L11 : Dynamic Programming 1 Practice Questions

1-Tut : AlphaCode-Question

Send Feedback

Alice and Bob need to send secret messages to each other and are discussing ways to encode their messages:

Alice: “Let’s just use a very simple code: We’ll assign ‘A’ the code word 1, ‘B’ will be 2, and so on down to ‘Z’ being assigned 26.”
Bob: “That’s a stupid code, Alice. Suppose I send you the word ‘BEAN’ encoded as 25114. You could decode that in many different ways!”
Alice: “Sure you could, but what words would you get? Other than ‘BEAN’, you’d get ‘BEAAD’, ‘YAAD’, ‘YAN’, ‘YKD’ and ‘BEKD’. I think you would be able to figure out the correct decoding. And why would you send me the word ‘BEAN’ anyway?”
Bob: “OK, maybe that’s a bad example, but I bet you that if you got a string of length 5000 there would be tons of different decodings and with that many you would find at least two different ones that would make sense.”
Alice: “How many different decodings?”
Bob: “Jillions!”
For some reason, Alice is still unconvinced by Bob’s argument, so she requires a program that will determine how many decodings there can be for a given string using her code.

Input Format:

First line will contain T (number of test case).
Each input is consists of a single line containing the message string

Output Format:

For each test case print the answer % mod (10^9 +7)

Constraints:

1 <= T <= 100
1 <= |S| <= 10^5
sum of length of all string doesn't exceed 5*10^6

Sample Input 1:

```
3
47974
6349988978
1001
```

Sample Output 1:

```
1
1

0
```

```cpp
#include <bits/stdc++.h>
using namespace std;
#define mod 1000000007

int alpha_code(int * input, int size) {
    vector<int> output(size + 1,0);
    output[0] = 1;
    output[1] = 1;

    for (int i = 2; i <= size; i++) {
        if (input[i-1] != 0)
        {
            output[i] = output[i - 1] % mod;
        }

        if (input[i-2] *10 + input[i - 1] <= 26)
        {
            if (input[i-2] != 0)
            {
                output[i] = (output[i - 2] + output[i])%mod;
            }
        }
    }
    int ans = output[size];
    return ans;
}

int main(){

    // write your code here
    int T; cin >> T;
    while(T--){
        string input; cin >> input;
        if(input[0] == '0') return 0;
        int len = input.length();
        int arr[len];
        for(int i=0; i<len; i++)
            arr[i] = input[i] - 48;
        cout << alpha_code(arr,len) << endl;
    }
    return 0;
}
```

2-Tut : **Largest Bitonic Subsequence**

Send Feedback

You are given an array of positive integers as input. Write a code to return the length of the largest such subsequence in which the values are arranged first in strictly ascending order and then in strictly descending order.

Such a subsequence is known as bitonic subsequence. A purely increasing or purely decreasing subsequence will also be considered as a bitonic sequence with the other part empty.

Note that the elements in bitonic subsequence need not be consecutive in the given array but the order should remain same.

Input Format:

First line will contain T (number of test case), each test is consists of two lines.
Line 1 : A positive Integer N, i.e., the size of array
Line 2 : N space-separated integers as elements of the array

Output Format:

Length of Largest Bitonic subsequence for each test case in a newline.

Input Constraints:

1 <= T <= 10
1<= N <= 5000

Sample Input 1:

```
1
6
15 20 20 6 4 2
```

Sample Output 1:

```
5
```

Sample Output 1 Explanation:

Here, longest Bitonic subsequence is {15, 20, 6, 4, 2} which has length = 5.

