

# Московская олимпиада школьников. Информатика. 9 класс. Дистанционный этап, 2024/25

20 дек 2024 г., 10:00 — 20 янв 2025 г., 23:59

100 баллов

## Шутка с таймером

В семье две сестры, Аня и Катя. Однажды, в 7:30 утра, Аня установила таймер на телефоне Кати, который должен сработать через  $k$  минут. Катя не подозревала об этом и отправилась в школу с телефоном. Занятия у Кати начинаются в 9:00. Каждый урок длится  $n$  минут, а перерыв между ними составляет  $m$  минут. Всего у неё запланировано  $t$  уроков. Определите, на каком уроке прозвонит таймер и удастся ли Ане осуществить свою шутку. Учтите, что звонок на урок или с урока начинается в первую секунду минуты, а таймер срабатывает в тридцатую секунду минуты.

Вам необходимо выяснить номер урока, когда прозвонит таймер. Если таймер сработает во время перемены, до начала первого урока или после завершения последнего, выведите -1.

## Формат входных данных

Четыре целых числа даны каждый в своей строке в следующем порядке:  $k, n, m, t$  — число минут на таймере, длина урока, длина перемены и количество уроков,  
 $1 \leq k \leq 1000, 10 \leq n \leq 100, 10 \leq m \leq 100, 2 \leq t \leq 10$ .

## Формат выходных данных

Выведите единственное число — номер урока в течение которого сработал будильник или  $-1$ , если будильник сработал во время перемены, до первого или после последнего урока, т. е. вне уроков.

## Критерии оценивания

Каждый тест оценивается независимо.

### Примеры

Ввод	Вывод
95 45 10 5	1
40 45 10 5	-1
400 45 15 5	-1

### Ограничения

Время выполнения: 1 секунда  
Память: 256 MB

Код

C++

```
1  #include "iostream"
2
3  typedef long double ld;
4
5  #define int long long
6  #define double ld
7
8  using namespace std;
9  signed main()
10 {
11     int k, n, m, t;
12     cin >> k >> n >> m >> t;
13
14     int start = 7 * 60;
15     int end = start + k;
16
17     int l = 8 * 60 + 30;
18
19     for (int i = 0; i < t; i++)
20     {
21         int r = l + n - 1;
22
23         if (end >= l && end <= r)
24             return cout << i + 1, 0;
25
26         l = r + 1 + m;
27     }
28
29     cout << -1;
30 }
31
32
```

100 баллов

## Подходящие числа

Математикам обычно нравятся только особенные числа, например, простые, у которых нет делителей кроме 1 и самого числа, или степени двойки. Назовем такие числа *подходящими*. Поэтому, если к ужину смарт-часы показывают некоторое количество пройденных за день шагов, и оно не является подходящим числом, то математик сделает минимально возможное дополнительное число шагов, чтобы счетчик шагов показывал подходящее число.

Помогите математикам в определении ближайшего к текущим показаниям часов подходящего числа.

## Формат входных данных

В первой строке дано одно целое число  $n$  ( $1 \leq n \leq 10^5$ ) — число дней, в которые снимались показания часов.

Во второй строке даны  $n$  целых чисел  $a_i$  ( $0 \leq a_i \leq 10^7$ ) — вечерние показания часов в течение  $n$  дней.

## Формат выходных данных

Выведите  $n$  чисел: для каждого  $a_i$  ближайшее подходящее число, не меньшее его.

## Критерии оценивания

Подзадача	Баллы	Ограничения
0	0	Тесты из условия
1	25	$n \leq 1000, a_i \leq 1000$
2	17	$n \leq 1000$
3	28	$a_i \leq 1000$
4	30	Без дополнительных ограничений

### Примеры

Ввод	Вывод
5 2020 1023 0 101 10000	2027 1024 1 101 10007

### Ограничения

Время выполнения: 4 секунды

Память: 256 MB

Код

C++

```

1  #include <bits/stdc++.h>
2
3  #define s second
4  #define f first
5  #define pb push_back
6  #define forn(i, n) for (int i = 0; i < (int)(n); ++i)
7  #define all(a) (a).begin(), (a).end()
8  #define len(a) (int)(a).size()
9  #define INF (int)(1e9)
10 // #define endl '\n'
11 // #define int long long
12
13 using namespace std;
14
15 const int N = 100100;
16 const int M = 10010010;
17
18 bool r[M]; // 0 -> prime
19 vector<int> good;
20
21 void preprocess() {
22     r[0] = r[1] = 1;
23     vector<int> p;
24     forn(i, M) {
25         if (!r[i]) {
26             p.pb(i);
27             for (int j = min(M + 0LL, 1LL * i * i); j < M; j += i)
28             {
29                 r[j] = 1;
30             }
31         }
32         int puw = 1;
33         vector<int> pw;
34         while (puw < M) {
35             pw.pb(puw);
36             puw *= 2;
37         }
38         good.resize(len(p) + len(pw));
39         merge(all(p), all(pw), good.begin());
40     }
41
42     signed main() {
43         ios_base::sync_with_stdio(false);
44         cin.tie(nullptr);
45         cout.precision(37);
46
47         preprocess();
48
49         int n;
50         cin >> n;
51         forn(i, n) {
52             int x;
53             cin >> x;
54             cout << good[lower_bound(all(good), x) - good.begin()] << '
55         ';
56     }
57     return 0;
58 }
59

```

100 баллов

## Симметричная впадина

Массив натуральных чисел называется *впадиной*, если он сначала строго убывает, затем строго возрастает. Длина впадины как минимум 3. Например, 4 3 2 6 9 — это впадина, потому что первые три числа в нем строго убывают, а потом с третьего по пятое число — строго возрастают.

Массив натуральных чисел называется *палиндромом*, если он один и тот же при записи слева направо и справа налево. Например, 1 3 2 3 1 — это палиндром.

