Информационная безопасность

2022/23 учебный год

Заключительный этап

Предметный тур

Информатика. 8–11 класс

Задача VI.1.1.1. Игра на любителя (20 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1 секунда.

Ограничение по памяти: 256 мегабайт.

Условие

Вася любит играть с числами. Вот и в этот раз Вася придумал следующую игру:

Для начала Вася выбирает целевое число x.

Игра начинается с числа $s = 0$. На каждом шаге Вася может выполнить одну из двух операций:

1. Прибавить 1 к числу s.
2. Умножить число s на 2.

Игра заканчивается, когда Вася получает $s = x$.

У Васи есть любимое число p. Теперь Вася решил поиграть с этим числом в свою новую игру. Но есть проблема — у Васи осталась только запись числа p в системе счисления $b = 2^k$.

Помогите Васе найти **минимальное количество** вышеописанных операций, выполнив которые, он сможет получить число p из числа $s = 0$.

Формат входных данных

В первой строке содержатся два целых числа n и k ($1 \leqslant n \leqslant 10^5, 1 \leqslant k \leqslant 4$) — количество цифр в числе p и показатель степени в системе счисления $b = 2^k$.

Во второй строке задаётся строка, состоящая из n символов $P_1, P_2, \ldots, P_n$ — запись числа p в системе счисления b. Гарантируется, что P_i является корректной цифрой в записи системы счисления b.

Формат выходных данных

Выведите целое число — **минимальное количество** операций, чтобы получить число p из числа $s = 0$.

Критерии оценивания

Группа	Баллы	Дополнительные ограничения	Необходимые группы
0	-	Тесты из условия	-
1	5	$1 \leqslant n \leqslant 10$, $k = 1$	0
2	15	$1 \leqslant n \leqslant 60$, $k = 1$	1
3	20	$k = 1$	2
4	20	$k = 2$	0
5	20	$k = 3$	0
6	20	$k = 4$	0

Примеры

Пример №1

Стандартный ввод
`5 1` `10011`
Стандартный вывод
`7`

Пример №2

Стандартный ввод
`2 4` `C9`
Стандартный вывод
`11`

Пояснения к примеру

Пояснение к **первому** примеру:

- Любимое число Васи $p = 10011_2 = 19_{10}$.
- Последовательность из примера задаёт следующие преобразования: $0 \to 1 \to 2 \to 4 \to 8 \to 9 \to 18 \to 19$.
- Итого — 7 операций.

Пояснение ко **второму** примеру:

- Любимое число Васи $p = C9_{16} = 201_{10}$.
- Последовательность из примера задаёт следующие преобразования: $0 \to 1 \to 2 \to 3 \to 6 \to 12 \to 24 \to 25 \to 50 \to 100 \to 200 \to 201$.
- Итого — 11 операций.

Отметим, что установлено следующее соответствие чисел 10-й системы счисления и цифр 16-й:

- $10_{10} = A_{16}$.
- $11_{10} = B_{16}$.
- $12_{10} = C_{16}$.

- $13_{10} = D_{16}$.
- $14_{10} = E_{16}$.
- $15_{10} = F_{16}$.

Решение

Сперва разберёмся со случаем $k = 1$, тогда p — двоичная строка. Заметим, что при умножении числа s на 2 все цифры в двоичном представлении s сдвигаются на один разряд влево.

Пусть имеется двоичная строка a. Возможны три перехода:

1. Чтобы перейти от строки «0» к строке «1», нужно выполнить операцию 1.
2. Чтобы перейти от строки a к строке a + «0», нужно выполнить операцию 2.
3. Чтобы перейти от строки a к строке a + «1», нужно выполнить операцию 21.

Обработаем строку p справа налево. Находясь в индексе $i \neq 1$, хотим перейти от строки $p_1 \dots p_{i-1}$ к строке $p_1 \dots p_i$. Запишем необходимые для этого перехода операции в начало строки ans, которая будет являться ответом. В случае $i = 1$ символ p_i обязан быть равен $1 \rightarrow$ припишем в начало строки ans операцию 1.

Данный процесс — ничто иное, как быстрое возведение в степень, только работает мы не с числом, а двоичным представлением числа. Можно показать, что работая с двоичным представлением числа, мы совершаем наименьшее количество операций.

Для $k = 2, 3, 4$ достаточно перевести p в двоичную систему счисления и применить вышеописанное решение. Для перевода из 2^k-й системы счисления достаточно знать, что любой символ 2^k-й системы счисления «развернётся» в двоичную строку длины k из нулей и единиц, равную двоичному представлению десятичного числа, которое соответствует этому символу. Таким образом нужно заменить каждый символ строки p.

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <cmath>
#include <iomanip>
#include <map>
#include <set>
#include <queue>
#include <string>
#include <numeric>
#include <bitset>
#include <algorithm>
using namespace std;

#define sp " "
#define enl '\n'
#define int int64_t
#define all(x) (x).begin(), (x).end()
#define fast(); ios_base::sync_with_stdio(0);cin.tie(NULL);
```

