**Test - 1 (Final)**

You are given a path on an infinite 2d lattice. The path consists of line segments parallel to either the x or y axis. VVA

Example : 1, 1 to 1, 3 to 6, 3 to 6, 6 to bla bla

Also There is a set of points given . You have to find how many of the given points lie on the path. The input was as follows

N M as two variables

There were n numbers that denoted x coordinates of n points

There were n numbers again which denoted y coordinates of those n points

So total there is the info of n points in space

Now there are two more arrays of size m , one consisting of x coordinates and other y coordinates correspondingly

These points are the turning points in a path.

Examples.

Say i went from 1, 1 to 1, 5 to 2, 5 so the given arrays will be

1, 1, 2

1, 5, 5

In the given path , we have to find how many of the given n points lie.

**testcases:**

**ip:**

**5 3**

**-2 0**

**2 0**

**2 2**

**0 2**

**0 -2**

**0 0**

**1 1**

**2 2**

**op:**

**2**

**ip:**

**14 7**

**-3 0**

**6 0**

**6 5**

**-1 5**

**-1 -7**

**0 -7**

**0 -10**

**-1 -10**

**-1 -15**

**2 -15**

**2 -20**

**-5 -20**

**-5 -25**

**0 -25**

**-1 0**

**3 4**

**2 -22**

**1 -19**

**-5 -17**

**-1 -17**

**0 -12**

*// Sarthak stalk krna buri baat hoti hai*

*#include <iostream> // for input and output streams*

*#include <vector> // for std::vector*

*#include <algorithm> // for algorithms like sort, max, min*

*#include <string> // for std::string*

*#include <map> // for std::map*

*#include <unordered\_map> // for std::unordered\_map*

*#include <set> // for std::set*

*#include <unordered\_set> // for std::unordered\_set*

*#include <queue> // for std::queue, std::priority\_queue*

*#include <stack> // for std::stack*

*#include <deque> // for std::deque*

*#include <list> // for std::list*

*#include <iterator> // for std::iterators*

*#include <functional> // for std::function, std::bind*

*#include <numeric> // for numeric algorithms like accumulate*

*#include <cmath> // for mathematical functions like sqrt, pow*

*#include <limits> // for limits of data types*

*#include <iomanip> // for std::setprecision, std::fixed*

*#include <fstream> // for file I/O*

*#include <sstream> // for string streams*

*#include <bitset> // for bitset operations*

*#include <tuple> // for std::tuple*

*#include <utility> // for std::pair, std::move, std::forward*

*#include <ctime> // for std::time*

*#include <chrono> // for time utilities*

*#include <cassert> // for assertions*

*#include <random> // for random number generation*

*#include <memory> // for smart pointers*

*using namespace std;*

*typedef long long ll;*

*typedef long double ld;*

*#define fastio \*

*ios\_base::sync\_with\_stdio(false); \*

*cin.tie(NULL); \*

*cout.tie(NULL)*

*#define max3(a, b, c) max(max(a, b), c)*

*#define max4(a, b, c, d) max(max(a, b), max(c, d))*

*#define fr(i, n) for (ll i = 0; i < n; i++)*

*ll gcd(ll a, ll b)*

*{*

*return b == 0 ? a : gcd(b, a % b);*

*}*

*int main()*

*{*

*fastio;*

*ll n, m;*

*cin >> n >> m;*

*vector<pair<ll, ll>> v(n);*

*for (auto &i : v)*

*cin >> i.first >> i.second;*

*vector<pair<ll, ll>> pts(m);*

*for (auto &i : pts)*

*cin >> i.first >> i.second;*

*map<ll, vector<ll>> abscissa, ordinate;*

*for (auto &i : pts)*

*{*

*abscissa[i.first].push\_back(i.second);*

*ordinate[i.second].push\_back(i.first);*

*}*

*set<pair<ll, ll>> st;*

*for (ll i = 1; i < n - 1; i++)*

*{*

*st.insert(v[i]);*

*}*

*for (auto &i : abscissa)*

*{*

*sort(i.second.begin(), i.second.end());*

*}*

*for (auto &i : ordinate)*

*{*

*sort(i.second.begin(), i.second.end());*

*}*

*set<pair<ll, ll>> res;*

*for (ll i = 0; i < n - 1; i++)*

*{*

*pair<ll, ll> st = v[i];*

*pair<ll, ll> ed = v[i + 1];*

*if (v[i].first == v[i + 1].first)*

*{*

*ll lg = max(st.second, ed.second);*

*ll sm = min(st.second, ed.second);*

*ll ulim = upper\_bound(abscissa[v[i].first].begin(), abscissa[v[i].first].end(), lg) - abscissa[v[i].first].begin() - 1;*