*Подмассив* массива — массив, полученный при удалении любого префикса (начала) и любого суффикса (конца) исходного массива. Он состоит из подряд идущих элементов исходного массива.

Для заданного массива определите, какова длина самого длинного подмассива данного массива, являющегося одновременно и впадиной, и палиндромом. Например, 4 3 2 3 4 — это палиндром и впадина одновременно.

## Формат входных данных

В первой строке дано одно целое число  $n$  — размер массива,  $1 \leq n \leq 10^6$ . Во второй строке  $n$  целых чисел через пробел  $a_i$  — элементы массива,  $1 \leq a_i \leq 10^9$ .

## Формат выходных данных

Выведите единственное целое число — максимальную длину палиндромной ямы или  $-1$ , если такой нет. Напомним, что длина ямы должна быть как минимум 3.

## Критерии оценивания

Все тесты разбиты на группы со следующими ограничениями:

$1 \leq n \leq 50$ , 15 баллов

$1 \leq n \leq 300$ , 15 баллов

$1 \leq n \leq 7000$ , 25 баллов

$1 \leq n \leq 1000000$ , 45 баллов

Очередная группа проверяется только после прохождения всех тестов всех предыдущих групп.

### Примеры

Ввод	Вывод
8 11 9 5 2 1 2 5 7	5

### Ограничения

Время выполнения: 2 секунды

Память: 256 MB

Код

C++

```

1  #include "iostream"
2  #include "algorithm"
3  #include "string"
4  #include "vector"
5  #include "cstring"
6
7
8  using namespace std;
9
10 #define maxi(a,b) a = max(a, b);
11 #define mini(a,b) a = min(a, b);
12
13 #define all(a) a.begin(), a.end()
14 #define rall(a) a.rbegin(), a.rend()
15 #define sqr(x) ((x) * (x))
16 #define SZ(a) ((int)(a.size()))
17 #define watch(x) cout << (#x) << " = " << x << endl;
18 typedef long double ld;
19
20 #define int long long
21 typedef vector<int> vi;
22
23 signed main()
24 {
25     ios::sync_with_stdio(0);
26     cout.tie(0); cin.tie(0);
27
28     int n;
29     cin >> n;
30
31     vi a(n);
32     for (int i = 0; i < n; i++)
33     {
34         cin >> a[i];
35     }
36
37     int ans = 0;
38     for (int i = 1; i < n - 1; i++)
39     {
40         if (a[i] < a[i - 1] && a[i] < a[i + 1])
41         {
42             int len = 0;
43             while (1)
44             {
45                 if (i - len - 1 < 0 || i + len + 1
46                 >= n)
47                     break;
48                 if (a[i - len - 1] <= a[i - len])
49                     break;
50                 if (a[i + len + 1] <= a[i + len])
51                     break;
52                 if (a[i - len - 1] != a[i + len +
53                 1])
54                     break;
55                 len++;
56             }
57             maxi(ans, 1 + 2 * len);
58         }
59     }
60 }
61
62
63

```

```
64         ans >= 3 ? cout << ans : cout << -1;  
65     }  
66  
67
```



100 баллов

## Дорога в школу

Будем считать, что дорога в школу начинается в момент времени 0 и состоит из  $n$  участков. Ученик передвигается по  $i$ -му участку пути со скоростью  $V_i$  в течение  $T_i$  единиц времени. Требуется  $q$  запросов определить, чему равна **средняя скорость** на части пути ученика в различные промежутки времени.

**Средняя скорость** на части пути вычисляется как отношение всего преодолённого расстояния к затраченному на это времени.  $V = \frac{S}{T}$ .

## Формат входных данных

В первой строке записаны два натуральных числа  $n, q (1 \leq n, q \leq 10^5)$  — количество участков пути и количество запросов.

В следующих  $n$  строках записаны пары целых чисел  $V_i, T_i (1 \leq V_i, T_i \leq 10^5)$  — информация об участках пути ученика в школу.

В следующих  $q$  строках даны запросы. Каждая строка содержит два целых числа  $L_q, R_q (0 \leq L_q < R_q \leq \sum_{i=1}^N T_i)$  — начальный и конечный моменты времени интересующего промежутка.

## Формат выходных данных

Для каждого запроса выведите среднюю скорость на части пути в соответствующем промежутке времени. Ответ выведите с точностью не менее чем четыре знака после точки (но, например, если ответ целый, лишние нули после точки можно не выводить, и саму точку можно не ставить).

## Критерии оценивания

Тесты разделены на несколько подзадач. Баллы выставляются только за подзадачу в целом. Следующая подзадача проверяется только при прохождении всех тестов предыдущих подзадач.

**Подзадача 1** (16 баллов):  $N \leq 100, Q \leq 100, \sum_{i=1}^N T_i \leq 1000$ .

**Подзадача 2** (14 баллов): на  $N$  и  $Q$  нет дополнительных ограничений,  $\sum_{i=1}^N T_i \leq 10^5$ .

**Подзадача 3** (15 баллов):  $N \leq 100, Q \leq 100$ , на  $\sum_{i=1}^N T_i$  нет дополнительных ограничений.

**Подзадача 4** (10 баллов): Гарантируется, что для каждого запроса  $q$  в момент времени  $L_q$  и в момент времени  $R_q$  Ученик находился на одном и том же участке пути (участок может быть разным для разных запросов).

**Подзадача 5** (15 баллов): Гарантируется, что для каждого запроса  $q$  выполняется следующее: либо ученик преодолел участок пути целиком, либо вообще не передвигался по нему.