```
20
21  #define OR(a, b) ((a) | (b))
22  #define AND(a, b) ((a) & (b))
23  #define XOR(a, b) ((a) ^ (b))
24
25  #define MOD_SUM(m, x, y) (((x) + (y)) % (m))
26  #define MOD_DIF(m, x, y) ((((x) - (y)) % (m) + (m)) % (m))
27
28  map<char, string> x16 = {{'0', "0000"}, {'1', "0001"}, {'2', "0010"}, {'3',
    ↪  "0011"},
29                           {'4', "0100"}, {'5', "0101"}, {'6', "0110"}, {'7',
                             ↪  "0111"},
30                           {'8', "1000"}, {'9', "1001"}, {'A', "1010"}, {'B',
                             ↪  "1011"},
31                           {'C', "1100"}, {'D', "1101"}, {'E', "1110"}, {'F',
                             ↪  "1111"}};
32
33  map<char, string> x8 = {{'0', "000"}, {'1', "001"}, {'2', "010"}, {'3', "011"},
34                          {'4', "100"}, {'5', "101"}, {'6', "110"}, {'7', "111"}};
35
36  map<char, string> x4 = {{'0', "00"}, {'1', "01"}, {'2', "10"}, {'3', "11"}};
37
38  int32_t main(){
39      fast();
40
41      int n, k;
42      string s;
43      cin >> n >> k >> s;
44
45      string t = "";
46      for(char c: s)
47          if(k == 1)
48              t += c;
49          else if(k == 2)
50              t += x4[c];
51          else if(k == 3)
52              t += x8[c];
53          else
54              t += x16[c];
55
56      int p = 0;
57      while(p < (int)t.size() && t[p] == '0')
58          ++p;
59      t = t.substr(p, (int)t.size());
60
61      int ans = 1;
62      for(int i = (int)t.size() - 1; i > 0; --i)
63          if(t[i] == '0')
64              ++ans;
65          else
66              ans += 2;
67      cout << ans;
68      return 0;
69  }
```

Задача VI.1.1.2. Оптимизация числа (40 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1 секунда.

Ограничение по памяти: 256 мегабайт.

Условие

Дано целое число x. К нему можно применять следующую операцию неограниченное количество раз:

- Поменять любые две **соседние** цифры a и b за $|a - b|$ секунд.

За какое наименьшее время удастся сделать число x максимальным?

Формат входных данных

В первой строке расположено единственное натуральное число x ($1 \leqslant x \leqslant 10^{300\,000} - 1$) — число, описанное в условии.

Формат выходных данных

Выведите единственное целое неотрицательное число t — минимальное количество секунд, которого хватит, чтобы посредством вышеупомянутой операции сделать число x максимальным.

Критерии оценивания

Группа	Баллы	Доп. ограничения	Необходимые группы
0	-	Тесты из условия	-
1	10	$1 \leqslant x \leqslant 10^3 - 1$	0
2	20	$10^3 \leqslant x \leqslant 10^9 - 1$	1
3	30	$10^9 \leqslant x \leqslant 10^{5000} - 1$	2
4	40	Без доп. ограничений	3

Примеры

Пример №1

Стандартный ввод
1459

Стандартный вывод
25

Пояснения к примеру

Приведём возможную последовательность обменов для числа из первого примера.

1. Совершим преобразование $1459 \Leftrightarrow 4159$ за $|1 - 4| = 3$ секунды.
2. Совершим преобразование $4159 \Leftrightarrow 4519$ за $|1 - 5| = 4$ секунды.

3. Совершим преобразование $4519 \Leftrightarrow 4591$ за $|1 - 9| = 8$ секунд.
4. Совершим преобразование $4591 \Leftrightarrow 4951$ за $|5 - 9| = 4$ секунды.
5. Совершим преобразование $4951 \Leftrightarrow 9451$ за $|4 - 9| = 5$ секунд.
6. Совершим преобразование $9451 \Leftrightarrow 9541$ за $|4 - 5| = 1$ секунду.

Итого, $3 + 4 + 8 + 4 + 5 + 1 = 25$ секунд.

Можно показать, что максимизировать число 1459 быстрее не выйдет.

Решение

Считаем число x и будем работать с ним, как со строкой. Заметим, что два символа x_i и x_j нужно будет поменять местами (наступит момент, когда они окажутся рядом) тогда и только тогда, когда $x_i < x_j$, а $i < j$.

Одно из возможных решений, работающих за $O(|x|^2)$ — отсортировать по убыванию строку x «*пузырьком*». При обмене символов x_i и x_{i+1} прибавим разницу между ними к ответу.

Для того чтобы решить задачу на полный балл, необходимо двигаться по строке слева направо. Обрабатывая цифру x_i, нужно знать, сколько цифр меньших x_i встречалось ранее. А эта информация будет храниться в массиве a размера 10, где a_k — количество цифр k, которые мы насчитали на префиксе $[1;\ i - 1]$.

Асимптотика решения — $O(|x|)$.

Пример программы-решения

Ниже представлено решение на языке C++.

```cpp
#include <iostream>
#include <iomanip>
#include <vector>
#include <cmath>
#include <set>
#include <map>
#include <bitset>
#include <random>
#include <algorithm>
using namespace std;

#define sp " "
#define int int64_t
#define all(x) (x).begin(), (x).end()

int32_t main(){
    string s;
    cin >> s;

    vector<int> a(10);
    int n = (int)s.size(), res = 0;
    for(int i = 0; i < n; ++i){
        int cur = (int)(s[i] - '0');
        for(int j = 0; j < cur; ++j)
            res += a[j] * (cur - j);
        ++a[cur];
    }
```

