перекладывать только на соседние места, то есть, если мы взяли кубик с ряда номер j, то можно положить его на ряд номер $j+1$ или $j-1$, если такие существуют.

Формат входных данных

Первая строка содержит одно целое число n ($1 \leqslant n \leqslant 10^5$) — количетсво рядов из кубиков, которые выложил Гена.

Во второй строке дано n чисел $a_i (0 \leqslant a_i \leqslant 10^6)$ — количество кубиков в i-м номере ряда.

Формат выходных данных

Выведите минимальное количество перекладываний, которое необходимо сделать Гене, чтобы все высоты стали равны, если Гена не сможет выстроить такой ряд из кубиков, выведите -1.

Примеры

Пример №1

Стандартный ввод
4 2 1 2 3

Стандартный вывод
2

Пример №2

Стандартный ввод
5 2 3 4 5 6

Стандартный вывод
10

Решение

Посчитаем количество кубиков в каждом ряду: $h = \frac{1}{n} \cdot \sum_{i=1}^{n} a_i$. Пусть $df = 0$ — количество лишних кубиков на префиксе(отрицательное число, если кубиков не хватает). Теперь рассматриваем каждый ряд от 1 до n, $df = df + (a_i - h)$ — новая разница кубиков. В каждой итерации к ответу прибавляется $|df|$ — если у нас переизбыток кубиков, в любом случае они перейдут дальше, иначе кубики с конца должны будут перейти в начало.

Пример программы

Ниже представлено решение на языке C++

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

using namespace std;

int main() {
    int n;
    long long sum = 0;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; i++) {
        cin >> a[i];
        sum += a[i];
    }
    //Если кубики не делятся на кучки поровну
    if (sum % n != 0) {
        cout << -1 << '\n';
        return 0;
    }
    long long k = sum / n;
    long long ans = 0;
    int j = 0;
    for (int i = 0; i < n; i++) {
        if (a[i] >= k) {
            continue;
        }
```

```
27          while (a[i] != k) {
28              //Ищем первую кучку, где не хватает.
29              while (j < n && a[j] <= k) {
30                  j++;
31              }
32              //Перекладываем или весь избыток в i или то, что нужно, чтобы дополнить j
33              int t = min(k - a[i], a[j] - k);
34              a[j] -= t;
35              a[i] += t;
36              ans += t * abs(i - j);
37          }
38      }
39      cout << ans << '\n';
40  }
```

Задача 4.3.5. (5 баллов)

В крупной компании по созданию программного обеспечения находится n серверов. Они соединяются сетью с помощью m проводов. Провод соединяет два сервера между собой.

Сервера A и B находятся в одной локальной сети, если сигнал от сервера A может по рабочим проводам дойти до сервера B, возможно проходя при этом через промежуточные сервера. Если сервер может соединиться только с собой, то считается, что он сам по себе представляет локальную сеть.

В датацентре компании появились грызуны, которые начали перегрызать провода. Пока ваш напарник поехал за отпугивателями грызунов, вам поручили посчитать полученный ущерб компании. Вам нужно ответить, сколько всего локальных сетей в компании возникало после выведения каждого провода из строя.

Формат входных данных

В первой строке вводится целое число n ($2 \leqslant n \leqslant 3 \cdot 10^5$) - количество серверов в компании.

Во второй строке вводится целое число m ($1 \leqslant m \leqslant 3 \cdot 10^5$) - количество проводов.

В следующих m строках вводятся пары различных чисел a, b ($1 \leqslant a, b \leqslant n$) номера серверов, которые соединяет i-ый провод.

В следующей строке вводится число q ($1 \leqslant q \leqslant m$) количество оборванных проводов.

В следующей строке вводится q различных чисел – номера оборванных кабелей. Все номера различны и идут в хронологическом порядке.

Формат выходных данных

Выведите q чисел, количество различных локальных сетей после выведения из строя следующего провода.

Комментарии

Первый пример: после удаления первого провода все компьютеры все еще находятся в одной сети. После удаления второго провода, сеть разбивается на две части: компьютеры 1,3 и компьютер 2.

Примеры

Пример №1

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

Пример №2

Стандартный ввод
4 3 1 2 1 4 4 2 1 3
Стандартный вывод
2

Решение

Давайте решать обратную задачу: провода не обрывают, а чинят(идем по запросам в обратном порядке). Так как запросов порядка 10^5, надо уметь быстро объединять две локальные сети. Для этого можно использовать структуру данных система непересекающихся множеств, которая позволяет объединить два множество за время $O(\log n)$. Пусть изначально ни один компьютер не соединен с другим, то есть всего n локальных сетей. Процесс соединения двух компьютеров: если два компьютера лежат в разных локальных сетях(разные множества), то количество сетей уменьшится и мы объединим два множества, иначе оно останется таким же.

Возьмем все целые провода(которые не подвергались хакерской атаке) и объединим ими наши компьютеры. Количество локальных сетей после объединения будет равно ответу после последней хакерской атаки. Далее идем в обратном порядке по атакам и объединяем сети.

Пример программы

Ниже представлено решение на языке C++