**Подзадача 6** (30 баллов): Ограничения из условия.

### Примеры

Ввод	Вывод
5 7	4.6250
3 4	9.0000
1 1	4.0000
4 2	2.0000
9 5	5.3333
2 4	2.8333
0 16	7.0000
9 10	
5 7	
13 15	
2 14	

0 6	
5 10	

Ограничения

Время выполнения: 2 секунды  
Память: 256 МВ

Код

C++

```

1  #include <iostream>
2  #include <bits/stdc++.h>
3
4  using namespace std;
5  using ll = long long int;
6
7
8  #define trace(arg) #arg << ": " << (arg) << "\n"
9  #define trace_log cerr << trace(__LINE__)
10
11 const int max_n = 1e5 + 5;
12
13
14 ll segments_amount;
15 ll time_pref[max_n];
16 ll dist_pref[max_n];
17 ll speed[max_n];
18
19
20 void solve();
21
22 int main() {
23     cin.tie(NULL);
24     cout.tie(NULL);
25     cin.sync_with_stdio(false);
26
27     int n;
28     int q = 1;
29     cin >> n >> q;
30     segments_amount = n;
31
32     for (int i = 1; i <= n; i++) {
33         ll v, t;
34         cin >> v >> t;
35         time_pref[i] = time_pref[i - 1] + t;
36         dist_pref[i] = dist_pref[i - 1] + v * t;
37         speed[i-1] = v;
38     }
39
40     while (q--) solve();
41     return 0;
42 }
43
44 ll get_coord(ll t) {
45     ll l = 0, r = segments_amount;
46     ll mid = (l + r) / 2;
47     while (r - l > 1) {
48         if (t <= time_pref[mid]) r = mid;
49         else l = mid;
50         mid = (l + r) / 2;
51     }
52     return dist_pref[l] + speed[l]*(t-time_pref[l]);
53 }
54
55 void solve() {
56     ll st, ft;
57     cin >> st >> ft;
58
59     double dur = ft - st;
60     double dist = get_coord(ft) - get_coord(st);
61
62     cout << setprecision(4) << fixed << dist / dur << endl;

```



100 баллов

# Шахматный путь

Друзья играют в следующую игру: один называет шахматную фигуру, второй — стартовую клетку на стандартной шахматной доске, третий — конечную клетку. Далее они соревнуются — кто раньше определит минимальное количество ходов, за которое эта фигура может добраться из **стартовой** клетки в **конечную**. Друзья договорились, что все фигуры будут белыми, а шахматная доска будет считаться свободной от других фигур.

Ряды на доске обозначаются буквами и цифрами, начиная от левого нижнего углового поля  $a1$  — чёрного цвета. Вертикальные ряды — латинскими буквами  $a-h$ , горизонтальные — цифрами 1-8.

**Правила передвижения фигур** (адаптированы под задачу):

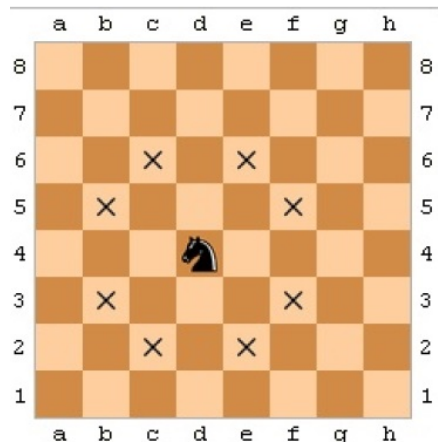
Король (KING) передвигается со своего поля на одно из свободных смежных полей.

Ферзь (QUEEN) ходит по вертикалям, диагоналям и горизонталям, на которых он находится.

Ладья (ROOK) ходит по вертикалям и горизонталям, на которых она находится.

Слон (BISHOP) ходит по диагоналям, на которых он находится.

Конь (KNIGHT) может пойти на одно из полей, ближайших к тому, на котором он стоит, но не на той же самой горизонтали, вертикали или диагонали.



Пешка (PAWN) передвигается на одно поле только вперёд. Если стартовая клетка соответствует стартовой позиции белой пешки в настоящей игре (это горизонталь  $a2-h2$ ), пешка может пойти как на одну, так и на две клетки вперёд в первом ходу. Любая пешка, достигающая крайней горизонтали (для белых это горизонталь  $a8-h8$ ), должна быть тем же ходом заменена на ферзя того же цвета, что и пешка. Если в тестовом случае пешка изначально находится на горизонтали  $a8-h8$ , то она сразу заменяется на ферзя (эта замена ходом не считается).

## Формат входных данных

В первой строке записано число  $T(1 \leq T \leq 24576)$  — количество случаев, на которые нужно получить ответ.

В следующих  $T$  строках описаны сами случаи. Каждое описание начинается со строки, задающей фигуру:

KING — король, QUEEN — ферзь, ROOK — ладья, BISHOP — слон, KNIGHT — конь, PAWN — пешка.

Далее, через пробел, записаны координаты стартовой клетки в формате  $sx$ , где  $s$  — строчная английская буква от 'a' до 'h', обозначающая ряд, а  $x$  — натуральное число от 1 до 8, обозначающее столбец. Далее, через пробел, задана конечная клетка в аналогичном формате.

## Формат выходных данных

Для каждого тестового случая выведите ответ в отдельной строке — минимальное количество ходов, которое необходимо сделать указанной фигуре, чтобы прийти из стартовой клетки в конечную клетку.

В случае, если конечная клетка недостижима из стартовой, выведите  $-1$ .

## Замечание

В примере у первой пешки стартовая позиция совпадает со стартовой позицией в настоящей игре, поэтому она может достичь конечной за один ход. Для второй пешки это не так, поэтому она может ходить вперед только в соседнюю клетку.

# Критерии оценивания

Все тесты оцениваются независимо.

Во многих тестах во всех случаях упоминается только одна из фигур.