```
28      cout << res;
29      return 0;
30  }
```

Задача VI.1.1.3. Олимпиада по информационной безопасности (30 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 2 секунды.

Ограничение по памяти: 256 мегабайт.

Условие

В олимпиаде по информационной безопасности участвуют n команд. Команды были рассажены за **круглый стол** так, что команда с номером $t+1$ сидит рядом с командой t, а команда 1 сидит рядом с командой n.

Для проведения было подготовлено n компьютеров — по одному на команду. Про компьютер с номером i известна его производительность a_i.

Организаторы могут как угодно распределить компьютеры между командами и заключили, что эффективнее всего будет максимизировать величину S — сумму абсолютных разниц в производительности компьютеров **соседних** команд.

Формально, $S = \left(\sum_{t=1}^{n-1} |a_{p_t} - a_{p_{t+1}}|\right) + |a_{p_1} - a_{p_n}|$, где p — перестановка чисел от 1 до n такая, что команда t работает за компьютером p_t.

Во время тура на площадке случаются скачки напряжения. В результате одного скачка изменяется производительность **ровно одного** компьютера. Обратите внимание, что изменение производительности носит постоянный характер (до конца соревнования или до следующего скачка с заданным компьютером).

После каждого скачка организаторы хотят знать, какой максимальной величины S можно достичь, если заново распределить компьютеры между командами оптимальным образом.

Формат входных данных

В первой строке даны два целых числа n и q ($3 \leqslant n \leqslant 2 \cdot 10^5$, $1 \leqslant q \leqslant 2 \cdot 10^5$) — количество команд и скачков напряжения соответственно.

В следующей строке перечислены n целых чисел массива a ($1 \leqslant a_i \leqslant 10^9$) — начальные производительности компьютеров.

Следующие две строки описывают скачки напряжения.

В первой строке заданы q целых чисел ($1 \leqslant index_i \leqslant n$), где $index_i$ — номер компьютера, производительность которого изменилась после i-го скачка напряжения.

Во второй строке заданы q целых чисел ($1 \leqslant efficiency_i \leqslant 10^9$), где $efficiency_i$ — производительность установившаяся на компьютере с номером $index_i$ после i-го скачка напряжения.

Формат выходных данных

На отдельных строках выведите q целых неотрицательных чисел $S_1, S_2, \ldots, S_q$, где S_j — максимальная сумма разниц в производительности компьютеров **соседних** команд, которую можно достичь после j-го скачка напряжения.

Критерии оценивания

Группа	Баллы	Дополнительные ограничения	Необходимые группы
0	-	Тесты из условия	-
1	10	$3 \leqslant n \leqslant 10$, $q = 1$	0
2	10	$3 \leqslant n \leqslant 10^3$, $1 \leqslant q \leqslant 10$	1
3	25	$1 \leqslant q \leqslant 10$	2
4	25	$1 \leqslant a_i \leqslant 100$, $1 \leqslant x_j \leqslant 100$	0
5	30	Без дополнительных ограничений	3, 4

Примеры

Пример №1

Стандартный ввод
5 3 7 2 8 1 5 3 2 4 2 10 4
Стандартный вывод
18 28 22

Пример №2

Стандартный ввод
6 2 1 2 3 4 5 6 1 5 10 1
Стандартный вывод
24 28

Пояснения к примеру

Разберём первый пример.

1. Случился первый скачок напряжения:
   - $a =$ [7, 2, 2, 1, 5]: компьютер i имеет производительность a_i.
   - Пусть $p =$ [1, 4, 5, 3, 2]: команда t пользуется компьютером p_t.
   - Тогда $S_1 = |7 - 1| + |5 - 1| + |5 - 2| + |2 - 2| + |7 - 2| = 18$.

2. Случился второй скачок напряжения:
   - $a = [7, 10, 2, 1, 5]$.
   - Пусть $p = [2, 4, 1, 3, 5]$.
   - Тогда $S_2 = |10 - 1| + |1 - 7| + |7 - 2| + |2 - 5| + |5 - 10| = 28$.
3. Случился третий скачок напряжения:
   - $a = [7, 10, 2, 4, 5]$.
   - Пусть $p = [2, 3, 1, 4, 5]$.
   - Тогда $S_3 = |10 - 2| + |2 - 7| + |7 - 4| + |4 - 5| + |5 - 10| = 22$.

Можно показать, что более оптимальных вариантов распределить компьютеры между командами — выбрать другую перестановку p — после каждого из трёх скачков не существует.

Решение

Эта задача сводится к расстановке чисел массива a по кругу так, чтобы величина S была максимальна.

Пусть числа k и m стоят рядом в произвольной расстановке чисел по кругу, причём $k < m$. Раскрыв модуль в выражении $|k - m|$, получим $m - k$ — слагаемое m вошло в сумму со знаком «+», слагаемое k вошло в сумму со знаком «−».

Из этих соображений следует конструктивное решение: отсортируем массив a по возрастанию — теперь хотим, чтобы числа первой половины массива вошли в сумму со знаком «−», а числа второй половины со знаком «+». Этого можно добиться, чередуя числа из первой и второй половины при расстановке по кругу.

Таким образом можно решить первые 4 подзадачи — будем сортировать массив a после каждого запроса. Асимптотика решения $O(n \cdot \log_2(n) \cdot q)$

Чтобы решить задачу на полный балл, нужно научиться поддерживать две половины — «минимумы» и «максимумы». Будем поддерживать «минимумы» в множестве пар «производительность — индекс» p, а «максимумы» в множестве пар q. При скачке напряжения, повлекшем $a_i = x$, необходимо определить, в каком множестве хранится элемент $\{a_i, i\}$, удалить этот элемент из множества. Затем определить, в какое множество добавить обновлённый элемент $\{x, i\}$. При необходимости переместить крайний элемент одного из множеств в другое, чтобы размеры множеств были одинаковыми. Эти операции должны сопровождаться пересчётом суммы чисел в обоих множествах. Тогда ответом будет удвоенная разность суммы чисел из множества «максимумов» и суммы чисел из множества «минимумов».

Не стоит забывать, что при нечётном n медиана массива войдёт в ответ один раз со знаком «+» и один раз со знаком «−».

Асимптотика решения — $O(n \cdot \log_2(n) + q \cdot \log_2(n))$

Пример программы-решения

Ниже представлено решение на языке C++.

```
1  #include <iostream>
2  #include <iomanip>
3  #include <vector>
```

```cpp
#include <cmath>
#include <set>
#include <map>
#include <bitset>
#include <random>
#include <algorithm>
using namespace std;

#define sp " "
#define int int64_t
#define all(x) (x).begin(), (x).end()

int32_t main(){
    ios_base::sync_with_stdio(0);
    cin.tie(NULL);

    int n, q;
    cin >> n >> q;
    vector<int> a(n);
    for(int& i: a)
        cin >> i;

    vector<pair<int, int>> s(n);
    for(int i = 0; i < n; ++i){
        s[i].first = a[i];
        s[i].second = i + 1;
    }
    sort(all(s));

    int sl = 0, sr = 0;
    set<pair<int, int>> l, r;
    for(int i = 0; i < n / 2; ++i){
        sl += s[i].first;
        l.insert({s[i].first, s[i].second});
    }
    for(int j = n / 2; j < n; ++j){
        sr += s[j].first;
        r.insert({s[j].first, s[j].second});
    }

    vector<int> indices(q), powers(q);
    for(int& i: indices)
        cin >> i;
    for(int& i: powers)
        cin >> i;

    for(int i = 0; i < q; ++i){
        int p = indices[i];
        int x = powers[i];
        int cur = a[p - 1];

        auto lref = l.find({cur, p});
        auto rref = r.find({cur, p});
        if(l.find({cur, p}) != l.end()){
            if(x > r.begin()->first){
                sl += r.begin()->first - cur;
                sr += x - r.begin()->first;

