

# Московская олимпиада школьников. Информатика. 9 класс. Отборочный этап, 2023/24

15 дек 2023 г., 00:00 — 17 янв 2024 г., 23:59

100 баллов

## Пробка

Глеб оказался в автомобильной пробке. Применяв зоркий глаз, Глеб насчитал, что перед ним стоят  $n$  машин. Также он знает, что зеленый свет горит ровно  $a$  секунд и за каждую секунду зеленого света с пробки успевают уехать ровно  $b$  машин. Красный свет горит ровно  $c$  секунд.

Глеб хочет узнать, сколько еще секунд он будет стоять в пробке, если прямо сейчас загорится зеленый свет.

## Формат входных данных

Первая строка входных данных содержит одно целое число  $n$  ( $1 \leq n \leq 10^9$ ) — количество машин, стоящих перед Глебом.

Вторая строка входных данных содержит три целых числа  $a, b$  и  $c$  ( $1 \leq a, b, c \leq 10^9$ ).

## Формат выходных данных

Выведите единственное число — сколько секунд придется стоять Глебу в пробке.

Обратите внимание, что ответ в этой задаче может быть довольно большим и не помещаться в 32-битные типы данных. Рекомендуется использовать 64-битный тип данных, например `long long` в C++, `long` в Java или `int64` в Pascal.

## Критерии оценивания

В этой задаче каждый тест, кроме тестов из условия, оценивается независимо.

### Примеры

```
5
4 2 3
```

```
3
```

```
11
4 1 3
```

```
18
```

```
2
2 1 5
```

```
8
```

```
2
1 2 5
```

```
7
```

### Ограничения

Время выполнения: 1 секунда

Память: 256 МВ

Код

C++

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4 #define int long long
5
6 signed main() {
7     int n, a, b, c;
8     cin >> n >> a >> b >> c;
9     int n1 = (n + 1 + b - 1) / b * b;
10    int n2 = n1 / b;
11    int k1 = (n2 - 1) / a * (a + c);
12    cout << k1 + ((n2 - 1) % a + 1);
13 }
14
```

## № 2

100 баллов

### Очевидная задача про подотрезок

Дан массив  $a$  длины  $n$ , а также число  $k$ . Требуется найти подотрезок этого массива минимальной длины, который содержит ровно  $k$  различных значений. Среди всех таких подотрезков выбрать такой, что его левая граница минимальна.

### Формат входных данных

Первая строка содержит два целых числа  $n$  и  $k$  ( $1 \leq k \leq n \leq 2 \cdot 10^5$ ) — количество элементов массива и число  $k$ .

Вторая строка содержит  $n$  целых чисел  $a_1, a_2, \dots, a_n$  ( $0 \leq a_i \leq 10^6$ ) — элементы массива  $a$ .

### Формат выходных данных

Выведите два числа — левую и правую границу искомого подотрезка, или число  $-1$ , если такого отрезка не существует.

### Критерии оценивания

Каждый тест в этой задаче оценивается независимо.

#### Примеры

```
4 2
1 1 2 3
```

```
2 3
```

```
4 3
1 2 2 1
```

```
-1
```

```
6 3
1 1 2 2 3 3
```

```
2 5
```

#### Ограничения

Время выполнения: 2 секунды

Память: 256 МВ

Код

C++

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  #define ll long long
6  #define ld long double
7
8  ld INF = (ld) 1e20;
9  int iINF = (int) 1e20;
10 ll mod = (ll) 1e9 + 7;
11
12 int main() {
13     ll n, k;
14     cin >> n >> k;
15     vector<ll> num(1000001, 0), a(n);
16     for (int i = 0; i < n; ++i) {
17         cin >> a[i];
18     }
19     int ansL = -2, ansR = -2;
20     int now = -1, cnt = 0;
21     for (int i = 0; i < n; ++i) {
22         while (now + 1 < n && cnt < k) {
23             ++now;
24             if (num[a[now]] == 0) {
25                 ++cnt;
26             }
27             ++num[a[now]];
28         }
29         if ((now - i < ansR - ansL || ansL == -2) && cnt == k) {
30             ansL = i, ansR = now;
31         }
32         num[a[i]]--;
33         cnt -= (num[a[i]] == 0);
34     }
35     if (ansL == -2) {
36         cout << -1;
37         return 0;
38     }
39     cout << ansL + 1 << ' ' << ansR + 1;
40 }
41