*ll llim = lower\_bound(abscissa[v[i].first].begin(), abscissa[v[i].first].end(), sm) - abscissa[v[i].first].begin();*

*for (ll it = llim; it <= ulim; it++)*

*{*

*res.insert({v[i].first, abscissa[v[i].first][it]});*

*}*

*}*

*else*

*{*

*ll lg = max(ed.first, st.first);*

*ll sm = min(ed.first, st.first);*

*ll ulim = upper\_bound(ordinate[v[i].second].begin(), ordinate[v[i].second].end(), lg) - ordinate[v[i].second].begin() - 1;*

*ll llim = lower\_bound(ordinate[v[i].second].begin(), ordinate[v[i].second].end(), sm) - ordinate[v[i].second].begin();*

*for (ll it = llim; it <= ulim; it++)*

*{*

*res.insert({ordinate[v[i].second][it], v[i].second});*

*}*

*}*

*}*

*cout << res.size() << "\n";*

*}*

**Test - 2 (Final)**

You are in charge of maintaining inventory for a warehouse. You have an initial stock of goods given by an array A of size N.

Your day starts off by getting an inflow of goods given by another array B of size N (so the stock becomes A[i] = A[i]+B[i] for all i).

After that, you can choose any one good, and export it, making its stock 0. Before you leave, you have to report the total stock of all items in the warehouse to the head quarters.

Your task is to find the minimum number of days required to make the total stock of all items <= K, where K is given.

If this is not possible, return -1.

#include <bits/stdc++.h>

using namespace std;

using namespace chrono;

int fn(int *i*, int *day*, vector<pair<int, int>> &*v*, int *n*, int *m*, int *lim*, vector<vector<int>> &*dp*)

{

if (*i* == n)

return 0;

if (*day* > lim)

return 0;

if (dp[*i*][*day*] != -1)

return dp[*i*][*day*];

int take = v[*i*].second + (v[*i*].first) \* *day* + fn(*i* + 1, *day* + 1, v, n, m, lim, dp);

int nottake = fn(*i* + 1, *day*, v, n, m, lim, dp);

return dp[*i*][*day*] = max(take, nottake);

}

void solve(int *t*)

{

int n, m;

cin >> n >> m;

int sum = 0;

vector<int> a(n), b(n);

vector<pair<int, int>> v(n);

int sa = 0, sb = 0;

for (int i = 0; i < n; i++)

{

cin >> a[i];

sa += a[i];

}

for (int i = 0; i < n; i++)

{

cin >> b[i];

sb += b[i];

}

for (int i = 0; i < n; i++)

{

v[i] = {b[i], a[i]};

}

sort(v.begin(), v.end());

int res = -1;

for (int day = 0; day <= n; day++)

{

vector<vector<int>> dp(n + 1, vector<int>(n + 1, -1));

int term = fn(0, 1, v, n, m, day, dp);

int rem = sa + day \* sb - term;

if (rem <= m)

{

res = day;

break;

}

}

cout << res << "\n";

}

int main()

{

auto start = high\_resolution\_clock::now(); *// Start time measurement*

int t;

cin >> t;

for (int i = 1; i <= t; i++)

solve(i);

auto end = high\_resolution\_clock::now(); *// End time measurement*

duration<double> elapsed = end - start;

cout << "Total execution time: " << elapsed.count() << " seconds\n";

return 0;

}

**Test - 3**

You will be given a necklace having only red and blue stones in it. Your task is to make the number of blue and red stones equal.

Stones can only be removed from either the left or right end.

Return the minimum number of stones that are to be removed to complete the given task.

For eg, input: BBRRBRBRBRBBR

output: 1

// try this

Length of string - ([longest length of subarray with equal no of “A” and ‘B’](https://leetcode.com/problems/contiguous-array/description/))

*// Sarthak stalk krna buri baat hoti hai*

#include <iostream> *// for input and output streams*

#include <vector> *// for std::vector*

#include <algorithm> *// for algorithms like sort, max, min*

#include <string> *// for std::string*

#include <map> *// for std::map*

#include <unordered\_map> *// for std::unordered\_map*

#include <set> *// for std::set*

#include <unordered\_set> *// for std::unordered\_set*

#include <queue> *// for std::queue, std::priority\_queue*

#include <stack> *// for std::stack*

#include <deque> *// for std::deque*

#include <list> *// for std::list*

#include <iterator> *// for std::iterators*

#include <functional> *// for std::function, std::bind*

#include <numeric> *// for numeric algorithms like accumulate*

#include <cmath> *// for mathematical functions like sqrt, pow*

#include <limits> *// for limits of data types*

#include <iomanip> *// for std::setprecision, std::fixed*

#include <fstream> *// for file I/O*

#include <sstream> *// for string streams*

#include <bitset> *// for bitset operations*

#include <tuple> *// for std::tuple*

#include <utility> *// for std::pair, std::move, std::forward*

#include <ctime> *// for std::time*

#include <chrono> *// for time utilities*

#include <cassert> *// for assertions*

#include <random> *// for random number generation*

#include <memory> *// for smart pointers*

#include <limits.h>

using namespace std;

typedef long long ll;

typedef long double ld;