## Примеры

Ввод	Вывод
9 ROOK c5 h2 KING a1 a2 PAWN b2 b4 PAWN b1 b4 BISHOP e5 e6 PAWN c5 e5 QUEEN e5 g7 KNIGHT a1 c1 KNIGHT a1 a1	2 1 1 3 -1 5 1 2 0

## Ограничения

Время выполнения: 1 секунда

Память: 256 MB

Код

C++

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  #include <queue>
5
6  #define vi vector<int>
7  #define pii pair<int, int>
8  #define vii vector<pii>
9  using namespace std;
10
11 int solveKing(int x1, int y1, int x2, int y2) {
12     return max(abs(x2 - x1), abs(y2 - y1));
13 }
14
15 int solveQueen(int x1, int y1, int x2, int y2) {
16     if (x1 == x2 && y1 == y2) return 0;
17     if (x1 == x2) return 1;
18     if (y1 == y2) return 1;
19     if (x1 + y1 == x2 + y2) return 1;
20     if (x1 - y1 == x2 - y2) return 1;
21     return 2;
22 }
23
24 int solveRook(int x1, int y1, int x2, int y2) {
25     if (x1 == x2 && y1 == y2) return 0;
26     if (x1 == x2) return 1;
27     if (y1 == y2) return 1;
28     return 2;
29 }
30
31 int solveKnight(int x1, int y1, int x2, int y2) {
32     queue<pii> q;
33     x1--;
34     y1--;
35     x2--;
36     y2--;
37
38     const int dx[8] = {1, 1, -1, -1, 2, 2, -2, -2};
39     const int dy[8] = {2, -2, 2, -2, 1, -1, 1, -1};
40
41     int dist[8][8];
42     for (int i = 0; i < 8; i++) {
43         for (int j = 0; j < 8; j++) dist[i][j] = -1;
44     }
45     dist[x1][y1] = 0;
46     q.push({x1, y1});
47     while (!q.empty()) {
48         int x = q.front().first;
49         int y = q.front().second;
50         q.pop();
51         for (int i = 0; i < 8; i++) {
52             int xto = x + dx[i];
53             int yto = y + dy[i];
54             bool inside = (xto >= 0 && xto < 8 && yto >= 0 && yto <
8);
55             if (!inside) continue;
56             if (dist[xto][yto] == -1) {
57                 dist[xto][yto] = dist[x][y] + 1;
58                 q.push({xto, yto});
59             }
60         }
61     }
62     return dist[x2][y2];
63 }
64

```

```

65     int solveBishop(int x1, int y1, int x2, int y2) {
66         if ((x1 + y1) % 2 != (x2 + y2) % 2) return -1;
67         if (x1 == x2 && y1 == y2) return 0;
68         if (y1 - x1 == y2 - x2) return 1;
69         if (x1 + y1 == x2 + y2) return 1;
70         return 2;
71     }
72
73     int solvePawn(int x1, int y1, int x2, int y2) {
74         bool first = true;
75         int res = 0;
76         if (x1 == x2 && y1 == y2) return 0;
77         if (x2 == x1 + 1 && y2 == y1) return 1;
78
79         while (x1 != 8) {
80             if (x1 == 2 && first) x1 += 2;
81             else x1++;
82             res++;
83             if (x1 == x2 && y1 == y2) return res;
84             first = false;
85         }
86
87         return res + solveQueen(x1, y1, x2, y2);
88     }
89
90     void solve() {
91         string w1, w2;
92         int x1, y1, x2, y2;
93         string s;
94         cin >> s >> w1 >> w2;
95         y1 = w1[0] - 'a' + 1;
96         x1 = w1[1] - '0';
97         y2 = w2[0] - 'a' + 1;
98         x2 = w2[1] - '0';
99         if (s == "KING") {
100             cout << solveKing(x1, y1, x2, y2);
101         }
102         else if (s == "BISHOP") {
103             cout << solveBishop(x1, y1, x2, y2);
104         }
105         else if (s == "KNIGHT") {
106             cout << solveKnight(x1, y1, x2, y2);
107         }
108         else if (s == "ROOK") {
109             cout << solveRook(x1, y1, x2, y2);
110         }
111         else if (s == "PAWN") {
112             cout << solvePawn(x1, y1, x2, y2);
113         }
114         else {
115             cout << solveQueen(x1, y1, x2, y2);
116         }
117         cout << "\n";
118     }
119
120     int main()
121     {
122         int T = 1;
123         cin >> T;
124         while (T--) {
125             solve();
126         }
127         return 0;
128     }

```





100 баллов

## Правильные подматрицы

Дана матрица (таблица) из  $n$  строк и  $m$  столбцов, заполненная строчными буквами латинского алфавита.

Назовем матрицу *правильной*, если в ней встречаются **ровно две различные** буквы, и они расположены в шахматном порядке (одна буква на местах белых клеток, вторая — чёрных).

Следующие матрицы **являются** правильными:

<i>o</i>	<i>x</i>	<i>o</i>
<i>x</i>	<i>o</i>	<i>x</i>
<i>o</i>	<i>x</i>	<i>o</i>

<i>x</i>	<i>o</i>	<i>x</i>
<i>o</i>	<i>x</i>	<i>o</i>
<i>x</i>	<i>o</i>	<i>x</i>

<i>a</i>
<i>b</i>
<i>a</i>

Следующие матрицы **не являются** правильными:

<i>o</i>	<i>x</i>	<i>o</i>
<i>x</i>	<i>x</i>	<i>x</i>
<i>o</i>	<i>x</i>	<i>o</i>

<i>a</i>	<i>a</i>	<i>a</i>
<i>a</i>	<i>a</i>	<i>a</i>
<i>a</i>	<i>a</i>	<i>a</i>

<i>b</i>
----------

Требуется найти количество *правильных* подматриц данной матрицы.

Подматрицей называется любая прямоугольная часть исходной матрицы. Она получается из исходной матрицы сначала отбрасыванием нескольких (возможно, 0) подряд идущих строк в начале и нескольких строк (возможно, 0) в конце, а затем в полученной матрице можно отбросить несколько (возможно, 0) столбцов в начале и несколько (возможно, 0) столбцов в конце.

## Формат входных данных

В первой строке даны два целых числа  $n$  и  $m$  ( $2 \leq n, m \leq 300$ ) — количество строк и столбцов в матрице соответственно.

В каждой из следующих  $n$  строк задана последовательность, состоящая из  $m$  строчных букв латинского алфавита.

## Формат выходных данных

Выведите единственное число — количество правильных подматриц данной матрицы.

## Критерии оценивания

Все тесты оцениваются независимо.

### Примеры

Ввод	Вывод
2 2 aa aa	0
2 2 ab cd	4

2 2 ab ba	5
-----------------	---

Ограничения

Время выполнения: 3 секунды

Память: 256 MB

Код

C++

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define pb  push_back
4  #define ll  long long
5  #define vi  vector<int>
6  #define vvi vector<vi >
7  #define all(x) x.begin(), x.end()
8
9  ll res = 0;
10 ll n, m;
11 string s[300], t[300];
12 ll d[300];
13
14 ll co(ll x) {
15     return (x * (x - 1)) / 2;
16 }
17
18 int main() {
19     ios_base::sync_with_stdio(false); cin.tie(0);
20     cin >> n >> m;
21     for (ll i = 0; i < n; ++i)
22         cin >> s[i];
23     for (ll i = 0; i < n; ++i) {
24         ll b = 0;
25         for (ll j = 1; j < m; ++j) {
26             if (s[i][j] == s[i][j - 1]) {
27                 res += co(j - b);
28                 b = j;
29             }
30             else if (j > 1 && s[i][j] == s[i][j - 2])
31                 ;
32             else {
33                 res += co(j - b);
34                 b = j - 1;
35             }
36         }
37         res += co(m - b);
38     }
39     for (ll i = 0; i + 1 < n; ++i) {
40         for (ll j = 0; j < m; ++j) {
41             d[j] = 2; t[j] = "++";
42         }
43         for (ll j = 0; j < m; ++j) {
44             if (s[i][j] == s[i + 1][j]) continue;
45             t[j][0] = s[i][j];
46             t[j][1] = s[i + 1][j];
47             if (j % 2) swap(t[j][0], t[j][1]);
48             for (ll I = i + 2; I < n; ++I) {
49                 if (s[I][j] != s[I - 2][j]) break;
50                 ++d[j];
51             }
52         }
53         for (ll h = 2; i + h - 1 < n; ++h) {
54             ll b = 0;
55             for (ll j = 0; j < m; ++j) {
56                 if (t[j] == "++" || d[j] < h) {
57                     res += co(j - b + 1);
58                     b = j + 1;
59                 }
60                 else if (j > 0 && t[j] == t[j - 1])
61                     ;
62                 else {
63                     res += co(j - b + 1);
64                     b = j;
65                 }