Sample Input 2:

```
1
2
1 5
```

Sample Output 2:

```
2
```

Sample Input 3:

```
1
2
5 1
```

Sample Output 3:

```
2
```

```
#include<iostream>
using namespace std;
#include<algorithm>

int* LISfront(int* arr,int n){
    int* output = new int[n];
    output[0] = 1;
    for(int i=1;i<n;i++){
        output[i] = 1;
        for(int j=i-1;j>=0;j--){
            if(arr[j]>=arr[i]){
```

```
            continue;
        }
        int currAns = output[j] + 1;
        if(currAns > output[i]){
            output[i] = currAns;
        }

    }
  }
  return output;
}

int* LISback(int* arr,int n){
  int* output = new int[n];
  output[n-1] = 1;
  for(int i=n-2;i>=0;i--){
    output[i] = 1;
    for(int j=i+1;j<n;j++){
      if(arr[j] >= arr[i]){
        continue;
      }
      int currAns = output[j]+1;
      if(currAns > output[i]){
        output[i] = currAns;
      }
    }

  }
  return output;
}


int longestBitonicSubarray(int *input, int n) {

  int* LISbegin = LISfront(input,n);
  int* LISend = LISback(input,n);
  int maximum = 1;
  for(int i=0;i<n;i++){
    maximum = max(maximum,LISbegin[i]+LISend[i]-1);
  }
  return maximum;
}

int main()
{
  int T; cin >>T;
  while(T--)
  {
    int n, input[100000];
```

```
        cin>>n;
        for(int i=0; i<n; i++)
        {
           cin>>input[i];
        }
      cout << longestBitonicSubarray(input, n) << endl;
    }
  return 0;
}
```

3-Tut : StairCase Problem

Send Feedback

A child is running up a staircase with n steps and can hop either 1 step, 2 steps or 3 steps at a time. Implement a method to count how many possible ways the child can run up to the stairs. You need to return all possible number of ways.

Time complexity of your code should be O(n).

Since the answer can be pretty large print the answer % mod(10^9 +7)

Input Format:

First line will contain T (number of test case).
Each test case is consists of a single line containing an integer N.

Output Format:

For each test case print the answer in new line

Constraints :

1 <= T < = 10
1 <= N <= 10^5
Sample input 1

2
2
3
Sample output 1

2
4
Explanation of sample input 1:

Test case 1:
To reach at top of 2nd stair, we have only two options i.e (2), (1,1)

Test case 2:
To reach at top of 3rd stair, we have four options i.e (1,1,1), (1,2) ,(2,1), (3)
Sample input 2:

2
5

10
Sample output 2:

13

274

```
#include<bits/stdc++.h>
using namespace std;
#include <iostream>
long long mod =1e9+7;

long long int staircase(int n){
    if(n<=1)
        return 1;

    long long a = 1, b = 1, c = 2; for (int i = 0; i <= n - 3; ++i) { long long d = (a + b + c) % mod; a = b; b = c; c = d; } return c; }


int main()
{
    int T;
    cin >> T;
    while(T--)
    {
        int N; cin >> N;
        cout << staircase(N) << endl;

    }
    return 0;
}
/*aur mod me 1e9+7 liya


yeh liya karo jab aisa hu kare
*/
```

4-Tut : Coin Change Problem

Send Feedback

You are given an infinite supply of coins of each of denominations D = {D0, D1, D2, D3, ... Dn-1}. You need to figure out the total number of ways W, in which you can make a change for Value V using coins of denominations D.

Note : Return 0, if change isn't possible.

W can be pretty large so output the answer % mod(10^9 + 7)

Input Format

First line will contain T (number of test case), each test case is consists of 3 three lines.
Line 1 : Integer n i.e. total number of denominations
Line 2 : N integers i.e. n denomination values
Line 3 : Value V

Output Format

For each test case print the number of ways (W) % mod(10^9 +7) in new line.
Constraints :

1 <= T <= 10
1 <= N <= 10
1 <= V <= 5000

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
long long mod =1e9+7;
using namespace std;
int getCount(int* arr,int n,int value,vector<vector<int>>&temp){
    if(value == 0){
        return 1;
    }
    if(n == 0){
        return 0;
    }
    if(value < 0){
        return 0;
    }
    if(temp[n][value] > -1){
        return temp[n][value];
    }
    int first = getCount(arr,n,value-arr[n-1],temp) % mod;
    int second = getCount(arr,n-1,value,temp) % mod;
    return temp[n][value] = (first+second) % mod;
}

int countWaysToMakeChange(int denominations[], int n, int value){

    vector<vector<int>>temp(n+1,vector<int>(value+1,-1));
            return getCount(denominations,n,value,temp);