```

100 баллов

## Интересная лекция

Сегодня Вася побывал на лекции по массивам от одного известного профессора. Лекция была очень интересная, и домашнее задание было не менее занимательным. Звучало оно так:

*Дан массив длины  $n$ . Требуется сделать отрезок максимальной длины из одинаковых элементов, используя максимум один разворот подотрезка массива.*

Васе понравилась эта задача, но сегодня вечером у него работа, поэтому он не успеет сделать домашнее задание. Но вы, как самый лучший друг Васи, согласились помочь ему решить эту задачу.

Напомним, что разворот подотрезка от  $l$  до  $r$  массива  $a_1, \dots, a_{l-1}, a_l, \dots, a_r, a_{r+1}, \dots, a_n$  делает из него массив  $a_1, \dots, a_{l-1}, a_r, a_{r-1}, \dots, a_{l+1}, a_l, a_{r+1}, \dots, a_n$ .

## Формат входных данных

В первой строке дано число  $n$  ( $1 \leq n \leq 2 \cdot 10^5$ ) — количество элементов массива.

Во второй строке даны  $n$  чисел  $a_i$  ( $1 \leq a_i \leq 10^6$ ) — элементы массива.

## Формат выходных данных

Требуется вывести длину самого длинного подотрезка из одинаковых элементов, который получается путем использования максимум одного раза разворота подотрезка массива.

## Критерии оценивания

Все тесты в этой задаче оцениваются независимо.

### Примеры

```
5
1 3 1 3 1
```

```
2
```

```
6
1 1 3 2 2 2
```

```
3
```

```
6
1 1 3 4 1 1
```

```
4
```

```
5
1 2 3 4 5
```

```
1
```

### Ограничения

Время выполнения: 2 секунды

Память: 256 МВ

Код

C++

```

1
2  #include <bits/stdc++.h>
3
4  using namespace std;
5
6  int main() {
7      int n;
8      cin >> n;
9      vector<int> a (n);
10     vector<int> first_max (1000001);
11     vector<int> second_max (1000001);
12     for (int i = 0; i < n; ++i) {
13         cin >> a[i];
14     }
15     int counter = 1;
16     for (int i = 1; i < n; ++i) {
17         if (a[i] == a[i - 1]) ++counter;
18         else {
19             if (counter > first_max[a[i - 1]]) {
20                 second_max[a[i - 1]] = first_max[a[i - 1]];
21                 first_max[a[i - 1]] = counter;
22             } else if (counter > second_max[a[i - 1]]) {
23                 second_max[a[i - 1]] = counter;
24             }
25             counter = 1;
26         }
27     }
28     if (counter > first_max[a[n - 1]]) {
29         second_max[a[n - 1]] = first_max[a[n - 1]];
30         first_max[a[n - 1]] = counter;
31     } else if (counter > second_max[a[n - 1]]) {
32         second_max[a[n - 1]] = counter;
33     }
34
35     int maximum = 0;
36     for (int i = 0; i < 1000001; ++i) {
37         maximum = max(maximum, first_max[i] + second_max[i]);
38     }
39
40     cout << maximum;
41 }
42

```

100 баллов

## Ёжик в матрице

Биологи провели следующий эксперимент.

Поле представляет собой матрицу из  $r$  строк и  $c$  столбцов. Изначально ёж находится на поле в левом верхнем углу, имеющем координаты  $(1, 1)$ . Ёж перемещается из текущего поля или в поле, которое находится справа от него, или в поле, которое находится снизу от него, и финиширует в правом нижнем углу.

В некоторых полях расположено по одной ягоде, на остальных полях ягод нет. Проходя через соответствующее поле, ёж съедает ягоду.

Биологи выяснили, что ёж всегда находит такой путь, который позволяет ему съесть наибольшее количество ягод. Назовём ягоду *ключевой*, если в случае, если эту ягоду (и только её!) убрать с поля, наибольшее количество ягод, которые может съесть ёж, уменьшится.