```
1 #include <bits/stdc++.h>
2
```

```
using namespace std;
typedef  long long ll;
typedef unsigned long long ull;
int inf_int=1e8;
ll inf_ll=1e18;
typedef pair<int,int> pii;
typedef pair<ll,ll> pll;
const double pi=3.1415926535898;

const int MAXN=3e5+10;

int parent[MAXN];
int rank[MAXN];

//Находим "главный" сервер в сети.
int find_parent(int v) {
    if(v==parent[v]) {
        return v;
    }
    return parent[v]=find_parent(parent[v]);
}

bool union_set(int a,int b) {
    //Найдем "главный" сервер в сети, где a и где b
    a=find_parent(a);
    b=find_parent(b);
    //Если он один и тот же, то сеть объединять не надо
    if(a!=b) {
        //Иначе сливаем в одну суть, "главным тот, у кого rank выше
        if(rank[a]<rank[b])
            swap(a,b);
        else if(rank[a]==rank[b])
            ++rank[a];
        parent[b]=a;
        return true;
    }
    return false;
}
int x[MAXN],y[MAXN];
char used[MAXN];
int a[MAXN];
int ans[MAXN];
void solve()
{
    int n,m,q;
    scanf("%d",&n);
    scanf("%d",&m);
    for(int i=1;i<=m;++i) {
        scanf("%d %d",&x[i],&y[i]);
    }
    scanf("%d",&q);
    for(int i=1;i<=q;++i) {
        scanf("%d",&a[i]);
        used[a[i]]=1;
    }

    //Изначально каждый в своей сети
    for(int i=1;i<=n;++i)
        parent[i]=i;

```