#define fastio \

ios\_base::sync\_with\_stdio(false); \

cin.tie(NULL); \

cout.tie(NULL)

#define max3(*a*, *b*, *c*) max(max(a, b), c)

#define max4(*a*, *b*, *c*, *d*) max(max(a, b), max(c, d))

#define fr(*i*, *n*) for (ll i = 0; i < n; i++)

ll gcd(ll *a*, ll *b*)

{

return *b* == 0 ? *a* : gcd(*b*, *a* % *b*);

}

int main()

{

fastio;

ll t = 1;

while (t--)

{

string s;

cin >> s;

ll n = s.length();

vector<ll> a(n + 1, 0);

for (ll i = 0; i < n; i++)

{

a[i + 1] = s[i] == 'B' ? 1 : -1;

}

for (ll i = 1; i <= n; i++)

{

a[i] += a[i - 1];

}

map<ll, ll> mp;

ll res = n;

for (ll i = n; i >= 0; i--)

{

if (mp.find(a[i]) != mp.end())

{

ll term = n - (mp[a[i]] - i);

res = min(res, term);

}

mp[a[i]] = max(mp[a[i]], i);

}

cout << res << "\n";

}

}

**Test - 4**

Q1. There is 2d Matrix of size h\*w representing the city.

Each cell can be 0,1,2,3,4

0-> road

1->Tree

2-> Garage

3-> warehouse

4-> Airport

There is a truck parked at the Garage, Truck’s task is to go to the warehouse (one or many at a time) , load the goods and unload the truck at the airport.

There is no limit on the number of goods a truck can carry.

There is a cost associated with a truck.

Cost is (Number of blocks it has moved \* (1+Number of goods truck carries)) like

To move one block cost is 1 on an empty truck in

Cost is 2 if truck has 1 good

3 if the truck has 2 goods.

……….

You have to tell how many maximum goods can be unloaded at airport using at max C cost

Constraints

Number of test cases - 50

h,w belongs to (2,40)

c belongs to (5,2000)

There can be at max 13 warehouses

Truck cannot pass a tree, if it is at any warehouse truck can or cannot load the goods, similarly if it's at an airport it's not necessary to unload the goods.

But starting point is fixed ie garage.

#include <iostream>

#include <vector>

#include <algorithm>

#include <string>

#include <map>

#include <unordered\_map>

#include <set>

#include <unordered\_set>

#include <queue>

#include <stack>

#include <deque>

#include <list>

#include <iterator>

#include <functional>

#include <numeric>

#include <cmath>

#include <limits>

#include <iomanip>

#include <fstream>

#include <sstream>

#include <bitset>

#include <tuple>

#include <utility>

#include <ctime>

#include <chrono>

#include <cassert>

#include <random>

#include <memory>

#include <limits.h>

using namespace std;

typedef long long ll;

typedef long double ld;

#define fastio \

ios\_base::sync\_with\_stdio(false); \

cin.tie(NULL); \

cout.tie(NULL)

#define max3(*a*, *b*, *c*) max(max(a, b), c)

#define max4(*a*, *b*, *c*, *d*) max(max(a, b), max(c, d))

#define fr(*i*, *n*) for (ll i = 0; i < n; i++)

ll gcd(ll *a*, ll *b*)

{

return *b* == 0 ? *a* : gcd(*b*, *a* % *b*);

}

ll dx[] = {0, 0, 1, -1};

ll dy[] = {1, -1, 0, 0};

bool issafe(ll *x*, ll *y*, ll *n*, ll *m*)

{

return ((0 <= *x*) && (*x* < *n*)) && ((0 <= *y*) && (*y* < *m*));

}

int main()

{

fastio;

ll t = 1;

cin >> t;

while (t--)

{

ll n, m, cost;

cin >> n >> m >> cost;

vector<vector<ll>> v(n, vector<ll>(m));

for (auto &i : v)

{

for (auto &j : i)

{

cin >> j;

}

}

vector<vector<vector<ll>>> dist(n, vector<vector<ll>>(m, vector<ll>(2001, 1e9)));

ll gx, gy, ax, ay;

for (ll i = 0; i < n; i++)

{

for (ll j = 0; j < m; j++)

{

if (v[i][j] == 2)

{

gx = i;

gy = j;

}

if (v[i][j] == 4)

{

ax = i;

ay = j;