}


signed main(){

    // write your code here
    int T; cin >> T;
```

```
    while(T--){
            int numDenominations;
            cin >> numDenominations;
            int* denominations = new int[numDenominations];
        for(int i = 0; i < numDenominations; i++)
        {
          cin >> denominations[i];
        }
        int value;
        cin >> value;
        cout << countWaysToMakeChange(denominations, numDenominations, value) << endl;
    }
    return 0;
}
```

5-Tut : Magic Grid Problem

Send Feedback

You are given a magrid S (a magic grid) having R rows and C columns. Each cell in this magrid has either a Hungarian horntail dragon that our intrepid hero has to defeat, or a flask of magic potion that his teacher Snape has left for him. A dragon at a cell (i,j) takes away |S[i][j]| strength points from him, and a potion at a cell (i,j) increases Harry's strength by S[i][j]. If his strength drops to 0 or less at any point during his journey, Harry dies, and no magical stone can revive him.

Harry starts from the top-left corner cell (1,1) and the Sorcerer's Stone is in the bottom-right corner cell (R,C). From a cell (i,j), Harry can only move either one cell down or right i.e., to cell (i+1,j) or cell (i,j+1) and he can not move outside the magrid. Harry has used magic before starting his journey to determine which cell contains what, but lacks the basic simple mathematical skill to determine what minimum strength he needs to start with to collect the Sorcerer's Stone. Please help him once again.

Input Format :

The first line contains the number of test cases T. T cases follow. Each test case consists of R C in the first line followed by the description of the grid in R lines, each containing C integers. Rows are numbered 1 to R from top to bottom and columns are numbered 1 to C from left to right. Cells with S[i][j] < 0 contain dragons, others contain magic potions.

Output Format :

Output T lines, one for each case containing the minimum strength Harry should start with from the cell (1,1) to have a positive strength through out his journey to the cell (R,C).

Constraints:

$1 \leq T \leq 5$

$2 \leq R, C \leq 500$

$-10^3 \leq S[i][j] \leq 10^3$

S[1][1] = S[R][C] = 0

Sample Input

```
3
2 3
0 1 -3
1 -2 0
2 2
0 1
2 0
3 4
0 -2 -3 1
-1 4 0 -2
1 -2 -3 0
```

Sample Output

```
2
1

2
```

```
#include<bits/stdc++.h>
using namespace std;

int getStrength(int** arr,int r,int c){
    int** temp = new int*[r];
    for(int i=0;i<r;i++){
        temp[i] = new int[c]();
    }
    temp[r-1][c-2] = 1;
    temp[r-2][c-1] = 1;
    temp[r-1][c-1] = 1;

    //setting last row
    for(int i=c-3;i>=0;i--){
        if(arr[r-1][i+1] < 0){
            temp[r-1][i] = temp[r-1][i+1] - arr[r-1][i+1];
        }else{
            temp[r-1][i] = max(1,temp[r-1][i+1] - arr[r-1][i+1]);
        }
    }

    for(int i=r-3;i>=0;i--){
        if(arr[i+1][c-1] < 0){
            temp[i][c-1] = temp[i+1][c-1] - arr[i+1][c-1];
        }else{
            temp[i][c-1] = max(1,temp[i+1][c-1] - arr[i+1][c-1]);
```

```
        }
    }

    for(int i=r-2;i>=0;i--){
        for(int j=c-2;j>=0;j--){
            int right,down;
            if(arr[i][j+1] < 0){
                right = temp[i][j+1] - arr[i][j+1];
            }else{
                right = max(1,temp[i][j+1] - arr[i][j+1]);
            }

            if(arr[i+1][j] < 0){
                down = temp[i+1][j] - arr[i+1][j];
            }else{
                down = max(1,temp[i+1][j]  - arr[i+1][j]);
            }

            temp[i][j] = min(right,down);
        }
    }

    int ans = temp[0][0];
    return ans;