```
63      int cur=n; //Изначально n сетей
64      //Добавляем провода, которые не быыли атакованы и меняем кол-во компонент связности
65      for(int i=1;i<=m;++i) {
66          if(!used[i]) {
67              cur-=union_set(x[i],y[i]);
68          }
69      }
70      //Теперь в обратно порядке добавляем провода, которые были атакованы
71      for(int i=q;i>=1;--i) {
72          ans[i]=cur;
73          cur-=union_set(x[a[i]],y[a[i]]);
74      }
75      for(int i=1;i<=q;++i) {
76          printf("%d ",ans[i]);
77      }
78
79
80  }
81
82  int main()
83  {
84          int t=1;
85          while(t--)
86             solve();
87          return 0;
88  }
```

Задачи для подготовки к Олимпиаде НТИ 2018/19

В данном разделе предложены задачи по математике и информатике, которые полезно было бы решить при подготовке к олимпиаде. Для некоторых задач приведены две формулировки: на языке машинного обучения и в привычных школьных терминах.

Предложенные ниже задачи по математике и информатике непосредственно связаны с методами машинного обучения и анализа данных.

При решении задачи 1 по математике как 9го, так и 10-11 класса получаются различные оценки количества разбиений множества прецедентов на обучение и контроль. Умение оценивать количество разбиений помогает при выборе стратегии скользящего контроля — общепринятой оценки обобщающей способности.

Задачи 2 и 3 по математике 9го и 10-11 классов, а также задача 4 по информатике посвящены обсуждению линейных классификаторов. Линейные классификаторы являются одними из самых используемых алгоритмов, потому важно понимать их не только на идейном, но и на формульном уровне. Предложенные задачи позволяют познакомиться с линейными классификаторами более внимательно.

В задачах 4 по математике 9го и 10-11 классов, а также задачах 1,3 и 5 по информатике обсуждаются тонкости применения и программной реализации другой основополагающей идеи машинного обучения — относить объекты к тому же классу, к которому относятся наиболее похожие на них объекты из множества прецидентов.

Задачи 2 и 5 по информатике обращают внимание на тонкости работы с размерностью. В частности, на возможность понижения размерности и на «проклятие размерности».

5.1. Задачи по математике (9 класс)

Задача 5.1.1. (2 балла)

Посчитайте количество способов разделить обучающую выборку длины L на обучающие и контрольные подвыборки. Как изменится результат, если потребовать, чтобы контрольная выборка была фиксированной длины k? (*Дано L различных шариков. Сколькими различными способами можно отложить часть из них в ящик? Как изменится результат, если в ящик необходимо отложить ровно k шариков?*)

Решение

В первом случае каждый объект независимо может быть отнесен как в обучающую так и контрольную подвыборки. Следовательно, количество вариантов разбиения равно 2^L. Если же контрольная выборка имеет длину k, поскольку все объекты различны, то определению количество способов такого выбора равно C_L^k.

Ответ: 2^L, C_L^k

Задача 5.1.2. (3 балла)

На плоскости дано n точек. Всегда ли можно провести прямую так, что она отделит ровно одну точку от остальных?

Решение

Точек конечное количество. Следовательно, можно построить выпуклый многоугольник, содержащий все заданные точки такой, что все его вершины будут точками из заданного множества. Рассмотрим вершину выпуклого многоугольника. проведем через нее прямую такую, что многоугольник будет лежать целиком в одной полуплоскости. Проведем прямую, параллельную данной таким образом, чтобы отделить одну единственную точку - вершину. Такое всегда возможно.

Данная задача составлена по мотивам линейного классификатора.

Ответ: Всегда возможно.

Задача 5.1.3. (3 балла)

Рассмотрим линейный классификатор в пространстве $\mathbb{R}^3$. Он представляет собой плоскость. Запишите уравнение плоскости. Сколько параметров это уравнение содержит? Однозначно ли оно определено? Можно ли считать свободный член всегда равным 1?

Решение

В $\mathbb{R}^2$ линейный классификатор это прямая, она имеет вид $a_2x_2 + a_1x_1 + a_0 = 0$. В $\mathbb{R}^3$ это плоскость, она имеет вид $a_3x_3 + a_2x_2 + a_1x_1 + a_0 = 0$. Мы можем разделить или умножить все уравнение на любое ненулевое число, однако если $a_0 = 0$, то такой операцией невозможно получить уравнение со свободным членом, равным 1. Уравнение содержит 4 параметра.

Ответ: $a_3x_3 + a_2x_2 + a_1x_1 + a_0 = 0$, a_0 не всегда можно считать равным 0.

Задача 5.1.4. (3 балла)

Алгоритм k ближайших соседей относит объект к тому классу, к которому относится большинство из k его ближайших соседей. Класса всего два. Неопределенность считается ошибкой. Может ли алгоритм 3 ближайших соседей ошибаться чаще, чем алгоритм 4 ближайших соседей?

Решение