```

```
66         }
67         res += co(m - b + 1);
68     }
69 }
70 cout << res << endl;
71
72 return 0;
73 }
74
```

100 баллов

## Доминирующий элемент

Дан массив из  $n$  натуральных чисел, проиндексированный с 1 до  $n$  включительно. Рассмотрим некоторый отрезок массива  $[l, r]$  включительно, и назовем элемент  $x$  *доминирующим* на этом отрезке, если он встречается на нём строго больше раз, чем все остальные элементы вместе взятые.

Дано  $q$  запросов, в каждом запросе даны  $[l_q, r_q]$  — границы отрезка. Найдите доминирующий элемент этого отрезка либо сообщите, что на этом отрезке его нет.

## Формат входных данных

В первой строке вам даны два числа  $n, q (1 \leq n, q \leq 10^5)$  — количество элементов в массиве и количество запросов.

Во второй строке задан массив из  $n$  натуральных чисел  $a_i (1 \leq a_i \leq 10^9)$ .

В следующих  $q$  строках заданы запросы, в каждой новой строке по два числа  $l_q, r_q (1 \leq l_q \leq r_q \leq n)$  — границы отрезка в очередном запросе.

## Формат входных данных

Для каждого запроса в новой строке выведите значение доминирующего элемента на этом отрезке, либо  $-1$ , если доминирующего элемента на отрезке нет.

## Критерии оценивания

**Подзадача 1** (18 баллов):  $n \leq 500, q \leq 500$ .

**Подзадача 2** (14 баллов):  $n \leq 3000, q \leq 3000$ .

**Подзадача 3** (20 баллов):  $n \leq 20000, q \leq 20000$ .

**Подзадача 4** (16 баллов): В массиве присутствует не более 10 различных значений.

**Подзадача 5** (32 балла): Ограничения из условия.

Баллы за подзадачу выставляются только при прохождении всех тестов подзадачи. Подзадача проверяется, если все тесты всех предыдущих подзадач пройдены.