}


int main(){

    // write your code here
    int T; cin >> T;
    while(T--){
        int R,C; cin >> R >> C;
        int **arr = new int *[R];
        for(int i = 0; i < R; i++){
            arr[i] = new int[C];
        }
        for(int i=0;i<R;i++){
            for(int j=0;j<C;j++){
                cin >> arr[i][j];
            }
        }
        int ans = getStrength(arr,R,C);
        cout << ans << endl;
    }
    return 0;
}
```

Live Question 1 : https://www.hackerrank.com/challenges/construct-the-array/problem

Live Question 2 : https://www.hackerrank.com/challenges/sam-and-substrings/problem

6-Ass : Loot Houses

Send Feedback

A thief wants to loot houses. He knows the amount of money in each house. He cannot loot two consecutive houses. Find the maximum amount of money he can loot.

Input Format :

The first line of input contains a single integer N denoting the total number of houses.
The second line of input contains N single space-separated integers, denoting the amount of money in every i-th house.

Output Format :

The only line of output will print the maximum amount of loot that is possible.

Input Constraints

0 <= N <= 10^5
0 <= A[i] <= 10^4
Where N is the total number of houses.
A[i] represents the money present in the i-th house.
Time limit: 1sec

Sample Input 1:

6
5 5 10 100 10 5

Sample Output 1 :

110

Sample Input 2:

4
10 2 3 11

Sample Output 2 :

21

Explanation to Sample Input 2:

Since the thief cant loot two consecutive houses, the ways in which he may loot are:

1. [10, 3]: a total loot of 13
2. [10, 11]: a total loot of 21
3. [2, 11]: a total loot of 13
4. [10]: a total loot of 10
5. [2]: a total loot of 2
6. [3]: a total loot of 3
7. [11]: a total loot of 11

We can't neglect the option to loot just either of the houses if it yields the maximum loot.

From all the possible seven ways, the second option yields the maximum loot amount and hence the answer.

```
#include<bits/stdc++.h>
using namespace std;
int loottotal(int *arr,int n, int si, int ei){
    // Recursive without DP correct solution but giving TLE
   if(n == 0){
      return 0;
   }

    if(si == ei){
      return arr[si];
    }
    if(si == ei-1){
      return max(arr[si], arr[ei]);
    }

    int loot1 = arr[si] + loottotal(arr,n,si+2,ei);
    int loot2 = loottotal(arr,n,si+1,ei);

    return max(loot1,loot2);

}
int main(){

    // write your code here
    int N; cin >> N;
    int *arr = new int[N];
    for(int i = 0 ; i < N;i++){
      cin >> arr[i];
    }

    cout << loottotal(arr,N,0,N-1) << endl;

    return 0;
}
```

```
int loottotal(int *arr,int n, int si, int ei){
    // iterative dp
    if(n == 0){
      return 0;
    }
    if(n == 1){
      return arr[0];
    }

    int *dp = new int[n];
```

```
    dp[0] = arr[0];
    dp[1] = max(arr[0],arr[1]);

    for(int i = 2; i < n; i++){

        int loot1 = arr[i] + dp[i-2];
        int loot2 = dp[i-1];

        dp[i] = max(loot1,loot2);

    }
    return dp[n-1];
}

```

Solution :

1. We can do it without declaring the dp array (space complexity reduced by dp array size) bu using 3 variables

```
int loottotal(int *arr,int n, int si, int ei){
    // iterative
    if(n == 0){
        return 0;
    }
    if(n == 1){
        return arr[0];
    }

    if(n == 2){
        return max(arr[0],arr[1]);
    }

    int curr;
    int prev1 = max(arr[0],arr[1]);
    int prev2 = arr[0];

    for(int i = 2; i < n; i++){
            curr = max(prev1,arr[i] + prev2);
            prev2 = prev1;
        prev1 = curr;

    }
    return curr;
}

```

7-Ass : Boredom

Send Feedback

Gary is bored and wants to play an interesting but tough game . So he figured out a new board game called "destroy the neighbours" . In this game there are N integers on a board. In one move, he can pick any integer x from the board and then all the integers with value x+1 or x-1 gets destroyed .This move will give him x points.

He plays the game until the board becomes empty . But as he want show this game to his friend Steven, he wants to learn techniques to maximise the points to show off . Can you help Gary in finding out the maximum points he receive grab from the game ?

Input Format :

First line will contain T (number of test case), each test case is consists of two line.
Line 1: Integer N
Line 2: A list of N integers

Output Format :

For each test case print maximum points, Gary can receive from the Game setup in a newline.

Constraints :

1 <= T <= 50
1 <= N <= 10^5
1 <= A[i] <= 1000

Sample Input :

1
2
1 2

Sample Output :

2

Explanation:

Gary can receive a maximum of 2 points, by picking the integer 2.