Ваша задача — по заданному расположению ягод посчитать количество ключевых ягод на поле.

## Формат входных данных

Первая строка входных данных содержит одно целое число  $t$  — количество тестовых примеров ( $1 \leq t \leq 10$ ).

## Критерии оценивания

Решения верно работающие для поля, каждая сторона которого не превосходит 100, будут набирать не менее 40 баллов. Тесты оцениваются независимо.

Каждый тестовый пример имеет следующий формат:

В первой строке тестового примера находятся три натуральных числа  $r$ ,  $c$  и  $b$  — количество строк в матрице, количество столбцов в матрице и количество ягод в матрице, соответственно ( $1 \leq r, c \leq 10^9$ ,  $1 \leq b \leq 2 \cdot 10^5$ ,  $1 \leq b \leq r \cdot c$ ).

Каждая из последующих  $b$  строк содержит два целых числа  $rb_i$  и  $cb_i$  ( $1 \leq rb_i \leq r$ ,  $1 \leq cb_i \leq c$ ) — координаты очередного поля, содержащего ягоду. Гарантируется, что все эти поля попарно различны.

## Формат выходных данных

Для каждого тестового примера в отдельной строке выведите целое число — количество ключевых ягод на поле.

### Примеры

```
1
6 5 5
5 5
2 2
3 3
4 4
1 1
```

5

```
1
4 4 5
2 1
3 1
4 1
1 2
1 3
```

3





```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  vector<int> lis(vector<int>& a) {
6      int n = a.size();
7      vector<int> d(n+1, 2e9);
8
9      vector<int> sol;
10     for (int i = 0; i < n; i++) {
11         int ind = upper_bound(d.begin(), d.end(), a[i]) -
d.begin();
12         if (d[ind] != 2e9){
13             sol.push_back(ind+1);
14             d[ind] = a[i];
15         } else {
16             d[ind] = a[i];
17             sol.push_back(lower_bound(d.begin(),
d.end(), 2e9) - d.begin());
18         }
19     }
20
21     return sol;
22 }
23
24 int main()
25 {
26     ios_base::sync_with_stdio(false);
27     cin.tie(nullptr);
28     cout.tie(nullptr);
29     cerr.tie(nullptr);
30
31     int t;
32     cin >> t;
33
34     for (int e = 0; e < t; ++e){
35         int n, m, p;
36         cin >> n >> m >> p;
37
38         vector<pair<int, int>> a;
39         for (int i = 0; i < p; ++i){
40             int x, y;
41             cin >> x >> y;
42
43             a.push_back({x, y});
44         }
45
46         sort(a.begin(), a.end());
47
48         vector<int> col;
49         for (int i = 0; i < p; ++i){
50             col.push_back(a[i].second);
51         }
52
53         vector<int> pref = lis(col);
54
55         vector<int> rev;
56         for (int i = 0; i < p; ++i){
57             rev.push_back(-1 * a[p-1-i].second);
58         }
59
60         vector<int> suf = lis(rev);

```

```
61         reverse(suf.begin(), suf.end());
62
63         vector<int> candidates;
64         vector<int> cnt(p+1,0);
65
66         int max_lis = *max_element(pref.begin(),
pref.end());
67
68         for (int i = 0; i < p; ++i){
69             if (pref[i] + suf[i] - 1 == max_lis){
70                 candidates.push_back(i);
71                 cnt[pref[i]]++;
72             }
73         }
74
75         int sol = 0;
76         for (int c : candidates){
77             if (cnt[pref[c]] == 1){
78                 ++sol;
79             }
80         }
81
82         cout << sol << '\n';
83     }
84 }
85 }
86 }
```

100 баллов

## Перемести фишку

Фишку можно перемещать по клетчатому полю, состоящему из  $n$  строк и  $m$  столбцов.

В каждой клетке поля записано целое число  $a_{ij}$  ( $1 \leq a_{ij} \leq 10^6$ ). Сначала фишка стоит на поле  $(1, 1)$ . За один ход можно переместить фишку с поля  $(i_1, j_1)$  на поле  $(i_2, j_2)$ , если  $\text{НОД}(a_{i_1 j_1}, a_{i_2 j_2}) > 1$ .