### Примеры

Ввод	Вывод
5 4	3
3 2 3 1 3	3
1 3	3
3 5	-1
1 5	
1 4	

### Ограничения

Время выполнения: 4 секунды

Память: 512 MB

Код

C++

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4
5  #define vi vector<int>
6  #define pii pair<int, int>
7  #define vii vector<pii>
8  using namespace std;
9
10 void compress(vi &a, vi &val) {
11     int n = a.size();
12     vii p(n);
13     for (int i = 0; i < n; i++) p[i] = {a[i], i};
14     sort(p.begin(), p.end());
15     int curval = 0;
16     for (int i = 0; i < n; i++) {
17         if (i && p[i].first != p[i - 1].first) curval++;
18         val[curval] = p[i].first;
19         a[p[i].second] = curval;
20     }
21 }
22
23
24 vi tree;
25 vector<vi> occ;
26 vi val, a;
27
28 bool check(int x, int l, int r) {
29     int L = (int)(lower_bound(occ[x].begin(), occ[x].end(), l) -
occ[x].begin());
30     int R = (int)(lower_bound(occ[x].begin(), occ[x].end(), r + 1)
- occ[x].begin() - 1);
31     return max(0, R - L + 1) > (r - l + 1) / 2;
32 }
33
34 void build(int v, int tl, int tr) {
35     if (tl == tr) {
36         tree[v] = a[tl];
37         return;
38     }
39     int tm = (tl + tr) / 2;
40     build(v * 2, tl, tm);
41     build(v * 2 + 1, tm + 1, tr);
42     if (tree[v * 2] != -1 && check(tree[v * 2], tl, tr)) tree[v] =
tree[v * 2];
43     else if (tree[v * 2 + 1] != -1 && check(tree[v * 2 + 1], tl,
tr)) tree[v] = tree[v * 2 + 1];
44     else tree[v] = -1;
45 }
46
47 int get(int v, int tl, int tr, int l, int r) {
48     if (l > r) return -1;
49     if (l == tl && r == tr) return tree[v];
50     int tm = (tl + tr) / 2;
51     int x1 = get(v * 2, tl, tm, l, min(r, tm));
52     int x2 = get(v * 2 + 1, tm + 1, tr, max(l, tm + 1), r);
53     if (x1 != -1 && check(x1, l, r)) return x1;
54     if (x2 != -1 && check(x2, l, r)) return x2;
55     return -1;
56 }
57
58 void solve() {
59     int n, q;
60     cin >> n >> q;
61     a.resize(n);

```

```
62     val.resize(n);
63     for (int i = 0; i < n; i++) {
64         cin >> a[i];
65     }
66     compress(a, val);
67     tree.resize(4 * n);
68     occ.resize(n);
69     for (int i = 0; i < n; i++) occ[a[i]].push_back(i);
70     build(1, 0, n - 1);
71     for (int i = 0; i < q; i++) {
72         int l, r;
73         cin >> l >> r;
74         l--;
75         r--;
76         int x = get(1, 0, n - 1, l, r);
77         if (x != -1) x = val[x];
78         cout << x << "\n";
79     }
80 }
81
82 int main()
83 {
84     int T = 1;
85     //cin >> T;
86     while (T--) {
87         solve();
88     }
89     return 0;
90 }
```



100 баллов

## Граф. Просто граф

Дан неориентированный связный граф, состоящий из  $n$  вершин, пронумерованных от 1 до  $n$ , и  $m$  ребер, а также  $q$  запросов. В каждом из запросов, вам требуется по заданным различным вершинам  $u$  и  $v$ , найти количество различных простых путей, соединяющих  $u$  и  $v$ . Напомним, что путь называется простым, если этот путь проходит по каждой из вершин графа не более чем по одному разу.

### Формат входных данных

В первой строке записаны целые неотрицательные числа  $n$  и  $m$  ( $1 \leq n \leq 200\,000$ ,  $0 \leq m \leq n + 4$ ) — количество вершин и рёбер графа.

Далее в следующих  $m$  строках записано по паре целых чисел  $u, v$ ,  $1 \leq u, v \leq n$ ,  $u \neq v$ .

В следующей строке записано целое число  $q$ ,  $1 \leq q \leq 2\,000\,000$  количество запросов. В следующих  $q$  строках записано по паре целых чисел  $u, v$ ,  $1 \leq u, v \leq n$ ,  $u \neq v$ , описывающих запрос.

Гарантируется, что в графе не существует петель и кратных ребер.

### Формат выходных данных

Для каждого запроса выведите в отдельной строке количество различных путей между вершинами  $u$  и  $v$ .

### Критерии оценивания

Каждый тест оценивается независимо. Как минимум в 60% тестов  $n$  и  $m$  не превосходят 50.

#### Примеры

Ввод	Вывод
3 3 1 2 2 3 3 1 3 1 2 3 1 2 3	2 2 2
4 4 1 2 2 3 3 1 1 4 6 1 2 1 3 1 4 2 3 2 4 3 4	2 2 1 2 2 2

#### Ограничения

Время выполнения: 4 секунды

Память: 512 MB

Код

C++

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  const int MOD = 1000000007;
6  const int MAXLOG = 20;
7  const int MAX_ADD_EDGES = 6;
8  const int MAX_BASE_VERTICES = 3 * MAX_ADD_EDGES - 1;
9
10 int n, nn, m, nmcd, nm, compNumber;
11 vector<vector<int>> > gr, ngr, achievableBaseVertices;
12 vector<int> used, h, pr, base, isBased, baseVerticesNumbers,
    baseEdgeNumbers, abv;
13 vector<pair<int, int>> edges, baseEdges;
14
15 int st[MAXLOG][1<<MAXLOG];
16 int firstPos[1<<MAXLOG], vs[1<<MAXLOG], hs[1<<MAXLOG],
    floorlog[1<<MAXLOG];
17 int stind;
18
19 int dp[1 << MAX_BASE_VERTICES][MAX_BASE_VERTICES];
20 int ans0[MAX_BASE_VERTICES][MAX_BASE_VERTICES];
21 int ans1[MAX_BASE_VERTICES][MAX_BASE_VERTICES];
22 int ans2[MAX_BASE_VERTICES][MAX_BASE_VERTICES];
23
24 void addEdge(int v1, int v2, vector<vector<int>> > &gr) {
25     gr[v1].push_back(v2);
26     gr[v2].push_back(v1);
27 }
28
29 void readData() {
30     scanf("%d%d", &n, &m);
31     gr.assign(n, vector<int>());
32     for (int i = 0; i < m; ++i) {
33         int v1, v2;
34         scanf("%d%d", &v1, &v2);
35         --v1;
36         --v2;
37         addEdge(v1, v2, gr);
38     }
39 }
40
41 void addToST(int v) {
42     vs[stind] = v;
43     hs[stind++] = h[v];
44 }
45
46 void dfsLCA(int v, int prv) {
47     used[v] = 1;
48     firstPos[v] = stind;
49     addToST(v);
50     for (int nv : gr[v]) {
51         if (nv == prv)
52             continue;
53         if (used[nv] == 0) {
54             pr[nv] = v;
55             h[nv] = h[v] + 1;
56             dfsLCA(nv, v);
57             addToST(v);
58         }
59         else if (used[nv] == 1)
60             edges.push_back(make_pair(v, nv));
61     }
62     used[v] = 2;
63 }