                l.erase(lref);
                l.insert(*r.begin());
```

```
64
65                  r.erase(r.begin());
66                  r.insert({x, p});
67              } else {
68                  sl += x - cur;
69                  l.erase(lref);
70                  l.insert({x, p});
71              }
72          }
73          else{
74              auto k = --l.end();
75              if(x < (--l.end())->first){
76                  sl += x - k->first;
77                  sr += k->first - cur;
78
79                  r.erase(rref);
80                  r.insert(*k);
81
82                  l.erase(k);
83                  l.insert({x, p});
84              } else {
85                  sr += x - cur;
86                  r.erase(rref);
87                  r.insert({x, p});
88              }
89          }
90
91          int ans = 2 * sr - 2 * sl;
92          if(n % 2 == 1)
93              ans -= 2 * r.begin()->first;
94          cout << ans << endl;
95          a[p - 1] = x;
96      }
97      return 0;
98  }
```

Задача VI.1.1.4. Трудности перевода (30 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1 секунда.

Ограничение по памяти: 256 мегабайт.

Условие

Ваш друг Харрис часто меняет пароль от компьютера. Когда пароль необходимо сменить, Харрис выбирает строку s длины n, составленную из цифр от 0 до 9, а также систему счисления k. Затем Харрис выполняет следующую последовательность шагов, пока строка s не пуста.

- Выбирает число m ($1 \leqslant m \leqslant |s|$, где $|s|$ — длина строки s на момент выполнения шага) такое, что десятичное число $\overline{s_1 \dots s_m}$ записывается одной **положительной** цифрой k-й системы счисления. Обратите внимание, что на этом шаге строка s не может начинаться с символа 0.
- Дописывает эту цифру в начало создаваемого пароля.

- Стирает первые m символов строки s. Таким образом, длина и индексация строки меняются.

В очередной раз выбрав строку s и систему счисления k, Харрис задался вопросом: «а сколько всего различных паролей я могу получить, следуя намеченному плану?»

Харрису трудно переводить большие числа из одной системы счисления в другую. Помогите Харрису посчитать количество различных паролей, которые он может получить. Это число может быть очень большим, поэтому выведите его остаток от деления на $10^9 + 7$.

Формат входных данных

В первой строке натуральное число n ($1 \leqslant n \leqslant 10^6$) — длина строки s.

Во второй строке натуральное число k ($11 \leqslant k \leqslant 10^{200\,000} - 1$) — система счисления, описанная в условии.

В третьей строке входных данных даётся строка s. Гарантируется, что строка s составлена из цифр десятичной системы счисления — {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}.

Формат выходных данных

Выведите единственное число — остаток от деления на $10^9 + 7$ количества различных паролей, которые может получить Харрис.

Критерии оценивания

Группа	Баллы	Доп. ограничения	Необходимые группы
0	-	Тесты из условия	-
1	10	$1 \leqslant n \leqslant 5$, $11 \leqslant k \leqslant 99$	0
2	20	$1 \leqslant n \leqslant 18$, $11 \leqslant k \leqslant 99$	1
3	20	$1 \leqslant n \leqslant 10^4$, $11 \leqslant k \leqslant 10^{1000} - 1$	2
4	20	Каждый символ строки s выбирается независимо и равновероятно	3
5	30	Без дополнительных ограничений	4

Примеры

Пример №1

Стандартный ввод
5 35 12345
Стандартный вывод
5

Пример №2

Стандартный ввод
18 30 142929242127101712
Стандартный вывод
320

Пример №1

Стандартный ввод
5 36 10001
Стандартный вывод
0

Пояснения к примеру

Пояснение к **первому** тестовому примеру.

В данном тесте Харрис разбивает строку 12345, а система счисления $k = 35$.

- Перечислим все допустимые способы составить пароль в первом примере. Пусть M — массив, в котором m_i равно количеству символов, из которых Харрис составит новую цифру на i-м шаге.
  1. $M = [1, 1, 1, 1, 1]$ — пароль будет «*5-4-3-2-1*».
  2. $M = [1, 1, 2, 1]$ — пароль будет «*5-34-2-1*».
  3. $M = [1, 2, 1, 1]$ — пароль будет «*5-4-23-1*».
  4. $M = [2, 1, 1, 1]$ — пароль будет «*5-4-3-12*».
  5. $M = [2, 2, 1]$ — пароль будет «*5-34-12*».
- Приведем несколько примеров паролей, которые Харрис не мог составить:
  1. $M = [2, 1, 2]$ — в таком случае число, взятое на последнем шаге, будет равно 45, что не является цифрой в системе счисления k;
  2. $M = [3, 1, 1]$ — в таком случае число, взятое на первом шаге, будет равно 123, что также не является цифрой в системе счисления k.

Пояснение ко **второму** тестовому примеру.

В данном примере Харрис выбрал строку $s =$ «*142929242127101712*» и $k = 30$.

Если на каждом шаге выбирать $m = 2$, то получится пароль «*12-17-10-27-21-24-29-29-14*».

Пояснение к **третьему** тестовому примеру.

Можно показать, что Харрис не может разбить данную строку ни одним корректным способом — в любом случае строка s будет начинаться с 0 в какой-либо момент разбиения.

Решение

Сформулируем задачу — сколькими способами можно разбить строку (число) на непересекающиеся отрезки так, чтобы каждый символ (цифра) принадлежал ровно одному отрезку и каждому отрезку соответствовала цифра k-й системы счисления (далее СС).

Применим метод динамического программирования: будем перебирать символы строки слева направо, считаем динамику в массив dp, где dp_i — количество способов разбить префикс строки $[1;\ i]$. Для пересчёта dp_i, будем перебирать количество символов, которые сформируют последнюю цифру k-й СС в разбиении — от 1 до $|k|$ символов.

Возьмём крайние l символов — подстроку $s_{i-l+1} \dots s_i$. Сперва проверим, что символ s_{i-l+1} не равен 0 — цифра k-й СС не может начинаться с 0. Далее два случая:

1. $l < |k|$ — тогда число $\overline{s_{i-l+1} \dots s_i} < k$. Выбранная подстрока может являться последней цифрой $\rightarrow$ прибавим к dp_i значение динамики из dp_{i-l}.
2. $l = |k|$ — необходимо проверить, что $\overline{s_{i-l+1} \dots s_i} < k$. Подстрока $s_{i-l+1} \dots s_i$ и строка k имеют одинаковую длину, достаточно проверить, что в первой отличающейся позиции меньший символ (цифра) стоит в подстроке $s_{i-l+1} \dots s_i$. Если это верно — прибавим к dp_i значение динамики из dp_{i-l}.

Во всех подзадачах, кроме последней, строки можно сравнивать посимвольно. Асимптотика — $O(n \cdot k)$.

Чтобы решить последнюю подзадачу, нужно уметь быстро получать сумму динамик, в позициях, за которыми не следует «0» — насчитаем массив динамик на префиксах. Помимо этого нужно быстро сравнивать подстроки, это можно сделать двумя способами:

1. Вычислим обратные полиномиальные хэши строк k и s — таким образом сможем получать хэш любых их подстрок за константное время. Теперь, чтобы сравнить строку k и какую-то подстроку $s_{i-|k|+1} \dots s_i$, нужно бинарным поиском найти первую позицию j такую, что хэш подстроки $k_1 \dots k_j$ не равен хэшу подстроки $s_{i-|k|+1} \dots s_{i-|k|+j}$. Останется сравнить символы на этой позиции у обеих подстрок. Такое решение будет работать за $O(n \cdot \log_2(|k|))$.
2. В предыдущем пункте мы научились находить первую позицию j, в которой подстроки $k_1 \dots k_j$, $s_{i-|k|+1} \dots s_{i-|k|+j}$ не равны. Теперь сделаем это по-другому. Составим строку $t = k + \text{«\#»} + s$ и вычислим для неё Z-функцию — массив z, в котором z_i равняется наибольшему числу символов, начиная с позиции i, совпадающих с началом строки (подробную информацию о Z-функции можно найти в сети Интернет). Теперь, чтобы сравнить подстроки k и $s_{i-|k|+1} \dots s_i$, нужно получить первую отличающуюся позицию при помощи ячейки z_{i+2}. Тогда отличающаяся позиция будет равна $s_{i-|k|+1+z_{i+2}}$. Если эта позиция выходит за границы подстроки, то $s_{i-|k|+1} \dots s_i = k$. Иначе, сравним символы на этой позиции. Асимптотика — $O(n + |k|)$.

Интересный факт: если заменить цифры в пароле на латинские буквы, то получится «*charlotte*»:

12_{10}	17_{10}	10_{10}	27_{10}	21_{10}	24_{10}	29_{10}	29_{10}	14_{10}
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
c_k	h_k	a_k	r_k	l_k	o_k	t_k	t_k	e_k

Пример программы-решения

Ниже представлено решение на языке C++ через Хэши.