За какое минимальное количество ходов можно переместить фишку с поля  $(1, 1)$  на поле  $(n, m)$ ?

## Формат входных данных

В первой строке записано два целых числа  $n$  и  $m$

( $1 \leq n \leq 5 \cdot 10^5, 1 \leq m \leq 5 \cdot 10^5, 1 \leq n \cdot m \leq 5 \cdot 10^5$ ) — количество строк и столбцов в клетчатом поле.

В каждой из следующих  $n$  строк записано по  $m$  целых чисел  $a_{ij}$  ( $1 \leq a_{ij} \leq 10^6$ ) — значения в клетках.

## Формат выходных данных

Выведите единственное целое число — минимальное количество ходов, чтобы переместить фишку из клетки  $(1, 1)$  в клетку  $(n, m)$ . Если это сделать невозможно, выведите  $-1$ .

## Замечание

Наибольший общий делитель  $\text{НОД}(a, b)$  двух положительных целых чисел  $a$  и  $b$  равняется самому большому целому числу, на которое без остатка делятся оба числа  $a$  и  $b$ .

## Критерии оценивания

В задаче 4 подзадачи. Подзадача 0 — тесты из условия, баллы не начисляются. Подзадачи 1, 2, 3 — баллы начисляются за каждый тест в отдельности.

Подзадача	Баллы	Ограничения
0	0 баллов	Тесты из условия
1	18 баллов	$1 \leq n \cdot m \leq 1000, 1 \leq a_{ij} \leq 10^6$
2	26 баллов	$1 \leq n \cdot m \leq 5 \cdot 10^5, 1 \leq a_{ij} \leq 500$
3	56 баллов	$1 \leq n \cdot m \leq 5 \cdot 10^5, 1 \leq a_{ij} \leq 10^6$

## Примеры

```
1 1
1
```

```
0
```

```
3 1
1
2
4
```

```
-1
```

```
3 3
6 9 5
8 6 21
77 14 11
```

```
3
```

## Ограничения

Время выполнения: 2 секунды

Код

C++

```

1  #define _USE_MATH_DEFINES
2  #include<iostream>
3  #include<fstream>
4  #include<string>
5  #include<vector>
6  #include<utility>
7  #include<algorithm>
8  #include<climits>
9  #include<set>
10 #include<map>
11 #include<cmath>
12 #include<iomanip>
13 #include<iterator>
14 #include<queue>
15 #include<stack>
16 #include<cctype>
17 #include<deque>
18 #include<time.h>
19 #include<bitset>
20 #include<random>
21 #include<functional>
22 #include<unordered_set>
23 #include<unordered_map>
24 #include<random>
25 #include<numeric>
26 #include<sstream>
27 #include<cassert>
28 #include<chrono>
29 #include<complex>
30
31 //by Skeef79
32
33 typedef long long ll;
34 typedef long double ld;
35 typedef unsigned long long ull;
36
37 #pragma warning(disable : 4996)
38 #pragma comment(linker, "/STACK:16777216")
39 #define pb push_back
40 #define en '\n'
41 #define forn(i,n) for(int i = 0;i<n;i++)
42 #define all(x) (x).begin(),(x).end()
43 #define rall(x) (x).rbegin(),(x).rend()
44 #define vec vector
45 #define pii pair<int,int>
46 #define pll pair<ll,ll>
47 #define szof(x) int(x.size())
48
49 using namespace std;
50
51 const int INF = 1000000000 + 1e8;
52 const ll LINF = 2000000000000000000;
53
54
55 const int N = 1e6 + 1010;
56 int lp[N];
57
58 void sieve() {
59     for (int i = 0; i < N; i++)
60         lp[i] = i;
61     for (int i = 2; i < N; i++) {
62         if (lp[i] == i) {