}

}

}

dist[gx][gy][0] = 0;

priority\_queue<vector<ll>, vector<vector<ll>>, greater<vector<ll>>> pq;

pq.push({0, gx, gy, 0});

while (pq.size() > 0)

{

auto z = pq.top();

pq.pop();

ll dis = z[0];

ll x = z[1];

ll y = z[2];

ll goods = z[3];

if (dist[x][y][goods] < dis)

continue;

if (((v[x][y] == 0) || (v[x][y] == 2)) || (v[x][y] == 4))

{

for (ll i = 0; i < 4; i++)

{

ll nx = x + dx[i];

ll ny = y + dy[i];

if (issafe(nx, ny, n, m) && (v[nx][ny] != 1))

{

if ((dist[nx][ny][goods] > (dis + goods + 1)) && ((dis + goods + 1) <= cost))

{

dist[nx][ny][goods] = dis + goods + 1;

pq.push({dist[nx][ny][goods], nx, ny, goods});

}

}

}

}

else if (v[x][y] == 3)

{

for (ll i = 0; i < 4; i++)

{

ll nx = x + dx[i];

ll ny = y + dy[i];

if (issafe(nx, ny, n, m) && (v[nx][ny] != 1))

{

for (ll j = goods; j <= 2000; j++)

{

if ((dist[nx][ny][j] > (dis + j + 1)) && ((dis + j + 1) <= cost))

{

dist[nx][ny][j] = dis + j + 1;

pq.push({dist[nx][ny][j], nx, ny, j});

}

}

}

}

}

}

ll res = 0;

for (ll i = 0; i <= 2000; i++)

{

if (dist[ax][ay][i] <= cost)

{

res = i;

}

}

cout << res << "\n";

}

}

testcases:(check for correctness)

ip:

3

2 6 30

2 0 0 3 0 0

0 0 0 3 0 4

1 5 5

2 0 0 3 4

2 6 30

2 0 0 3 0 0

0 0 0 1 0 4

op:

12

1

8

Q2. Given N tiles of given width and height, we have to select K out of it, we need to minimise the maximum of the difference between any two tiles selected, the difference between any two tiles is defined as the maximum of the height difference and width difference.

#include <iostream>

#include <vector>

#include <algorithm>

#include <string>

#include <map>

#include <unordered\_map>

#include <set>

#include <unordered\_set>

#include <queue>

#include <stack>

#include <deque>

#include <list>

#include <iterator>

#include <functional>

#include <numeric>

#include <cmath>

#include <limits>

#include <iomanip>

#include <fstream>

#include <sstream>

#include <bitset>

#include <tuple>

#include <utility>

#include <ctime>

#include <chrono>

#include <cassert>

#include <random>

#include <memory>

#include <limits.h>

using namespace std;

typedef long long ll;

typedef long double ld;

#define fastio \

ios\_base::sync\_with\_stdio(false); \

cin.tie(NULL); \

cout.tie(NULL)

#define max3(a, b, c) max(max(a, b), c)

#define max4(a, b, c, d) max(max(a, b), max(c, d))

#define fr(i, n) for (ll i = 0; i < n; i++)

ll gcd(ll a, ll b)

{

return b == 0 ? a : gcd(b, a % b);

}

bool checker(vector<pair<ll, ll>> &tiles, ll n, ll k, ll mid)

{

vector<ll> width(n), height(n);

for (ll i = 0; i < n; i++)

{

width[i] = tiles[i].first;

height[i] = tiles[i].second;

}

for (ll i = 0; i < n; i++)

{

ll j;

for (j = i; j < n; j++)

{

if ((width[j] - width[i]) > mid)

{

break;

}

}

if ((width[j] - width[i]) > mid)

j--;

if(j==n) j--;

if(j-i+1<k)

continue;

vector<ll> temp;

for (ll z = i; z <= j; z++)

{

temp.push\_back(height[z]);

}

sort(temp.begin(), temp.end());

ll res = 1e10;

for (ll z = i; z + k - 1 <= j; z++)

{

res = min(res, temp[z + k - 1] - temp[z]);

}

if (res <= mid)

{

return true;

}

}

return false;

}

int main()

{

fastio;

ll t = 1;

while (t--)

{

ll n, k;

cin >> n >> k;

vector<pair<ll, ll>> tiles(n);

for (auto &i : tiles)

cin >> i.first >> i.second;

ll lo = 1;

sort(tiles.begin(), tiles.end());

vector<ll> width(n), height(n);

for (ll i = 0; i < n; i++)

{

width[i] = tiles[i].first;

height[i] = tiles[i].second;

}

ll hi = max(\*max\_element(height.begin(), height.end()) - \*min\_element(height.begin(), height.end()), \*max\_element(width.begin(), width.end()) - \*min\_element(width.begin(), width.end()));

ll ans = hi;

while (lo <= hi)

{

ll mid = lo + (hi - lo) / 2;

bool res = checker(tiles, n, k, mid);

if (res)

{

ans = mid;

hi = mid - 1;

}

else

{

lo = mid + 1;

}

}

cout << ans << "\n";

}

}

Q3. There's a cost of removing stone, cost is different if it has one neighbour, two neighbour and it's zero if no neighbour.