Рассмотрим объект, который классификатор 4 ближайших соседей отметил ошибкой из-за неопредленности. Алгоритм 3 ближайших соседей мог на нем как ошибиться, так и классифицировать его верно. Рассмотрим объект, который классификатор 4 соседей отнес к первому классу. Среди первых 4 его соседей не менее 3 относится к первому классу, а значит среди первых трех его соседей не менее 2 относится к первому классу. Следовательно, классификатор с $k = 3$ и $k = 4$ классифицирует его одинаково. Аналогично для объектов второго класса. Таким образом, все объекты, которые алгоритм с $k = 4$ классифицировал верно, верно классифицировал и алгоритм с $k = 3$. Таким образом, алгоритм 3 ближайших соседей не может ошибаться чаще, чем алгоритм 4 ближайших соседей.

Замечание: именно по этой причине алгоритм k ближайших соседей, как правило, используют с нечетным k.

Ответ: Такое невозможно.

Задача 5.1.5. (4 балла)

В $\mathbb{R}^n$ дан куб со стороной 1 и второй куб со стороной $0,99$, находящийся внутри первого. В кубе со стороной 1 случайным образом (распределение равномерное) выбирается точка. Оцените вероятность того, что точка будет лежать вне второго куба для $n = 1$, 3, 100.

Решение

Вероятность попасть в меньший куб равна отношению объемов. Таким образом, вероятность попасть в первый куб и не попасть во второй равна $1 - \left(\frac{0{,}99}{1}\right)$. Для $n = 1$ ответ 0,01, для $n = 3$ значение примерно равно 0,03. Для $n = 100$ вероятность примерно равна 0,63.

Ответ: примерно равны: 0,01; 0,03; 0,63.

5.2. Задачи по математике (10-11 класс)

Задача 5.2.1. (2 балла)

Посчитайте количество способов разделить обучающую выборку, состоящую из 200 объектов первого класса и 300 объектов второго класса на обучающие и контрольные подвыборки с сохранением соотношения между классами. Контрольная выборка имеет длину 100 (*Дано 200 белых и 300 черных различных шариков. Сколькими различными способами можно отложить 100 из них в ящик с сохранением соотношения?*)

Решение

Сохранение отношения между классами значит, что в контрольной подвыборке должно оказаться 40 объектов первого класса и 60 объектов второго класса. Все объекты считаются различными. Тогда количество способов сделать выбор равно $C_{200}^{40} C_{300}^{60}$.

Ответ: $C_{200}^{40} C_{300}^{60}$

Задача 5.2.2. (3 балла)

На плоскости дано n точек. Всегда ли можно провести прямую так, что она отделит ровно одну точку от остальных? Ровно k точек?

Решение

Проведем произвольную прямую α. Будем рассматривать проекции всех точек на нее. Если проекции каких-то двух точек совпали, значит прямая, проходящая через них, перпендикулярна данной. Проведем все возможные прямые, проходящие через каждые две точки из n. Таких прямых не более $0,5(n-1)n$, то есть конечное число. Следовательно, мы всегда можем так провести прямую α, чтобы проекции никаких двух точек не совпадали. Но тогда мы пожем отсчитать ровно k проекций на прямую слева направо и между k-й и $k+1$-й провести перпендикуляр. Полученная прямая отделит ровно k точек.

Данная задача составлена по мотивам линейного классификатора.

Ответ: Возможно всегда

Задача 5.2.3. (3 балла)

Рассмотрим линейный классификатор в пространстве $\mathbb{R}^n$. Он представляет собой гиперплоскость — обобщение плоскости на случай n-мерия. Запишите уравнение гиперплоскости. Сколько параметров это уравнение содержит? Однозначно ли оно определено? Можно ли считать свободный член всегда равным 1?

Решение

В $\mathbb{R}^2$ гиперплоскость это прямая, она имеет вид $a_2x_2 + a_1x_1 + a_0 = 0$. В $\mathbb{R}^3$ гиперплоскость это плоскость, она имеет вид $a_3x_3+a_2x_2+a_1x_1+a_0 = 0$. В $\mathbb{R}^3$ гиперплоскость соответственно имеет вид $a_nx_n+a_{n-1}x_{n-1}+\ldots+a_1x_1+a_0 = 0$. Во всех случаях мы можем разделить или умножить все уравнение на любое ненулевое число, однако если $a_0 = 0$, то такой операцией невозможно получить уравнение со свободным членом, равным 1. Уравнение содержит $n + 1$ параметр.

Ответ: $a_nx_n + a_{n-1}x_{n-1} + \ldots + a_1x_1 + a_0 = 0$, a_0 не всегда можно считать равным 0.

Задача 5.2.4. (3 балла)

Алгоритм k ближайших соседей относит объект к тому классу, к которому относится большинство из k его ближайших соседей. Класса всего два. Неопределенность считается ошибкой. Может ли алгоритм $2t - 1$ ближайших соседей ошибаться чаще, чем алгоритм $2t$ ближайших соседей?

Решение

Рассмотрим объект, который классификатор $2t$ ближайших соседей отметил ошибкой из-за неопредленности. Алгоритм $2t - 1$ ближайших соседей мог на нем как ошибиться, так и классифицировать его верно. Рассмотрим объект, который классификатор $2t$ соседей отнес к первому классу. Среди первых $2t$ его соседей не менее $t + 1$ относится к первому классу, а значит среди первых $2t - 1$ его соседей не менее t относится к первому классу. Следовательно, классификатор с $2t - 1$ и $2t$ классифицирует его одинаково. Аналогично для объектов второго класса. Таким образом, все объекты, которые алгоритм с $k = 2t$ классифицировал верно, верно классифицировал и алгоритм с $k = 2t$. Таким образом, алгоритм $2t - 1$ ближайших соседей не может ошибаться чаще, чем алгоритм $2t$ ближайших соседей.