```cpp
#pragma GCC optimize("O3,unroll-loops")
#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")

#include <iostream>
#include <algorithm>
#include <vector>
#include <cmath>
#include <iomanip>
#include <map>
#include <set>
#include <queue>
#include <numeric>
#include <bitset>
#include <algorithm>
using namespace std;

#define sp " "
#define enl '\n'
#define fr first
#define sc second
#define all(x) (x).begin(), (x).end()
#define fast(); ios_base::sync_with_stdio(0);cin.tie(NULL);

#define OR(a, b) ((a) | (b))
#define AND(a, b) ((a) & (b))
#define XOR(a, b) ((a) ^ (b))

#define MOD_SUM(m, x, y) (((x) + (y)) % (m))
#define MOD_DIF(m, x, y) ((((x) - (y)) % (m) + (m)) % (m))

const int p = 257;
const int mod1 = 1e9 + 7;
const int mod2 = 1e9 + 9;
vector< pair<int, int> > h, p_pow;

pair<int, int> get_hash(int l, int r){
    pair<int, int> ans;
    ans.fr = (h[r].fr - h[l - 1].fr * 1ll * p_pow[r - l + 1].fr) % mod1;
    if(ans.fr < 0)
        ans.fr += mod1;
    ans.sc = (h[r].sc - h[l - 1].sc * 1ll * p_pow[r - l + 1].sc) % mod2;
    if(ans.sc < 0)
        ans.sc += mod2;
    return ans;
}

int32_t main(){
    fast();

    int n;
    string k, s;
    cin >> n >> k >> s;

    if(s[0] == '0'){
        cout << 0;
        return 0;
```

```
    }

    string t = "#" + k + s;
    int len_t = (int)t.size();
    int len_k = (int)k.size();

    h.resize(len_t);
    p_pow.resize(len_t);
    h[0] = {0, 0};
    p_pow[0] = {1, 1};


    for(int i = 1; i < len_t; i++){
        h[i].fr = (h[i - 1].fr * 1ll * p + t[i]) % mod1;
        h[i].sc = (h[i - 1].sc * 1ll * p + t[i]) % mod2;

        p_pow[i].fr = (p_pow[i - 1].fr * 1ll * p) % mod1;
        p_pow[i].sc = (p_pow[i - 1].sc * 1ll * p) % mod2;
    }

    vector<int> dp(len_t); dp[len_k] = 1;
    vector<int> sum_dp(len_t); sum_dp[len_k] = 1;

    for(int i = len_k + 1; i < len_t; ++i){
        // add sum of corrects dynamics
        sum_dp[i] = sum_dp[i - 1];
        dp[i] = sum_dp[i - 1] - sum_dp[i - len_k];
        if(dp[i] < 0)
            dp[i] += mod1;

        // check if the length of prefix more than len_k
        if(i - len_k < len_k || t[i - len_k + 1] == '0'){
            if(i + 1 < len_t && t[i + 1] != '0'){
                sum_dp[i] += dp[i];
                if(sum_dp[i] >= mod1)
                    sum_dp[i] -= mod1;
            }
            continue;
        }

        // find common prefix
        int l = 0, r = len_k + 1;
        while(r - l > 1){
            int m = (l + r) / 2;
            if(get_hash(1, m) == get_hash(i - len_k + 1, i - len_k + m))
                l = m;
            else
                r = m;
        }

        // compare first different symbols
        if(l < len_k && t[l + 1] > t[i - len_k + 1 + l]){
            dp[i] += dp[i - len_k];
            if(dp[i] >= mod1)
                dp[i] -= mod1;
        }

        // update sum of dynamics
        if(i + 1 < len_t && t[i + 1] != '0'){
            sum_dp[i] += dp[i];
```

```
            if(sum_dp[i] >= mod1)
                sum_dp[i] -= mod1;
        }
    }
    cout << dp[len_t - 1];
    return 0;
}
```

Ниже представлено решение на языке C++ через Z-функцию.

```
#pragma GCC optimize("O3,unroll-loops")
#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")

#include <iostream>
#include <algorithm>
#include <vector>
#include <cmath>
#include <iomanip>
#include <map>
#include <set>
#include <queue>
#include <numeric>
#include <bitset>
#include <algorithm>
using namespace std;

#define sp " "
#define enl '\n'
#define all(x) (x).begin(), (x).end()
#define fast(); ios_base::sync_with_stdio(0);cin.tie(NULL);

#define OR(a, b) ((a) | (b))
#define AND(a, b) ((a) & (b))
#define XOR(a, b) ((a) ^ (b))

#define MOD_SUM(m, x, y) (((x) + (y)) % (m))
#define MOD_DIF(m, x, y) ((((x) - (y)) % (m) + (m)) % (m))

vector<int> zet_func(int n, string s){
    vector<int> z(n);
    int l = 0, r = 0;
        for (int i = 1; i < n; ++i) {
                if (i <= r)
                        z[i] = min(r - i + 1, z[i - l]);

                while (i + z[i] < n && s[z[i]] == s[i + z[i]])
                        ++z[i];

                if (i + z[i] - 1 > r){
            l = i;
            r = i + z[i] - 1;
                }
        }
        return z;
}

const int mod = 1e9 + 7;