Within a given sequence of stones, find minimum cost to remove all of them. Each stone had different costs for each neighbour condition

| #**include** <bits/stdc++.h> **using** **namespace** std; **int** **f**(**int** ind, **int** flag, vector<vector<**int**>> &dp, vector<**int**> &a, vector<**int**> &b){  **if**(ind == 1){  **if**(flag) **return** a[1];  **return** 0;  }  **if**(dp[ind][flag] !=-1) **return** dp[ind][flag];  **if**(flag){  **return** dp[ind][flag] = min(f(ind-1,0,dp,a,b)+b[ind],f(ind-1,1,dp,a,b)+a[ind]);  }**else**{  **return** dp[ind][flag] = min(f(ind-1,0,dp,a,b)+a[ind],f(ind-1,1,dp,a,b));  } }  **int** **main**() {  **int** n;  cin>>n;  vector<**int**>a(n+1),b(n+1);  **for**(**int** i=1;i<=n;i++){  cin >> a[i];  }  **for**(**int** i=1;i<=n;i++){  cin >> b[i];  }  vector<vector<**int**>> dp(n+1,vector<**int**>(2,-1));  cout << f(n,0,dp,a,b);   **return** 0; } |
| --- |

**Test - 5**

**Q1.**

You are given an array of strings. You can merge two strings, arr[i] and arr[j], only if,

i < j and the last letter of arr[i] == first letter of arr[j]

Eg, you have two strings -> “123” and “389” -> after merging it becomes “123389”

You can keep merging strings like this, but the ‘final’ string you form must be such that the first letter of the string should be the same as the last letter.

Eg, after merging several strings, the ‘final’ string becomes -> “123389…………1”

Print the max length of the ‘final’ string that can be formed in this way.

Test case - 1:  
Array of strings = [“14”, “123”, “323”, “321”, “421”, “535”]  
Possible ‘final’ strings -> “323”, “535”, “14421”, “123323321”, “123321”  
Ans = 9 (which is the length of “123323321”)

Test case – 2:  
Array of strings = [“14”, “15”, “89”, “22”]  
Possible ‘final’ strings -> “22”  
Ans = 2 (which is the length of “22”)

1 <= ai <=1e9. 1 <=N<= 1e5

*// Sarthak stalk krna buri baat hoti hai*

#include <iostream> *// for input and output streams*

#include <vector> *// for std::vector*

#include <algorithm> *// for algorithms like sort, max, min*

#include <string> *// for std::string*

#include <map> *// for std::map*

#include <unordered\_map> *// for std::unordered\_map*

#include <set> *// for std::set*

#include <unordered\_set> *// for std::unordered\_set*

#include <queue> *// for std::queue, std::priority\_queue*

#include <stack> *// for std::stack*

#include <deque> *// for std::deque*

#include <list> *// for std::list*

#include <iterator> *// for std::iterators*

#include <functional> *// for std::function, std::bind*

#include <numeric> *// for numeric algorithms like accumulate*

#include <cmath> *// for mathematical functions like sqrt, pow*

#include <limits> *// for limits of data types*

#include <iomanip> *// for std::setprecision, std::fixed*

#include <fstream> *// for file I/O*

#include <sstream> *// for string streams*

#include <bitset> *// for bitset operations*

#include <tuple> *// for std::tuple*

#include <utility> *// for std::pair, std::move, std::forward*

#include <ctime> *// for std::time*

#include <chrono> *// for time utilities*

#include <cassert> *// for assertions*

#include <random> *// for random number generation*

#include <memory> *// for smart pointers*

#include <limits.h>

using namespace std;

typedef long long ll;

typedef long double ld;

#define fastio \

ios\_base::sync\_with\_stdio(false); \

cin.tie(NULL); \

cout.tie(NULL)

#define max3(*a*, *b*, *c*) max(max(a, b), c)

#define max4(*a*, *b*, *c*, *d*) max(max(a, b), max(c, d))

#define fr(*i*, *n*) for (ll i = 0; i < n; i++)

ll gcd(ll *a*, ll *b*)

{

return *b* == 0 ? *a* : gcd(*b*, *a* % *b*);

}

int main()

{

fastio;

ll t = 1;

while (t--)