Замечание: именно по этой причине алгоритм k ближайших соседей, как правило, используют с нечетным k.

Ответ: Такое невозможно.

Задача 5.2.5. (4 балла)

В $\mathbb{R}^n$ дан куб со стороной 1 и второй куб со стороной $0,99$, находящийся внутри первого. В кубе со стороной 1 случайным образом (распределение равномерное) выбирается точка. Задано небольшое число $\delta > 0$. Всегда ли можно подобрать такое n, что вероятность попасть во второй куб меньше δ.

Решение

Вероятность попасть в меньший куб равна отношению объемов. Таким образом, вероятность попасть во второй куб равна $\left(\frac{0{,}99}{1}\right)$. Предел отношения $\left(\frac{0{,}99}{1}\right)$ стремится к 0, при $n \to +\infty$. Таким образом для любого $\delta > 0$ всегда ли можно подобрать такое n, что вероятность попасть во второй куб меньше δ.

Замечание: Представим, что большой куб — это пространство, на котором задана функция. Мы измеряем ее значение в точках. Предположим, второй куб — это расположение некотрой особенности функции, которую мы хотим измерять. Оказывается, что с ростом n и при случайных измерениях требуется все больше и больше точек, чтобы хорошо описать искомую функцию. То есть, выражаясь на языке машинного обучения: даже если классы расположены в виде компактных сгустков точек, если признаков слишком много, нам потребуется очень-очень много данных, чтобы получить хорошее качество классификации. Данное явление называют проклятем размерности. Типичный метод борьбы с ним — отбор информативных признаков.

Ответ: Всегда возможно

5.3. Задачи по информатике

Задача 5.3.1. (3 балла)

1. Рассмотрим алгоритм ближайшего соседа. Пусть обучающая выборка содержит L объектов. Оцените число шагов, которые необходимо совершить алгоритму для классификации нового объекта x по обучающей выборке? (*Дано n точек в пространстве. Пусть задали некоторую новую точку x. Какое наименьшее число расстояний между точками пространства необходимо измерять и сколько сравнений этих расстояний произвести, чтобы найти ближайшую к x точку среди n.*)
2. Дано L объектов. Оцените количество шагов, которое необходимо сделать алгоритму для подсчета 3-го профиля компактности $P(3)$. (*Оцените количество шагов алгоритма, который находит для каждой из L точек ее 3-го соседа, если даны все попарные расстояния между точками.*)

Решение

1. Для поиска ближайшего соседа алгоритм должен измерять все расстояния от данной точки до остальных: n измерений. Затем найти из этих n минимальное, что делается также за $n-1$ шаг. В этом случае легко удается посчитать оптимальное количество шагов. Однако как правило, точны подсчет и не трк-буется: интересует асимптотическая оценка. Пример асимптотической оценки приведем для второго пункта этой задачи.
2. Для того, чтобы найти 3го соседа каждого объекта необходимо рассмотреть все расстояния. Их $\frac{n(n-1)}{2}$, то есть $O(n^2)$. Это значит, что задача не может быть решена менее чем за $O(n^2)$ или, как еще говорят, квадратичное число шагов. С другой стороны для каждого объекта можно найти среди остальных

3го соседа за время не превосходящее $3n$, а всего объектов n. Следовательно, задача может быть решена за $3n^2$, то есть $O(n^2)$ шагов. То есть задача может быть решена за квадратичное время и не может быть решена менее чем за квадратичное время.

Ответ: Количество шагов первого алгоритма можно оценить $2n - 1$ шагом измерения и сравнения. Количество шагов второго алгоритма оценивается асимптотически $O(n^2)$.

Задача 5.3.2. (3 балла)

Дано $t \geqslant 3$ точек на плоскости. Напишите программу, определяющую, лежат ли они на одной прямой.

Формат входных данных

В каждой строке два целых числа — координаты точки.

Формат выходных данных

Выводит одно число: 1 если лежат на одной прямой, 0 если не лежат.

Примеры

Пример №1

Стандартный ввод
1 2 2 4 4 8
Стандартный вывод
1

Решение

Данную задачу возможно решать различными методами. Например, можно зафиксировать две точки $A(x_1, y_1)$ и $B(x_2, y_2)$. Для каждой следующей точки $C(x_3, y_3)$ определять, колинеарен ли вектор AB вектору BC при помощи скалярного произведения. Точки лежат на одной прямой тогда и только тогда, когда вектора коллинеарны, то есть когда

$$(x1 - x2) * (y3 - y2) = (x3 - x2) * (y1 - y2)$$

Замечание: Признаковое пространство в задачах машинного обучения бывает большой размерности. Размерность равна количеству признаков. Большая размерность ведет к «проклятию размерности» и переобучению. Важным вопросом является эффективная размерность — минимальная размерность подпространства, в которую можно поместить все наши точки с сохранением расстояний. Например, если точки на плоскости на самом деле лежат на одной прямой, то эффективная размерность пространства равна размерности прямой, то есть 1.

Пример программы

Ниже представлено решение на языке Python3