int32_t main(){
    fast();
```

```
51
52      int n;
53      string k, s;
54      cin >> n >> k >> s;
55
56      if(s[0] == '0'){
57          cout << 0;
58          return 0;
59      }
60
61      string t = k + "#" + s;
62      int len_t = (int)t.size();
63      int len_k = (int)k.size();
64      vector<int> z = zet_func(len_t, t);
65
66      vector<int> dp(len_t); dp[len_k] = 1;
67      vector<int> sum_dp(len_t); sum_dp[len_k] = 1;
68
69      for(int i = len_k + 1; i < len_t; ++i){
70          sum_dp[i] = sum_dp[i - 1];
71
72          dp[i] = sum_dp[i - 1] - sum_dp[i - len_k];
73          if(dp[i] < 0)
74              dp[i] += mod;
75
76          if(i - len_k < len_k || t[i - len_k + 1] == '0' || z[i - len_k + 1] >=
            ↪  len_k){
77              if(i + 1 < len_t && t[i + 1] != '0'){
78                  sum_dp[i] += dp[i];
79                  if(sum_dp[i] >= mod)
80                      sum_dp[i] -= mod;
81              }
82              continue;
83          }
84
85          // compare first different symbols
86          int common = z[i - len_k + 1];
87          char p = k[common];
88          char q = t[i - len_k + 1 + common];
89          if(p > q){
90              dp[i] += dp[i - len_k];
91              if(dp[i] >= mod)
92                  dp[i] -= mod;
93          }
94
95          // update sum of dynamics
96          if(i + 1 < len_t && t[i + 1] != '0'){
97              sum_dp[i] += dp[i];
98              if(sum_dp[i] >= mod)
99                  sum_dp[i] -= mod;
100         }
101     }
102     cout << dp[len_t - 1];
103     return 0;
104 }
```

Задача VI.1.1.5. Ограбление по-берляндски (30 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1 секунда.

Ограничение по памяти: 256 мегабайт.

Условие

Компьютеризированный банк Берляндии «*Ко-Банк*» прославился самой надёжной системой управления счетами. Разумеется, хакерская группировка «*Amonamies*» времени зря не теряла и уже установила, что банковские счета управляются компьютерами с номерами 1, ..., N. Про компьютер с номером i известно:

- t_i — время в секундах, требующееся для взлома компьютера;
- c_i — количество бурлей на банковских счетах, к которым имеет доступ компьютер;
- p_i — номер компьютера, который необходимо взломать, чтобы перейти ко взлому i-го.

Помимо этого «*Amonamies*» нашли самое уязвимое место сети — компьютер с номером 1: взлом этого компьютера не требует получения доступа к любому другому компьютеру, поэтому во входных данных $p_1 = 0$.

У «*Amonamies*» будет не более S секунд на взлом компьютеров — далее внутренние механизмы безопасности засекут проникновение и отключат доступ ко всей сети. Сервер устроен так, что в один момент времени «*Amonamies*» могут взламывать **ровно один** компьютер.

Вам, как стажёру в отделе информационной безопасности «*Ко-Банка*», поручили определить максимально возможное количество бурлей, которые «*Amonamies*» могут вывести со взломанных компьютеров (вывод средств производится мгновенно).

Формат входных данных

В первой строке находятся два целых числа N, S ($1 \leqslant N \leqslant 100$, $1 \leqslant S \leqslant 10^5$) — количество компьютеров в сети и время на взлом, которым располагают «*Amonamies*».

Затем следуют N строк с информацией о компьютерах. Строка $i + 1$ содержит три целых числа t_i, c_i, p_i ($1 \leqslant t_i \leqslant 10^3$, $1 \leqslant c_i \leqslant 10^6$, $1 \leqslant p_i < i$ для $i \neq 1$, $p_1 = 0$) — время для взлома, количество бурлей на банковских счетах и номер компьютера, взлом которого необходим для взлома данного компьютера.

Формат выходных данных

Выведите единственное число — максимально возможное количество украденных бурлей.

Критерии оценивания

Группа	Баллы	Дополнительные ограничения	Необходимые группы
0	-	Тесты из условия	-
1	7	$p_i = max(1, i-2)$ для всех $i \geqslant 2$	-
2	10	$p_i = 1$ для всех $i \geqslant 2$	-
3	13	$p_i = \lfloor \frac{i}{2} \rfloor$	-
4	20	$1 \leqslant N \leqslant 20$	-
5	20	$t_i = 1$	-
6	30	Без дополнительных ограничений	$0 - 5$

Примеры

Пример №1

Стандартный ввод
6 20 1 10 0 11 1 1 3 8 1 5 7 2 6 7 3 2 6 4
Стандартный вывод
26

Пример №2

Стандартный ввод
7 45 25 10 0 16 43 1 3 46 1 12 43 1 3 30 3 20 42 3 2 50 2
Стандартный вывод
129

Пояснения к примеру

В первом примере «*Amonamies*» выгоднее всего взломать компьютеры с номерами 1, 2, 3, 4.

Во втором примере «*Amonamies*» выгоднее всего взломать компьютеры с номерами 1, 3, 4, 5.

Решение

Эта задача сводится к задаче о рюкзаке — набрать наиболее ценный груз заданного веса. В данном случае груз — компьютеры, ценность — деньги на счетах, вес — время.

Представим зависимости между компьютерами в виде подвешенного за вершину 1 дерева. Перейдём от задачи максимизации ценности за время не большее S к задаче о минимизации ценности за время не меньшее $\sum_{i=1}^{n} t_i - S$ — хотим «не украсть» как можно меньше за время, которое не можем использовать. Тогда нужно выбрать непересекающиеся поддеревья дерева так, чтобы суммарное время на их взлом было не меньше $\sum_{i=1}^{n} t_i - S$, а ценность минимальная.

Чтобы организовать пересчёт динамики, сохраним вершины в массив в порядке эйлерова обхода — пусть, вершина v лежит в позиции i, тогда все вершины её поддерева вместе с ней будут лежать на отрезке $[i - sz_v + 1;\ i]$, где sz_v — размер поддерева вершины v.

Пусть, $dp_{i,\,j}$ — наименьшая ценность, набранная с весом не менее j, если были рассмотрены первые i вершин из массива. В эту ячейку нужно поместить минимум из ячеек с индексами $[i - sz_v;\ j - time_v] + cost_v$ и $[i - 1;\ j]$, где $time_v$ — суммарное время на взлом всех вершин поддерева v, а $cost_v$ — суммарная ценность всех вершин поддерева вершины v.

Это решение работает за $O\left(n \cdot \left(\sum_{i=1}^{n} t_i - S\right)\right)$ и решает задачу на полный балл.

Пример программы-решения

Ниже представлено решение на языке C++.

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <cmath>
#include <set>
#include <map>
#include <bitset>
#include <random>
#include <numeric>
#include <algorithm>
using namespace std;

#define sp " "
#define all(x) (x).begin(), (x).end()

vector<int> sz, timing, value, order;
vector<vector<int>> e;

void dfs(int v){
    order.push_back(v);
    for(int to: e[v]){
        dfs(to);
        sz[v] += sz[to];
```

```
        timing[v] += timing[to];
        value[v] += value[to];
    }
}

int32_t main(){
    int n, S;
    cin >> n >> S;
    e.resize(n);
    sz.resize(n, 1);
    timing.resize(n);
    value.resize(n);

    int at_least = 0, sum_val = 0;
    for(int i = 0; i < n; ++i){
        int p;
        cin >> timing[i] >> value[i] >> p;
        if(p != 0)
            e[--p].push_back(i);
        at_least += timing[i];
        sum_val += value[i];
    }
    at_least = max((int)0, at_least - S);

    dfs(0);
    reverse(all(order));

    vector<vector<int>> dp(n + 1, vector<int> (at_least + 1, 1e9));
    vector<vector<int>> p(n + 1, vector<int> (at_least + 1));
    dp[0][0] = 0;

    for(int i = 1; i <= n; ++i){
        int v = order[i - 1];
        for(int j = at_least; j > timing[v]; --j)
            if(dp[i - 1][j] > dp[i - sz[v]][j - timing[v]] + value[v]){
                p[i][j] = 1;
                dp[i][j] = dp[i - sz[v]][j - timing[v]] + value[v];
            }
            else
                dp[i][j] = dp[i - 1][j];


        for(int j = min(at_least, timing[v]); j > 0; --j)
            if(dp[i - 1][j] > value[v]){
                p[i][j] = 1;
                dp[i][j] = value[v];
            }
            else
                dp[i][j] = dp[i - 1][j];

        dp[i][0] = 0;
    }

    cout << sum_val - dp[n][at_least];
    return 0;

    set<int> ans;
    for(int i = 0; i < n; ++i)
        ans.insert(i);