{

ll n;

cin >> n;

vector<string> v(n);

for (auto &i : v)

cin >> i;

vector<vector<ll>> mp(10, vector<ll>(10, 0));

for (ll i = n - 1; i >= 0; i--)

{

for (ll j = 0; j <= 9; j++)

{

if (mp[v[i].back() - '0'][j] != 0)

{

mp[v[i][0] - '0'][j] = max(mp[v[i][0] - '0'][j], (ll)(mp[v[i].back() - '0'][j] + v[i].length()));

}

}

mp[v[i][0] - '0'][v[i].back() - '0'] = max(mp[v[i][0] - '0'][v[i].back() - '0'], (ll)v[i].length());

}

ll res = 0;

for (ll i = 0; i <= 9; i++)

{

ll term = mp[i][i];

res = max(res, term);

}

cout << res << "\n";

}

}

**Q2**.

2 arrays given

a = [1, 2, 3, 4, 5]  
b = [6, 7, 8, 9, 10]

Value of an array is the total score of elements,   
and score of element is 1 if it is less than or equal to D and 2 if it is greater than D,

Find D such that A\_score - B\_Score is maximum

constraints -

0 <= D <= 1e9  
1 <= a[i] <= 1e8  
1 <= b[i] <= 1e8  
1 <= a.size(), b.size() <= 1e5

*// Sarthak stalk krna buri baat hoti hai*

#include <iostream> *// for input and output streams*

#include <vector> *// for std::vector*

#include <algorithm> *// for algorithms like sort, max, min*

#include <string> *// for std::string*

#include <map> *// for std::map*

#include <unordered\_map> *// for std::unordered\_map*

#include <set> *// for std::set*

#include <unordered\_set> *// for std::unordered\_set*

#include <queue> *// for std::queue, std::priority\_queue*

#include <stack> *// for std::stack*

#include <deque> *// for std::deque*

#include <list> *// for std::list*

#include <iterator> *// for std::iterators*

#include <functional> *// for std::function, std::bind*

#include <numeric> *// for numeric algorithms like accumulate*

#include <cmath> *// for mathematical functions like sqrt, pow*

#include <limits> *// for limits of data types*

#include <iomanip> *// for std::setprecision, std::fixed*

#include <fstream> *// for file I/O*

#include <sstream> *// for string streams*

#include <bitset> *// for bitset operations*

#include <tuple> *// for std::tuple*

#include <utility> *// for std::pair, std::move, std::forward*

#include <ctime> *// for std::time*

#include <chrono> *// for time utilities*

#include <cassert> *// for assertions*

#include <random> *// for random number generation*

#include <memory> *// for smart pointers*

#include <limits.h>

using namespace std;

typedef long long ll;

typedef long double ld;

#define fastio \

ios\_base::sync\_with\_stdio(false); \

cin.tie(NULL); \

cout.tie(NULL)

#define max3(*a*, *b*, *c*) max(max(a, b), c)

#define max4(*a*, *b*, *c*, *d*) max(max(a, b), max(c, d))

#define fr(*i*, *n*) for (ll i = 0; i < n; i++)

ll gcd(ll *a*, ll *b*)

{

return *b* == 0 ? *a* : gcd(*b*, *a* % *b*);

}

int main()

{

fastio;

ll t = 1;

while (t--)

{

ll n, m;

cin >> n >> m;

vector<ll> a(n), b(m);

set<ll> st;

for (auto &i : a)

{

cin >> i;

st.insert(i);

}

for (auto &i : b)

{

cin >> i;

st.insert(i);

}

sort(a.begin(), a.end());

sort(b.begin(), b.end());

ll D = -1;

ll scorediff = -1e9;

ll insert1 = min(a[0], b[0]) - 1;

ll insert2 = max(a[n - 1], b[m - 1]) + 1;

st.insert(insert1);

st.insert(insert2);

for (auto &i : st)

{

ll x = i;

ll lessina = upper\_bound(a.begin(), a.end(), x) - a.begin();

ll scorea = ((n - lessina) \* 2) + lessina;

ll lessinb = upper\_bound(b.begin(), b.end(), x) - b.begin();

ll scoreb = ((m - lessinb) \* 2) + lessinb;

ll term = scorea - scoreb;

if (term > scorediff)

{

scorediff = term;

D = x;

}

}

cout << D << "\n";

}

}

TEST-6

Q1.

There are N cars parked at random positions on x-y plane.

You'll be given M which means all the Locations of cars lies between the square (-M,M), (M,M), (M,-M) (-M,-M).

Your task is to move them at a given location (p,q) at the same time i.e. all of them should reach (p,q) at same time in minimum drives. And if not possible then return -1.

1st drive : you should take 1 move

2nd drive : you should take 2 moves

Tth drive : you should take t moves

You can go left, right, up or down in each move.