```

```

63         if (i > 2000)
64             break;
65         for (int j = i * i; j < N; j += i)
66             lp[j] = min(lp[j], lp[i]);
67     }
68 }
69 }
70
71 vec<int> factors[N];
72 vec<pii> points[N];
73 bool used[N];
74
75 void factorize() {
76
77     sieve();
78     vec<int> cur;
79     for (int i = 2; i < N; i++) {
80         int x = i;
81         cur.clear();
82         while (x != lp[x]) {
83             cur.pb(lp[x]);
84             x /= lp[x];
85         }
86
87         cur.pb(lp[x]);
88         sort(all(cur));
89         cur.resize(unique(all(cur)) - cur.begin());
90         factors[i] = cur;
91     }
92 }
93
94 void solve() {
95     int n, m;
96     cin >> n >> m;
97
98     factorize();
99     vec<vec<int>> a(n, vec<int>(m));
100    for (int i = 0; i < n; i++)
101        for (int j = 0; j < m; j++)
102            cin >> a[i][j];
103
104    for (int i = 0; i < n; i++) {
105        for (int j = 0; j < m; j++) {
106            for (auto p : factors[a[i][j]]) {
107                points[p].pb({ i, j });
108            }
109        }
110    }
111
112    vec<vec<int>> d(n, vec<int>(m, INF));
113    d[0][0] = 0;
114    queue<pii> q;
115    q.push({ 0, 0 });
116
117    while (!q.empty()) {
118        auto [i, j] = q.front();
119        q.pop();
120
121        for (auto p : factors[a[i][j]]) {
122            if (!used[p]) {
123                for (auto [ni, nj] : points[p]) {
124                    if (d[ni][nj] > d[i][j] +

```

1) {

```

125         d[ni][nj] = d[i][j]
126     + 1;
127         q.push({ ni,nj });
128     }
129     used[p] = true;
130 }
131 }
132 }
133
134     if (d[n - 1][m - 1] == INF)
135         cout << -1;
136     else
137         cout << d[n - 1][m - 1];
138 }
139 }
140
141 int main() {
142     srand(time(0));
143     ios::sync_with_stdio(false);
144     cin.tie(NULL);
145     cout.tie(NULL);
146
147     #ifdef _DEBUG
148         freopen("input.txt", "r", stdin);
149     #endif
150
151     int tst = 1;
152     //cin >> tst;
153
154     while (tst--) {
155         solve();
156     }
157
158 }
159

```



100 баллов

## Гиревой спорт

В школьном тренажерном зале оказалось  $n$  различных гирь, разложенных в некотором порядке так, что они образовали массив  $a$ , где значение  $a_i$  равно удовольствию, которое можно получить при поднятии гири с номером  $i$  (удовольствие также может быть отрицательным, например, если гиря задела самолюбие).

Спортсмен хочет выбрать некоторый подотрезок гирь и поднять их. Удовольствие, которое он получит, будет равно сумме удовольствий гирь на этом подотрезке. Все было бы просто, но при поднятии каждой  $k$ -й гири из выбранного диапазона спортсмен теряет  $x$  единиц удовольствия.

Например, если  $k = 2$ , а спортсмен поднял 5 гирь, то он получит удовольствие равное сумме удовольствий от поднятия выбранных гирь минус  $2 \cdot x$ .

Найдите такой подотрезок, чтобы удовольствие, которое получит спортсмен, будет максимально, и выведите величину этого удовольствия.

Можно не поднимать ни одной гири, если это будет оптимально.

## Формат входных данных

Первая строка содержит три целых числа  $n$ ,  $k$  и  $x$  ( $1 \leq n \leq 2 \cdot 10^5$ ,  $1 \leq k \leq 5$ ,  $1 \leq x \leq 10^4$ ) — количество элементов массива, а также числа  $k$  и  $x$ .

Вторая строка содержит  $n$  целых чисел  $a_1, a_2, \dots, a_n$  ( $-10^4 \leq a_i \leq 10^4$ ) — элементы массива  $a$ .

## Формат выходных данных

Выведите одно число — максимальное удовольствие, которое можно получить при описанных правилах.

## Пояснения к примерам

В первом примере оптимально взять отрезок с третьего по шестой элемент, тогда спортсмен получит  $14 - 2 + 9 + 4 = 25$  единиц удовольствия от гирь и потеряет 8 единиц удовольствия от третьей гири.

Во втором примере оптимально не брать ни одну гирю.