```

```
84      int k = n;
85      while(at_least > 0){
86          int v = order[k - 1];
87          if(p[k][at_least]){
88              at_least -= timing[v];
89
90              for(int i = 0; i < sz[v]; ++i, --k)
91                  ans.erase(order[k - 1]);
92          }
93          else
94              --k;
95      }
96
97      cout << ans.size() << endl;
98      for(int i: ans)
99          cout << i + 1 << sp;
100     return 0;
101 }
```

Математика. 8–9 классы

Задача VI.1.2.1. (15 баллов)

Чебурашка решил сделать подарок Гене и придумал новую шахматную фигуру, которую назвал «крокодил». «Крокодил» ходит на две клетки прямо и четыре в сторону. Может ли «крокодил» обойти всю шахматную доску, побывав в каждой клетке хотя бы один раз?

Решение

При любом начальном положении фигуры цвет клетки после хода не меняется.

Ответ: нет.

Критерии оценивания

- 15 баллов.
- Только ответ без обоснования — 0 баллов.

Задача VI.1.2.2. (20 баллов)

На окружности отмечены одна белая и 9 синих точек. Сколько различных выпуклых многоугольников с вершинами в этих точках можно построить, если одна из вершин обязательно белая?

Решение

Треугольников C_9^2. Выбираем из 9 синих точек 2 вершины.

Четырёхугольников C_9^3 и т. д.

Всего $C_9^2 + C_9^3 + C_9^4 + C_9^5 + C_9^6 + C_9^7 + C_9^8 + C_9^9 = 2^9 - C_9^0 - C_9^1 = 2^9 - 1 - 9 = 502$.

Ответ: 502.

Критерии оценивания

- Правильно записана сумма через C_n^k или правильно обоснован путь вычисления искомой величины — +10 баллов.
- Правильно вычислено — +10 баллов.

Задача VI.1.2.3. (20 баллов)

Вася сказал, что придумал трёхзначное простое число, все цифры которого различны и первая цифра равна произведению двух последних. А Юра считает, что таких чисел не существует. Кто из них прав?

Решение

Пусть число имеет вид $\overline{abc}$. $a = b \cdot c$, a, b, c — цифры. Все цифры различны.

Число простое, следовательно, c — нечётное число, $c \neq 5$.

Все цифры различные, следовательно, $c \neq 1$, иначе получаем $a = b$.

Если $c = 7$, то имеем $b = 1$, т. к. $b \cdot c \leqslant 9$. Тогда $b = a = 1$. Но $b \neq a$.

Аналогично, если $c = 9$.

Если $c = 3$, то при $b = 1$ $a = c$. Противоречие с условием.

При $b = 2$ $a = 6$. Проверим, является ли число 623 простым.

$623 = 7 \cdot 89$, следовательно, составное число.

Ответ: прав Юра.

Критерии оценивания

- Только ответ без обоснования — 0 баллов.
- Если ответ верный, но решение недостаточно обосновано — 10 баллов.

Задача VI.1.2.4. (20 баллов)

Из круга вырезали треугольник со сторонами 3, 5 и 6. Известно, что радиус этого круга наименьший из возможных. Найдите его.

Решение

Диаметром будет наибольшая сторона, т. к. треугольник тупоугольный.

А противоположная вершина будет находиться внутри круга.

Центр его описанной окружности находится вне треугольника, два радиуса описанной окружности больше наибольшей стороны треугольника.

Ответ: 3.

Критерии оценивания

- Только ответ без обоснования — 0 баллов.
- Если ответ верный, но решение недостаточно обосновано — 10 баллов.

Задача VI.1.2.5. (25 баллов)

Решите уравнение $\dfrac{x^4 - 1}{x - 1} = 5^y$ в целых числах.

Решение

При любом y $5^y > 0$.

$$\begin{cases} x \neq 1 \\ \frac{(x^2+1)(x^2-1)}{x-1} \end{cases} = 5^y \begin{cases} x \neq 1 \\ \frac{(x^2+1)(x-1)(x+1)}{x-1} = 5^y \end{cases}$$

$$(x^2+1)(x+1) = 5^y,\ x \in Z \Rightarrow 5^y \in Z \Rightarrow y \geqslant 0.$$

Рассмотрим $y = 0$.

$$5^y = 1 \Rightarrow x^3 + x^2 + x + 1 = 1 \Rightarrow x(x^2 + x + 1) = 0 \Rightarrow x = 0.$$

При $y > 0$.

$$5^y \vdots 5 \Rightarrow (x^2+1)(x+1) \vdots 5 \Rightarrow x + 1 = 5k, k \in Z \Rightarrow x = 5k - 1.$$

Но $x^2 + 1 = (5k-1)^2 + 1 = 25k^2 - 10k + 2$ не делится на 5, а правая часть является степенью 5.

Ответ: $x = 0$, $y = 0$.

Критерии оценивания

- Только ответ без обоснования единственности — 5 баллов.
- За верное обоснование $y \geqslant 0$ — +5 баллов.
- Если рассматривается идея делимости на 5 обеих частей уравнения — +5 баллов.

Математика. 10–11 классы

Задача VI.1.3.1. (20 баллов)

Все четырехзначные числа, составленные из цифр 1, 2, 3, занумерованы в порядке возрастания. Какое число находится под номером 49?