Also, you can revisit your previous location i.e if you are in say 5th drive, you can do 2 downs, 2 ups and 1 right or any possible combination.

Example 1:

N=2

P,q is (1,1)

Locations of cars are (2,3) and (-4,1)

Ans : 5 drives

(Total moves 1 + 2 + 3 + 4 + 5 = 15).

Example 2:

N=2

P,q = (0,0)

Locations : (0,1), (0,2)

Ans : -1

Constraints:

1 <= N <= 100

1 <= M <= 1e17

*// Sarthak stalk krna buri baat hoti hai*

#include <iostream> *// for input and output streams*

#include <vector> *// for std::vector*

#include <algorithm> *// for algorithms like sort, max, min*

#include <string> *// for std::string*

#include <map> *// for std::map*

#include <unordered\_map> *// for std::unordered\_map*

#include <set> *// for std::set*

#include <unordered\_set> *// for std::unordered\_set*

#include <queue> *// for std::queue, std::priority\_queue*

#include <stack> *// for std::stack*

#include <deque> *// for std::deque*

#include <list> *// for std::list*

#include <iterator> *// for std::iterators*

#include <functional> *// for std::function, std::bind*

#include <numeric> *// for numeric algorithms like accumulate*

#include <cmath> *// for mathematical functions like sqrt, pow*

#include <limits> *// for limits of data types*

#include <iomanip> *// for std::setprecision, std::fixed*

#include <fstream> *// for file I/O*

#include <sstream> *// for string streams*

#include <bitset> *// for bitset operations*

#include <tuple> *// for std::tuple*

#include <utility> *// for std::pair, std::move, std::forward*

#include <ctime> *// for std::time*

#include <chrono> *// for time utilities*

#include <cassert> *// for assertions*

#include <random> *// for random number generation*

#include <memory> *// for smart pointers*

#include <limits.h>

using namespace std;

typedef long long ll;

typedef long double ld;

#define fastio \

ios\_base::sync\_with\_stdio(false); \

cin.tie(NULL); \

cout.tie(NULL)

#define max3(*a*, *b*, *c*) max(max(a, b), c)

#define max4(*a*, *b*, *c*, *d*) max(max(a, b), max(c, d))

#define fr(*i*, *n*) for (ll i = 0; i < n; i++)

ll gcd(ll *a*, ll *b*)

{

return *b* == 0 ? *a* : gcd(*b*, *a* % *b*);

}

int main()

{

fastio;

ll t = 1;

while (t--)

{

ll n;

cin >> n;

vector<pair<ll, ll>> v(n);

for (auto &i : v)

cin >> i.first >> i.second;

pair<ll, ll> dest;

cin >> dest.first >> dest.second;

set<ll> parities;

for (ll i = 0; i < n; i++)

{

parities.insert((abs(dest.first - v[i].first) + abs(dest.second - v[i].second)) % 2);

}

if (parities.size() > 1)

{

cout << "-1\n";

}

else

{

ll parity = \*parities.begin();

ll maxterm = 0;

for (ll i = 0; i < n; i++)

{

ll term = (abs(dest.first - v[i].first) + abs(dest.second - v[i].second));

maxterm = max(maxterm, term);

}

ll lo = 1;

ll hi = (ll)(sqrtl(maxterm \* 2) + 1);

ll ans = hi;

while (lo <= hi)

{

ll mid = lo + (hi - lo) / 2;

ll sum = (mid \* (mid + 1)) / 2;

if (sum >= maxterm)

{

ans = mid;

hi = mid - 1;

}

else

{

lo = mid + 1;

}

}

ll anssum = (ans \* (ans + 1)) / 2;

if ((anssum % 2) == parity)

{

cout << ans << "\n";

}

else

{

ll sum1 = ((ans + 1) \* (ans + 2)) / 2;

ll sum2 = ((ans + 2) \* (ans + 3)) / 2;

if ((sum1 % 2) == parity)

{

cout << ans + 1 << "\n";

}

else

{

cout << ans + 2 << "\n";

}

}

}

}

}

**TEST-7**

**Q1.** Robot Sum

Given an array of garbage value at index i.

* A robot can be deployed at any index with cost m.
* A robot can clean garbage at index and can move only to i+1.
* At any point the cost is Sum of garbage which remains uncleaned.

Find the minimum cost of cleaning all garbage and any number of robots can be deployed at any index.

* Soln. -> 2D DP at any index you either deploy a new robot at that index or get it cleaned by a robot deployed at a previous index.

**Q2.** There is a company that issues gift certificates every day. The company will apply the following rules to the gift certificates

* A gift certificate is printed with its own serial number which is made up of only numbers. So, all gift certificates will have different serial numbers.
* The digit sum of a serial number is S. Gift certificates issued on the same day have the same S.

