**Test - 1**

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 follow

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 of 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.

**Test - 2**

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 headquarters.

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

**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/))

/\*

class Solution {

public:

int findMaxLength(vector<int>& v) {

int n=v.size();

int sum=0;

map<int,int>m;

m[0]=0;

int ans=0;

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

if(v[i]==0) sum--;

else sum++;

if(sum==0) ans=i+1;

else if(m[sum]!=0) ans=max(ans,i+1-m[sum]);

if(m[sum]==0) m[sum]=i+1;

}

return ans;

}

};

\*/

**Test - 4**

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. The 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 good truck carries)) like

To move one block cost is 1 on an empty truck

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

A 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

Q2. Given N tiles of given width an koid 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.

Q3. There's a cost of removing stone, the cost is different if it has one neighbour, two neighbours 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

**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”)

/\*

#include <bits/stdc++.h>

using namespace std;

using ll = long long;

ll dp[100001][11][11];

ll f(ll index, char start, char end, vector<string> &v)

{

if (index == v.size())

{

if (start == end)

{

return 0;

}

else

{

return -1e15;

}

}

if (dp[index][(start == '@' ? 10 : start - '0')][(end == '@' ? 10 : end - '0')] != -1)

{

return dp[index][(start == '@' ? 10 : start - '0')][(end == '@' ? 10 : end - '0')];

}

ll ans = -1e15;

ans = max(ans, f(index + 1, start, end, v));

if (end == v[index][0] || end == '@')

{

ans = max(ans, (ll)v[index].size() + f(index + 1, (start == '@' ? v[index][0] : start), v[index][v[index].size() - 1], v));

}

return dp[index][(start == '@' ? 10 : start - '0')][(end == '@' ? 10 : end - '0')] = ans;

}

void solve()

{

ll n;

cin >> n;

vector<string> v(n);

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

{

cin >> v[i];

}

ll ans = 0;

memset(dp, -1, sizeof(dp));

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

{

ans = max(ans, f(i, '@', '@', v));

}

cout << (ans <= 0 ? 0 : ans) << "\n";

}

**Iterative:**

#include <bits/stdc++.h>

using namespace std;

int main() {

int n;

cin >> n;

vector<string> vtr(n);

for(int i = 0; i < n; i++) cin >> vtr[i];

vector<vector<int>> prev(10, vector<int>(10, 0)), cur(prev);

prev[vtr[0][0] - '0'][vtr[0].back() - '0'] = vtr[0].length();

for(int i = 1; i < (int)vtr.size(); i++) {

for(int start = 0; start <= 9; start++)

for(int end = 0; end <= 9; end++)

cur[start][end] = prev[start][end];

int curStart = vtr[i][0] - '0';

int curEnd = vtr[i].back() - '0';

int curLen = vtr[i].length(in);

for(int start = 0; start <= 9; start++) {

if(prev[start][curStart] == 0) continue;

cur[start][curEnd] = max(cur[start][curEnd], prev[start][curStart] + curLen);

}

cur[curStart][curEnd] = max(cur[curStart][curEnd], curLen);

prev = cur;

}

int maxi = 0;

for(int endPoint = 0; endPoint <= 9; endPoint++) maxi = max(maxi, cur[endPoint][endPoint]);

cout << maxi << endl;

}

\*/

**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

**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 mov⁸es

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.

Constraints:

1 <= N <= 100

1 <= M <= 1e7

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

**TEST-7**

**Q1.** Robot Sum

Given an array of garbage values 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.

* 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 on 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