В третьем примере можно взять либо отрезок с первого по второй элемент, либо взять только четвертый элемент.

## Критерии оценивания

Все тесты в этой задаче оцениваются независимо.

### Примеры

```
6 3 8
1 -3 14 -2 9 4
```

```
17
```

```
5 5 5
-1 -2 -3 -4 -10
```

```
0
```

```
4 2 5
6 6 -9 7
```

```
7
```

### Ограничения

Время выполнения: 2 секунды

Код

C++

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  #define ll long long
6  #define ld long double
7
8  ld INF = (ld) 1e20;
9  int iINF = (int) 1e20;
10 ll mod = (ll) 1e9 + 7;
11
12 int main() {
13     ll n, y, x;
14     cin >> n >> y >> x;
15     vector<ll> a(n);
16     ll maxim[n], sum[n], dp[n];
17     for (int i = 0; i < n; ++i) {
18         cin >> a[i];
19         maxim[i] = 0;
20         sum[i] = 0;
21         dp[i] = 0;
22     }
23     for (int i = 0; i < n; ++i) {
24         ll now = 0;
25         for (int j = 0; j < y - 1; ++j) {
26             if (i + j >= n)
27                 break;
28             now += a[i + j];
29             if (now > maxim[i]) {
30                 maxim[i] = now;
31             }
32         }
33         if (i + y - 1 < n) {
34             now += a[i + y - 1];
35         }
36         sum[i] = now;
37     }
38     ll ans = 0;
39     for (int i = n - 1; i >= 0; --i) {
40         dp[i] = max(dp[i], maxim[i]);
41         if (i + y < n) {
42             dp[i] = max(dp[i], sum[i] + dp[i + y] - x);
43         }
44         ans = max(ans, dp[i]);
45     }
46     cout << max(ans, (ll) 0);
47 }
48
```

## Каждой твари по паре

Коллекционировать животных можно не только на Ноевом ковчеге, но и на дереве. А дерево, как известно из теории графов, является связным ациклическим графом. Иными словами, дерево состоит из  $n$  вершин и  $n - 1$  ребра между ними. Каждое ребро соединяет некоторые две вершины дерева. Из любой вершины дерева можно добраться по ребрам до любой другой. Также дерево не содержит циклов, т. е. нельзя построить путь по ребрам из некоторой вершины и вернуться в нее, не пройдя по одному и тому же ребру дважды.

Для расположения животных на нашем дереве сделано  $q$  гнезд, где  $q$  — четное число. Животным каждого вида предназначены два гнезда.

После расселения по дереву каждая пара одного вида может перемещаться по ребрам дерева между своими гнездами. Они не должны мешать другим видам, т. е. чтобы никакие два пути не имели общего ребра. При этом допускается пересечение путей в вершине. Вам необходимо разместить  $\frac{q}{2}$  видов животных по гнездам таким образом, чтобы пути животных одного вида удовлетворяли этому условию.

## Формат входных данных

В первой строке входных данных находится два целых числа  $n$  и  $q$  ( $2 \leq n \leq 2 \cdot 10^5$ ,  $2 \leq q \leq n$ ,  $q$  четно) — общее количество вершин в дереве и количество вершин, в которых расположены гнезда.

Во второй строке входных данных находятся  $q$  различных целых чисел  $m_i$  ( $1 \leq m_i \leq n$ ) — номера вершин с гнездами.

В каждой из следующих  $n - 1$  строк входных данных находится два целых  $u_i$  и  $v_i$  ( $1 \leq u_i, v_i \leq n$ ,  $u_i \neq v_i$ ). Это означает, что между вершинами  $u_i$  и  $v_i$  в дереве проведено ребро.

Гарантируется, что ребра во входных данных образуют дерево.

## Формат выходных данных

Выходные данные должны содержать  $\frac{q}{2}$  строк.

В каждой строке должно содержаться два целых числа  $x_i$  и  $y_i$  ( $1 \leq x_i, y_i \leq n$ ) — номера вершин, в которых будут располагаться гнезда  $i$ -го вида животных.

Все числа в выходных данных должны быть различны и должны представлять собой номера вершин с гнездами.

Если разместить животных требуемым образом невозможно, выведите число  $-1$ .

## Критерии оценивания

Баллы начисляются за каждый тест в отдельности, но ряд тестов имеют специфическую структуру:

- $u_i = i, v_i = i + 1$
- $u_i = 1, v_i = i + 1$
- $n \leq 3000, q \leq 6$
- Нет дополнительных ограничений.

### Примеры