```
t = int(input())
coordinates = []
for i in range(t):
    x, y = map(int, input().split())
    coordinates.append([x,y])


epsilon = 1e-20
flag = 1

x1 = coordinates[0][0]
y1 = coordinates[0][1]

x2 = coordinates[1][0]
y2 = coordinates[1][1]

for i in range(2, t):
    x3 = coordinates[i][0]
    y3 = coordinates[i][1]
    if ((x1-x2)*(y3-y2) - (x3-x2)*(y1-y2))**2 > epsilon:
        flag = 0
        break
print(flag)
```

Задача 5.3.3. (3 балла)

Алгоритм k ближайших соседей относит объект к тому классу, к которому относится большинство из k его ближайших соседей. Неопределенность считается ошибкой. Напишите алгоритм k ближайших соседей для точек на плоскости, k — нечетное. Считаем, что класса всего два: 0 и 1.

Формат входных данных

В первой строке целое число — количество точек, для которых дано значение класса. Во второй строке целое число — количество соседей. В третьей строке два целых числа — координаты точки, класс которой требуется определить.

В каждой следующей строке три целых числа: первые две — координаты объекта, третье число — класс.

Формат выходных данных

Выводит одно число: имя класса, к которому алгоритм отнес объект. Число 0 или 1 соответственно.

Примеры

Пример №1

Стандартный ввод
4 1 1 1 1 2 0 2 5 0 4 8 1 1 5 0

Стандартный вывод
0

Решение

Для точки, которую требуется классифицировать, измеряются все расстояния до других точек. Выбираются k точек с наименьшими расстояниями до них. Выводится тот класс, к которому относится большинство из выбранных k точек.

Пример программы

Ниже представлено решение на языке Python3

```
n = int(input())
k = int(input())
x0, y0 = map(int, input().split())

coordinates = []
class_labels = []
for i in range(n):
    x, y, point_class = map(int, input().split())
    class_labels.append(point_class)
    coordinates.append([x,y])

def distance(x1, x2):
    return ((x1[0]-x2[0])**2 + (x1[1]-x2[1])**2)**(0.5)

nearest_neighbors = []
neighbors_distances = []
neighbors_classes = []

for i in range(k):
    nearest_neighbors.append(coordinates[j])
    neighbors_distances.append(distance([x0, y0], coordinates[i]))
    neighbors_classes.append(class_labels[i])

'''
 Осуществляем поиск k ближайших соседей
'''

for i in range(k, n):
    for j in range(k):
        if distance([x0, y0], coordinates[i]) < neighbors_distances[j]:
            neighbors_distances[j] = distance([x0, y0], coordinates[i])
            nearest_neighbors[j] = coordinates[i]
            neighbors_classes[j] = class_labels[i]

```