Решение

Пусть все числа выписаны в строчку в порядке возрастания. Всего чисел будет $3^4 = 81$. Чисел, начинающихся с 1, будет $3^3 = 27$. Чисел, начинающихся с 1 или 2, будет 54. Следовательно, искомое число будет на пятом месте слева от наибольшего числа, начинающегося с 2, то есть от числа 2333 (№ 54). Перед ним стоят 2332 (№ 53), 2331 (№ 52), 2323 (№ 51), 2322 (№ 50), 2321 (№ 49).

Ответ: 2321.

Критерии оценивания

- Посчитано количество всех чисел — 5 баллов.
- Верные рассуждения, но найденное число «сдвинуто» на 1 влево/вправо — 15 баллов.

Задача VI.1.3.2. (20 баллов)

Существует ли натуральное число, которое в 1511 раза больше суммы его цифр?

Решение

Пусть искомое число A, сумма его цифр a. Требуется, чтобы $1511a = A$, то есть $1510a = A-a$. Поскольку числа A и a имеют одинаковый остаток при делении на 9, то разность $A-a$ кратна 9, и 1510 не кратно 3, то a кратно 9. Перебирая числа a кратные 9, убеждаемся, что $1511 \cdot 27 = 40797$ удовлетворяет условию задачи, поскольку сумма цифр числа равна 27.

Ответ: существует. Например, 40797.

Критерии оценивания

- Доказано, что искомое число кратно 9 — 5 баллов.
- Верные рассуждения, но есть одна вычислительная ошибка — 15 баллов.
- Приведен верный ответ с обоснованием, что это число подходит — 20 баллов.

Задача VI.1.3.3. (20 баллов)

a и b — положительные числа. Сумма минимальных значений функций

$$f(x) = 2ax^2 + 2023x + 6b \text{ и } g(x) = 3bx^2 - 2023x + 4a$$

равна 0.

Чему равны эти минимальные значения?

Решение

Дискриминанты обоих трехчленов, имеющих положительные коэффициенты при старшей степени, одинаковы. В случае их положительности минимальные значения многочленов отрицательны и их сумма не равна 0. В случае их отрицательности минимальные значения многочленов положительны и их сумма не равна 0. Следовательно, дискриминанты равны 0 и минимальное значение многочленов равно 0.

Ответ: 0.

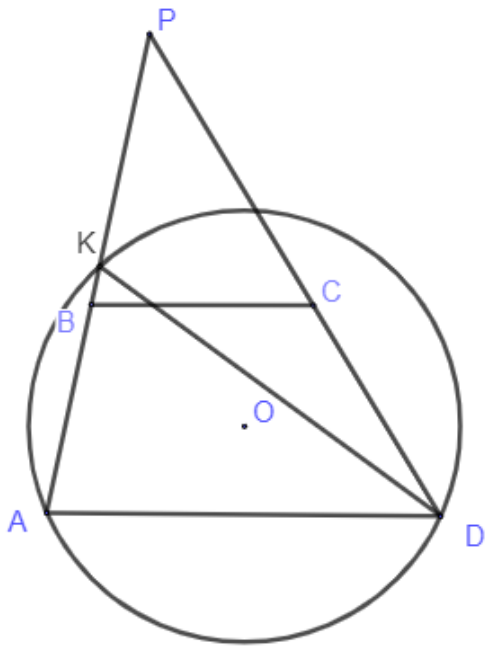
Критерии оценивания

- Доказано, что данные трехчлены имеют одинаковые дискриминанты — 5 баллов.
- Вычислены наименьшие значения трехчленов — 5 баллов.

Задача VI.1.3.4. (20 баллов)

В трапеции $ABCD$ продолжения боковых сторон AB и CD пересекаются в точке P. Окружность радиуса 6 проходит через точки A и D и пересекает луч AB в точке K. Найдите радиус описанной окружности вокруг треугольника KPD, если $AD = 10$, $BC = 4$, $CD = 5$.

Решение



Из подобия треугольников APD и BPC следует, что $\dfrac{PD}{PC} = \dfrac{AD}{BC}$, то есть

$$\frac{PD}{PD-5} = \frac{10}{4}.$$

Откуда $PD = \dfrac{25}{3}$.

Для треугольника AKD данная окружность является описанной, поэтому ее радиус

$$R = \frac{AD}{2\sin\angle AKD}.$$

Откуда $\sin\angle AKD = \dfrac{AD}{2R} = \dfrac{5}{6}$.

Искомая окружность является описанной для треугольника KPD, поэтому ее радиус, учитывая равенство синусов смежных углов AKD и PKD,

$$r = \frac{PD}{2\sin\angle PKD} = \frac{PD}{2\sin\angle AKD} = 5.$$

Ответ: 5.

Критерии оценивания

- Верно найден отрезок PD — 5 баллов.
- Решение верное, но содержит одну вычислительную ошибку — 15 баллов.

Задача VI.1.3.5. (20 баллов)

На всемирной конференции за круглым столом уселись 2023 человек — разведчики, которые всегда говорят правду, и шпионы, которые всегда говорят неправду. Каждого из сидящих спросили, про его двоих соседей — слева и справа — «сколько среди его соседей шпионов?». Каждый человек дал одинаковый ответ. Какое наибольшее количество среди сидящих за столом может быть разведчиков, если известно, что за столом обязательно есть хотя бы 1 разведчик и хотя бы 1 шпион?

Решение

Допустим, что за столом сидят не менее 1012 разведчиков. Тогда обязательно найдутся два разведчика, которые сидят рядом. Тогда если рассматривать всех таких разведчиков, которые сидят рядом, то дойдем до такого, у которого другой сосед – шпион. Поэтому этот разведчик скажет число «1». Поэтому все сидящие должны сказать такое же число. Таким образом, возле каждого разведчика должен быть ровно 1 разведчик, а другой — шпион. А возле каждого шпиона должны быть или 2 разведчика, или 2 шпиона. Поскольку есть рядом сидящие 2 разведчика, то дальнейшая расстановка возможна лишь такая — РРШРРШ. . . , таким образом, количество сидящих за столом должно быть кратным 3, а это не так.

Покажем, что 1011 разведчиков могут быть: РШРШ. . . РШРШШРШ, то есть чередуются Р и Ш и только в одном месте идут подряд ШШ. Тогда каждый может сказать «2» (для Р — это будет правдой, для Ш — ложью).

Ответ: 1011.

Критерии оценивания

- Замечено, что не мог быть дан ответ «два» — 5 баллов.
- Приведена только оценка, что разведчиков не может быть более половины — 10 баллов.
- Приведен только пример, обеспечивающий точность оценки — 5 баллов.
- Наличие оценки и примера — 20 баллов.