```

```

64
65 void buildLCAAndFindNonTreeEdges() {
66     stind = 0;
67     used.assign(n, 0);
68     h.assign(n, 0);
69     pr.assign(n, -1);
70     edges.clear();
71
72     dfsLCA(0, -1);
73
74     floorlog[0] = -1;
75     for (int i = 0; i < stind; ++i) {
76         st[0][i] = i;
77         floorlog[i + 1] = 1 + floorlog[(i + 1) / 2];
78     }
79     for (int l = 0; l + 1 < MAXLOG; ++l)
80         for (int i = 0; i < stind; ++i) {
81             st[l+1][i] = (((i + (1 << l) >= stind) || (hs[st[l]
[ i ]]) <= hs[st[l][i + (1 << l)]])) ? st[l][i] : st[l][i + (1 <<
l)]);
82         }
83     }
84
85     vector<int> compressVector(vector<int> a) {
86         sort(a.begin(), a.end());
87         a.resize(unique(a.begin(), a.end()) - a.begin());
88         return a;
89     }
90
91     int findLCA(int v1, int v2) {
92         v1 = firstPos[v1];
93         v2 = firstPos[v2];
94         if (v1 > v2)
95             swap(v1, v2);
96         int l = floorlog[v2 - v1 + 1];
97         int ans1 = st[l][v1];
98         int ans2 = st[l][v2 - (1 << l) + 1];
99         return ((hs[ans1] < hs[ans2]) ? vs[ans1] : vs[ans2]);
100     }
101
102     void compressAndMarkBase() {
103         base = compressVector(base);
104         for (int v : base)
105             isBased[v] = 1;
106     }
107
108     void findBaseVertices() {
109         base.clear();
110         isBased.assign(n, 0);
111         for (pair<int, int> p : edges) {
112             base.push_back(p.first);
113             base.push_back(p.second);
114         }
115         compressAndMarkBase();
116
117         int kol = base.size();
118         for (int i = 0; i < kol; ++i)
119             for (int j = i + 1; j < kol; ++j)
120                 base.push_back(findLCA(base[i], base[j]));
121
122         compressAndMarkBase();
123
124         baseVerticesNumbers.assign(n, -1);
125         for (int i = 0; i < (int)base.size(); ++i)
126             baseVerticesNumbers[base[i]] = i;
127     }

```

```

128
129 int getBaseNumber(int v) {
130     return lower_bound(base.begin(), base.end(), v) - base.begin();
131 }
132
133 void findBaseEdges() {
134     nn = base.size();
135     ngr.assign(nn, vector<int>());
136     baseEdges.clear();
137     for (int v : base) {
138         int nv = v;
139         do {
140             nv = pr[nv];
141         }
142         while (nv >= 0 && !isBased[nv]);
143
144         if (nv >= 0) {
145             int bv = getBaseNumber(v);
146             int bnv = getBaseNumber(nv);
147             if (bv > bnv)
148                 swap(bv, bnv);
149             baseEdges.push_back(make_pair(bv, bnv));
150         }
151     }
152
153     assert(base.size() == 0 || baseEdges.size() == base.size() -
154 1);
155     sort(baseEdges.begin(), baseEdges.end());
156     for (pair<int, int> p : edges)
157         baseEdges.push_back(make_pair(getBaseNumber(p.first),
158 getBaseNumber(p.second)));
159 }
160
161 void dfsABV(int v) {
162     // cerr << "dfsABV " << v << ' ' << compNumber << "\n";
163     if (isBased[v]) {
164         abv.push_back(v);
165         return;
166     }
167
168     used[v] = compNumber;
169     for (int nv : gr[v])
170         if (used[nv] < 0)
171             dfsABV(nv);
172 }
173
174 void findAchievableBaseVertices() {
175     used.assign(n, -1);
176     compNumber = 0;
177     abv.clear();
178     for (int i = 0; i < n; ++i)
179         if (!isBased[i] && used[i] < 0) {
180             dfsABV(i);
181             ++compNumber;
182             achievableBaseVertices.push_back(abv);
183
184             assert(1 <= abv.size() && abv.size() <= 2);
185             if (abv.size() == 1)
186                 baseEdgeNumbers.push_back(-1);
187             else {
188                 int v1 = getBaseNumber(abv[0]);
189                 int v2 = getBaseNumber(abv[1]);
190                 if (v1 > v2)
191                     swap(v1, v2);