The maximum value of a serial number is A and its number of digits is N. (For example, if A = 34567, N is 5)  
  
Given A and S for gift certificates to be issued on a day, you are required to write a program that calculates the maximum issuable number of gift certificates

The calculated result may be too big. So, print the answer modulo 10^9 + 7.

You may increase the number from 1 to A by 1 and compare S with the sum of all digits. However, this method cannot be used because the given A can be too large. So you are required to write a more efficient program.

Examples:

* A = 101, S = 3, Answer: 4  
  Explanation: 21, 12, 3, and 30 can be made. As up to 101 can be made, 102 cannot be made
* A = 172, S = 3, Answer: 7  
  Explanation: 3, 12, 21, 30, 102, 111, and 120 can be made
* A = 50, S = 4, Answer 5  
  Explanation: 4, 40, 13, 31, and 22 can be made.
* A = 999 , S = 500 , Answer 0  
  Explanation: The maximum digit sum is 27. No gift certificate that meets the condition of S = 500 can be issued

Constraints: 1 <= A < 10^100, 1 <= S <= 1000

*// Sarthak stalk krna buri baat hoti hai*

#include <iostream> *// for input and output streams*

#include <vector> *// for std::vector*

#include <algorithm> *// for algorithms like sort, max, min*

#include <string> *// for std::string*

#include <map> *// for std::map*

#include <unordered\_map> *// for std::unordered\_map*

#include <set> *// for std::set*

#include <unordered\_set> *// for std::unordered\_set*

#include <queue> *// for std::queue, std::priority\_queue*

#include <stack> *// for std::stack*

#include <deque> *// for std::deque*

#include <list> *// for std::list*

#include <iterator> *// for std::iterators*

#include <functional> *// for std::function, std::bind*

#include <numeric> *// for numeric algorithms like accumulate*

#include <cmath> *// for mathematical functions like sqrt, pow*

#include <limits> *// for limits of data types*

#include <iomanip> *// for std::setprecision, std::fixed*

#include <fstream> *// for file I/O*

#include <sstream> *// for string streams*

#include <bitset> *// for bitset operations*

#include <tuple> *// for std::tuple*

#include <utility> *// for std::pair, std::move, std::forward*

#include <ctime> *// for std::time*

#include <chrono> *// for time utilities*

#include <cassert> *// for assertions*

#include <random> *// for random number generation*

#include <memory> *// for smart pointers*

#include <limits.h>

using namespace std;

typedef long long ll;

typedef long double ld;

#define fastio \

ios\_base::sync\_with\_stdio(false); \

cin.tie(NULL); \

cout.tie(NULL)

#define max3(*a*, *b*, *c*) max(max(a, b), c)

#define max4(*a*, *b*, *c*, *d*) max(max(a, b), max(c, d))

#define fr(*i*, *n*) for (ll i = 0; i < n; i++)

ll gcd(ll *a*, ll *b*)

{

return *b* == 0 ? *a* : gcd(*b*, *a* % *b*);

}

ll f(string &*s*, ll *idx*, ll *n*, ll *targetsod*, ll *sodtillnow*, ll *smaller*, vector<vector<vector<ll>>> &*dp*)

{

if (*idx* == *n*)

{

if (*sodtillnow* == *targetsod*)

{

return 1;

}

else

{

return 0;

}

}

else

{

if (*sodtillnow* > *targetsod*)

{

return 0;

}

if (*dp*[*idx*][*sodtillnow*][*smaller*] != -1)

return *dp*[*idx*][*sodtillnow*][*smaller*];

if (*smaller* == 1)

{

ll ans = 0;

for (ll i = 0; i <= 9; i++)

{

ans += f(*s*, *idx* + 1, *n*, *targetsod*, (*sodtillnow* + i), *smaller*, *dp*);

}

return *dp*[*idx*][*sodtillnow*][*smaller*] = ans;

}

else

{

ll ans = 0;

for (ll i = 0; i < (*s*[*idx*] - '0'); i++)

{

ans += f(*s*, *idx* + 1, *n*, *targetsod*, (*sodtillnow* + i), 1, *dp*);

}

ans += f(*s*, *idx* + 1, *n*, *targetsod*, (*sodtillnow* + (*s*[*idx*] - '0')), 0, *dp*);

return *dp*[*idx*][*sodtillnow*][*smaller*] = ans;

}

}

}

int main()

{

fastio;

ll t = 1;

while (t--)

{

ll n, sod;

cin >> n >> sod;

string s;

cin >> s;

vector<vector<vector<ll>>> dp(n + 1, vector<vector<ll>>(sod + 1, vector<ll>(2, -1)));

cout << f(s, 0, n, sod, 0, 0, dp) << "\n";

}

}