```
35 point_class = 0
36 if neighbors_classes.count(1) > neighbors_classes.count(0):
37     point_class = 1
38 print(point_class)
```

Задача 5.3.4. (3 балла)

Объекты, как известно описываются признаками. Бывает удобно представлять их себе как точки в некотором многомерном пространстве. Тогда линейный классификатор в этом пространстве будет выглядеть как обобщение плоскости — гиперплоскость.

В пространстве $\mathbb{R}^n$ гипреплоскость задана своим уравнением. Пусть даны координаты двух точек. Определить, лежат ли они по одну или по разные стороны гиперплоскости.

Формат входных данных

В первой строке задано целое число n

Во второй строке заданы коэффициенты уравнения плоскости: $n+1$ целое число.

Во третей и четвертой строках дано по n целых чисел — координат точек.

Формат выходных данных

Одно число: 1, если точки лежат по одну сторону, 0 если с разных сторон.

Примеры

Пример №1

Стандартный ввод
5 3 4 5 6 7 1 2 3 4 1 0 2 8 1 3 6
Стандартный вывод
0

Решение

По уравнению гиперплоскости $a_n z_n + a_{n-1} z_{n-1} + \ldots + a_1 z_1 + a_0 = 0$ и координатам точки $(x_1, x_2, \ldots, x_n)$ можно определить положение точки относительно плоскости следующим образом: определить знак выражения $a_n x_n + a_{n-1} x_{n-1} + \ldots + a_1 x_1 + a_0$.

Точки лежат по одну сторону от плоскости тогда и только тогда, когда знаки совпадают.

Замечание: Гиперплоскость соответствует линейному классификатору. Тогда вопрос задачи можно было бы сформулировать так: к одному или разным классам отнесет заданный линейный классификатор данные две точки.

Пример программы

Ниже представлено решение на языке Python3

```
n = int(input())
coefficients = list(map(int, input().split()))
x = list(map(int, input().split()))
y = list(map(int, input().split()))

scalar_product1 = coefficients[-1]
scalar_product2 = coefficients[-1]
'''
Определяем, как расположены точки: выше или ниже плоскости.
'''
for i in range(n):
    scalar_product1 += x[i] * coefficients[i]
    scalar_product2 += y[i] * coefficients[i]
'''
Определяем, лежат ли они с одной и той же стороны от плоскости.
'''
if scalar_product1 * scalar_product2 > 0:
    print(1)
else:
    print(0)
```

Задача 5.3.5. (3 балла)

Профиль компактности $P(m)$ — доля объектов, для которых их класс и класс их m-го соседа отличаются. Напишите программу, вычисляющую первый профиль компактности $P(1)$. Считаем, что класса всего два: 0 и 1.

Формат входных данных

В первой строке целое число — количество точек, для которых дано значение класса.

В каждой следующей строке три целых числа: первые две — координаты объекта, третье число — класс.

Формат выходных данных

Значение $P(1)$.

Примеры

Пример №1

Стандартный ввод
4 0 0 0 0 1 0 4 8 1 4 7 1

Стандартный вывод
0

Решение

Предложенное здаесь решение не является оптимальным. Существует множество способов ускорить рассчеты профиля компактности. Возможные оптимизации мы оставляем на фантазию читателя. Мы предполагаем, что приведенный ниже код поможет Вам проверить, правильный ли результат выдает Ваш алгоритм.

Замечание: Профиль компактности позволяет узнать, выполняется ли гипотеза компактности для данного множества объектов с заданной функцией расстояний. Во всех задачах мы использовали стандартную Евклидову функцию расстояний. С другими вариантами расстояний можно будет познакомиться в задачах финального тура.

Пример программы

Ниже представлено решение на языке Python3

```python
n = int(input())
coordinates = []
labels = []
for i in range(n):
    x, y, label = map(int, input().split())
    labels.append(label)
    coordinates.append([x,y])

def distance(x1, x2):
    return ((x1[0]-x2[0])**2 + (x1[1]-x2[1])**2)**(0.5)

n_different_neighbors = 0

for i in range(n):
    x = coordinates[i][0]
    y = coordinates[i][1]
    neighbor_label = labels[i-1]
    min_distance = distance([x, y], coordinates[i-1])
    for j in range(n):
        if (i != j) and (distance([x, y], coordinates[j]) < min_distance):
            min_distance = distance([x, y], coordinates[j])
            nearest_neighbor_label = labels[j]
    if nearest_neighbor_label != labels[i]:
        n_different_neighbors += 1

print(n_different_neighbors/n)
```