```

```

192         int numEdge = lower_bound(baseEdges.begin(),
baseEdges.begin() + base.size() - 1, make_pair(v1, v2)) -
baseEdges.begin();
193         baseEdgeNumbers.push_back(numEdge);
194     }
195
196     abv.clear();
197 }
198
199     for (int v : base) {
200         achievableBaseVertices.push_back(vector<int>(1, v));
201         used[v] = compNumber++;
202         baseEdgeNumbers.push_back(-1);
203     }
204 }
205
206 void calcDP(int st) {
207     memset(dp, 0, sizeof(dp));
208     dp[1 << st][st] = 1;
209     int nn = base.size();
210     for (int mask = 0; mask < (1 << nn); ++mask)
211         for (int v = 0; v < nn; ++v)
212             if (dp[mask][v])
213                 for (int nv : ngr[v])
214                     if (!(mask & (1 << nv)))
215                         dp[mask | (1 << nv)][nv] += dp[mask][v];
216 }
217
218 void buildNgr(int f1, int f2) {
219     ngr.assign(nn, vector<int>());
220     for (int i = 0; i < (int)baseEdges.size(); ++i)
221         if (i != f1 && i != f2)
222             addEdge(baseEdges[i].first, baseEdges[i].second, ngr);
223 }
224
225 void findAns0() {
226     buildNgr(-1, -1);
227     memset(ans0, 0, sizeof(ans0));
228     for (int v = 0; v < nn; ++v) {
229         calcDP(v);
230         for (int mask = 0; mask < (1 << nn); ++mask)
231             for (int w = 0; w < nn; ++w)
232                 ans0[v][w] += dp[mask][w];
233     }
234 }
235
236 void findAns1() {
237     memset(ans1, 0, sizeof(ans1));
238     for (int e = 0; e < nn - 1; ++e) {
239         buildNgr(e, -1);
240         calcDP(baseEdges[e].first);
241         for (int mask = 0; mask < (1 << nn); ++mask)
242             for (int w = 0; w < nn; ++w)
243                 ans1[e][w] += dp[mask][w];
244         calcDP(baseEdges[e].second);
245         for (int mask = 0; mask < (1 << nn); ++mask)
246             for (int w = 0; w < nn; ++w)
247                 ans1[e][w] += dp[mask][w];
248     }
249 }
250
251 void findAns2() {
252     memset(ans2, 0, sizeof(ans2));
253     for (int e1 = 0; e1 < nn - 1; ++e1)
254         for (int e2 = e1 + 1; e2 < nn - 1; ++e2) {
255             buildNgr(e1, e2);

```

```

256         calcDP(baseEdges[e1].first);
257         for (int mask = 0; mask < (1 << nn); ++mask)
258             ans2[e1][e2] += dp[mask][baseEdges[e2].first] +
dp[mask][baseEdges[e2].second];
259         calcDP(baseEdges[e1].second);
260         for (int mask = 0; mask < (1 << nn); ++mask)
261             ans2[e1][e2] += dp[mask][baseEdges[e2].first] +
dp[mask][baseEdges[e2].second];
262         ans2[e2][e1] = ans2[e1][e2];
263     }
264 }
265
266 void findAllPossibleAnswers() {
267     findAns0();
268     findAns1();
269     findAns2();
270 }
271
272 void performQuery() {
273     int v1, v2;
274     scanf("%d%d", &v1, &v2);
275     --v1;
276     --v2;
277
278     if (n == m + 1) {
279         cout << "1\n";
280         return;
281     }
282
283     if (achievableBaseVertices[used[v1]].size() >
achievableBaseVertices[used[v2]].size())
284         swap(v1, v2);
285
286     vector<int>& abv1 = achievableBaseVertices[used[v1]];
287     vector<int>& abv2 = achievableBaseVertices[used[v2]];
288     vector<int>& bvn = baseVerticesNumbers;
289
290     /*for (int v : abv1)
291         cerr << v << ' ';
292     cerr << "are abv1\n";
293     for (int v : abv2)
294         cerr << v << ' ';
295     cerr << "are abv2\n";
296     */
297
298     if (abv1.size() == 1) {
299         if (abv2.size() == 1)
300             cout << ans0[bvn[abv1[0]]][bvn[abv2[0]]] << "\n";
301         else
302             cout << ans1[baseEdgeNumbers[used[v2]]][bvn[abv1[0]]]
<< "\n";
303     }
304     else {
305         int e1 = baseEdgeNumbers[used[v1]];
306         int e2 = baseEdgeNumbers[used[v2]];
307         if (e1 != e2)
308             cout << ans2[e1][e2] << "\n";
309         else {
310             int b1 = abv1[0];
311             int b2 = abv1[1];
312
313             if (h[b1] > h[b2])
314                 swap(b1, b2);
315             int lca1 = findLCA(b2, v1);
316             int lca2 = findLCA(b2, v2);
317             if (lca1 == lca2)

```

```

318         cout << "1\n";
319     else
320         cout << ans1[baseEdgeNumbers[used[v1]][bvn[b2]] -
1 + 1 << "\n";
321     }
322 }
323 }
324
325 void performQueries() {
326     int q;
327     scanf("%d", &q);
328     while (q--)
329         performQuery();
330 }
331
332 int main() {
333     readData();
334     if (m + 1 > n) {
335         buildLCAAndFindNonTreeEdges();
336         findBaseVertices();
337         findBaseEdges();
338
339         /*for (int v : base)
340             cerr << "BASE VERTEX: " << v << "\n";
341         for (pair<int, int> e : baseEdges)
342             cerr << "BASE EDGE: " << e.first << ' ' << e.second <<
"\n";
343         */
344
345         findAchievableBaseVertices();
346         findAllPossibleAnswers();
347
348     }
349     performQueries();
350     return 0;
351 }
352
353 /*
354 void dfsRebuildGraph(int v, int prv) {
355     used[v] = 1;
356     pr[v] = prv;
357     for (int nv : gr[v]) {
358         if (nv == prv)
359             continue;
360         if (used[nv] == 0) {
361             addEdge(v, nv, ngr);
362             h[nv] = h[v] + 1;
363             dfsRebuildGraph(nv, v);
364         }
365         else if (used[nv] == 1) {
366             int newv = ngr.size();
367             ngr.push_back(vector<int>());
368             h.push_back(h[v] + 1);
369             addEdge(nv, newv, ngr);
370             addEdge(newv, nv, ngr);
371             pr.push_back(v);
372         }
373     }
374     used[nv] = 2;
375 }
376
377 void rebuildGraph() {
378     used.assign(n, 0);
379     pr.assign(n, -1);
380     ngr.assign(n, vector<int>());
381     h.assign(n, 0);

```



```
382     dfsRebuildGraph(0, -1);
383     basen = n;
384     n = ngr.size();
385     gr.swap(ngr);
386 }
387
388 */
389
```