```
9 6
1 3 2 5 8 6
3 1
3 2
3 4
3 5
3 6
4 7
```

7 8  
1 9

3 2  
8 5  
1 6

**Ограничения**

Время выполнения: 1 секунда

Память: 256 МВ

Код

C++

```

1  #include <vector>
2  #include <array>
3  #include <iostream>
4  using namespace std;
5  const int N = 400100;
6  vector<int> g[N];
7  vector<array<int,2>> pairs;
8  int n, q, paired[N];
9
10 void make_paired(int v, int p) {
11     for (int u : g[v]) {
12         if (u == p) continue;
13
14         make_paired(u, v);
15
16         if (paired[u] >= 0) {
17             if (paired[v] >= 0) {
18                 pairs.push_back({paired[v], paired[u]});
19                 paired[v] = -1;
20                 paired[u] = -1;
21             } else {
22                 paired[v] = paired[u];
23                 paired[u] = -1;
24             }
25         }
26     }
27 }
28
29 int main() {
30     ios_base::sync_with_stdio(0); cin.tie(0);
31
32     cin >> n >> q;
33     fill(paired, paired + n, -1);
34     for (int i = 0; i < q; i++) {
35         int v; cin >> v;
36         v--;
37         paired[v] = v;
38     }
39     for (int i = 1; i < n; i++) {
40         int x, y; cin >> x >> y;
41         x--; y--;
42         g[x].push_back(y);
43         g[y].push_back(x);
44     }
45
46     make_paired(0, -1);
47
48     for (auto pair : pairs) {
49         cout << pair[0] + 1 << " " << pair[1] + 1 << '\n';
50     }
51
52     return 0;
53 }
54

```

## Железнодорожные диаметры

Железная дорога в городе  $M$  состоит из  $n$  станций, соединённых  $n - 1$  перегонами с двусторонним движением так, что между любыми двумя станциями можно проехать по одному или нескольким перегонам.

Любые две станции, кратчайший путь между которыми по сети перегонов не меньше кратчайшего пути между любыми двумя другими станциями, считаются противоположными станциями, а количество перегонов между такими станциями называется диаметром железной дороги.

Министерство транспорта запланировало модернизацию железной дороги. В рамках модернизации можно не более  $k$  раз выполнить следующее действие: закрыть один из перегонов и построить вместо него новый перегон между существующими станциями, не соединёнными перегонном, так, чтобы попрежнему между любыми двумя станциями можно было проехать по одному или нескольким перегонам.

Целью модернизации является максимизация диаметра получившейся в итоге железной дороги. Ваша задача — найти количество перегонов в таком диаметре.

## Формат входных данных

Первая строка входных данных содержит целое число  $t$  ( $1 \leq t \leq 10$ ) — количество сценариев модернизации.

Первая строка каждого сценария содержит два целых числа  $n$  и  $k$  ( $2 \leq n \leq 3000$ ,  $0 \leq k \leq n$ ) — количество станций и максимальное разрешённое количество замен, соответственно. Каждая из последующих  $n - 1$  строк содержит по два целых числа  $i$  и  $j$  ( $1 \leq i, j \leq n$ ,  $i \neq j$ ) — номера станций, соединённых очередным перегонном. Гарантируется, что между любыми двумя станциями можно проехать по одному или нескольким перегонам.

## Формат выходных данных

Для каждого сценария выведите одно целое число — максимальное количество перегонов в диаметре, которое получится после завершения модернизации.

## Критерии оценивания

Баллы начисляются за каждый тест в отдельности, но ряд тестов имеют специфическую структуру:

1.  $u_i = i$ ,  $v_i = i + 1$
2.  $u_i = 1$ ,  $v_i = i + 1$
3.  $n \leq 3000$ ,  $q \leq 6$
4.  $n \leq 3000$ ,  $q = n$
5.  $q = n$
6. Нет дополнительных ограничений.

### Примеры

```
3
5 0
4 1
4 3
2 4
2 5
5 1
1 3
4 3
2 3
5 2
2 1
1 2
```

```
3
4
1
```

### Ограничения

Время выполнения: 2 секунды

Память: 256 МВ

Код

C++

```

1  #include<bits/stdc++.h>
2  #define maxn 3005
3  using namespace std;
4  int t;
5  int n,k;
6  vector<int> a[maxn];
7  int dp[maxn][maxn][3];
8  int gp[maxn][3];
9  int sz[maxn];
10 void dfs(int u,int pr=-1) {
11     sz[u]=1;
12     dp[u][0][0]=dp[u][0][1]=dp[u][0][2]=0;
13     dp[u][1][0]=-1e9;
14     dp[u][1][1]=0;
15     dp[u][1][2]=0;
16     for(auto v:a[u]) {
17         if(v!=pr) {
18             dfs(v,u);
19
20             for(int i=sz[u]+1;i<=sz[u]+sz[v] && i<=k+1;i++) {
21                 dp[u][i][0]=dp[u][i][1]=dp[u][i][2]=0;
22             }
23             dp[u][sz[u]+sz[v]][0]=-1e9;
24             for(int i=0;i<=sz[u]+sz[v] && i<=k+1;i++) {
25                 gp[i][0]=gp[i][1]=gp[i][2]=0;
26             }
27             gp[sz[u]+sz[v]][0]=-1e9;
28
29             for(int i=0;i<=sz[u] && i<=k+1;i++) {
30                 for(int j=0;j<=sz[v] && i+j<=k+1;j++) {
31                     gp[i+j][0]=max(gp[i+j][0],dp[u][i][0]+dp[v][j]
32 [2]);
33                     if(i) {
34                         gp[i+j][1]=max(gp[i+j][1],dp[u][i][1]+dp[v]
35 [j][2]);
36                     }
37                     if(j) {
38                         gp[i+j][1]=max(gp[i+j][1],dp[u][i][0]+dp[v]
39 [j][1]+1);
40                     }
41                     if(i>0 && j>0) {
42                         gp[i+j-1][2]=max(gp[i+j-1][2],dp[u][i]
43 [1]+dp[v][j][1]+1);
44                     }
45                     gp[i+j][2]=max(gp[i+j][2],dp[u][i][2]+dp[v][j]
46 [2]);
47                 }
48             }
49
50             for(int i=0;i<=sz[u]+sz[v] && i<=k+1;i++) {
51                 for(int j=0;j<=2;j++) {
52                     dp[u][i][j]=max(dp[u][i][j],gp[i][j]);
53                 }
54             }
55             sz[u]+=sz[v];
56         }
57     }
58     for(int i=0;i<=sz[u];i++) {
59         dp[u][i][2]=max(dp[u][i][0],max(dp[u][i][1],dp[u][i][2]));

```



```

58     }
59     /*cout<<u<<" "<<sz[u]<<endl;
60     for(int i=0;i<=sz[u];i++) {
61         cout<<"dp["<<u<<"]["<<i<<"][0]="<<dp[u][i][0]<<"; "<<"dp["
<<u<<"]["<<i<<"][1]="<<dp[u][i][1]<<"; "<<"dp["<<u<<"]["<<i<<"]
[2]="<<dp[u][i][2]<<"; "<<endl;
62     }*/
63 }
64 int main() {
65
66     scanf("%d",&t);
67     while(t-->0) {
68         scanf("%d %d",&n,&k);
69         k=min(k+1,n);
70         for(int i=0;i<n-1;i++) {
71             int u,v;
72             scanf("%d %d",&u,&v);
73             a[u].push_back(v);
74             a[v].push_back(u);
75         }
76
77         dfs(1);
78
79         int ans=0;
80         for(int i=0;i<=k;i++) {
81             for(int j=0;j<=2;j++) {
82                 ans=max(ans,dp[1][i][j]+max(i-1,0));
83             }
84         }
85
86         printf("%d\n",ans);
87
88         for(int i=1;i<=n;i++) a[i].clear();
89     }
90     return 0;
91 }
92

```