```
#include<bits/stdc++.h>
using namespace std;
int destroyneighbour(int *arr, int N){

    unordered_map<int,int> m;

    for(int i=0;i<N;i++){
        m[arr[i]] += 1;
    }
    int* output = new int[1005]();
    output[1] = 1*m[1];

    for(int i=2;i<=1000;i++){
        output[i] = max(output[i-1],((i)*(m[i]))+output[i-2]);
    }
    int ans = output[1000];
```

```
    return ans;
}

int main(){

    // write your code here
    int T; cin >> T;
    while(T--){

        int N; cin >> N;
        int *arr = new int[N];
        for(int i = 0; i < N;i++){
            cin>>arr[i];
        }

        cout << destroyneighbour(arr,N) <<endl;
    }
    return 0;
}
```

Solution :

```
int destroyneighbour(int *arr, int n){

                        int freq[1001];
                        int dp[1001];
                        for (int i = 0; i <= 1000; i++)
                                {
            freq[i] = 0;
            dp[i] = 0;
            }
        for (int i = 0; i < n; i++)
         {
            freq[arr[i]]++;
         }
            dp[1] = freq[1];
        for (int i = 2; i <= 1000; i++)
        {
            dp[i] = max(dp[i - 1], dp[i - 2] + i * freq[i]);

        }
    return dp[1000];
}


```

8-Ass : Minimum Number of Chocolates

Send Feedback

Noor is a teacher. She wants to give some chocolates to the students in her class. All the students sit in a line and each of them has a score according to performance. Noor wants to give at least 1 chocolate to each student. She distributes chocolates to them such that If two students sit next to each other then the one with the higher score must get more chocolates. Noor wants to save money, so she wants to minimise the total number of chocolates.

Note that when two students have equal score they are allowed to have different number of chocolates.

Input Format:

First line will contain T(number of test case), each test case consists of two lines.
First Line: Integer N, the number of students in Noor's class.
Second Line: Each of the student's score separated by spaces.

Output Format:

Output the minimum number of chocolates Noor must give for each test case in a newline.

Input Constraints

1 <= T <= 50
1 <= N <= 50000
1 <= score <= 10^9

Sample Input:

1
4
1 4 4 6

sample Output:

6

Explanation:

The number of chocolates distributed could be:
1 2 1 2

Sample Input:

1
3
8 7 5

sample Output:

6

```
#include<bits/stdc++.h>
using namespace std;

int getMin(int *arr, int n){

    int* output = new int[1000005]();
    output[0] = 1;
    int count = 1;
    //left to right assigning with min possible value
        for(int i=1;i<n;i++){
```

```
        if(arr[i] > arr[i-1]){
            output[i] = ++count;
        }else{
            output[i] = 1;
            count = 1;
        }
    }
    //right to left re-traversing to make sure two person adjacent get correct chocolates (ex : 8 7 5) (1,1,1) -> (3,2,1)
    for(int i=n-1;i>=0;i--){
        if(arr[i-1] > arr[i] && output[i-1] <= output[i]){
            output[i-1] = output[i]+1;
        }
    }
    int sum = 0;
    for(int i=0;i<=n;i++){
        sum += output[i];
    }
   return sum;
}

int main(){

    // write your code here
    int T; cin >> T;
    while(T--){
        int n;
        cin >> n;
        int *arr = new int[n];
        for(int i = 0; i < n; i++){
            cin >> arr[i];
        }
       cout << getMin(arr, n) << endl;
    }

    return 0;
}
```

Solution :

```
int getMin(int *arr, int n){

            int dp[n];
                        dp[0] = 1;
                        int i = 0;
          int sum = 0;
        for (i = 1; i < n; i++)
          {
            if (arr[i] > arr[i - 1])
                        {
```

```
            dp[i] = dp[i - 1] + 1;
        }
        else
        dp[i] = 1;
    }
  for (i = n - 2; i >= 0; i--)
    {
        if (arr[i] > arr[i + 1] && dp[i] <= dp[i + 1])
        {
        dp[i] = dp[i + 1] + 1;
        }
    }
        for (i = 0; i < n; i++)
      sum += dp[i];
  return sum;
}
```

9-Ass : Minimum Count

Send Feedback

Given an integer N, find and return the count of minimum numbers, sum of whose squares is equal to N. That is, if N is 4, then we can represent it as : {1^2 + 1^2 + 1^2 + 1^2} and {2^2}. Output will be 1, as 1 is the minimum count of numbers required.

Note : x^y represents x raise to the power y.

Input Format :

First line will contain T(number of test case), each test case consists of a single line containing an integer N.

Output Format :

For each test case print the required minimum count in a newline.

Constraints :

1 <= T <= 1000

1 <= N <= 1000

Sample Input 1 :

1

12

Sample Output 1 :

3

Sample Output 1 Explanation :

12 can be represented as :

1^1 + 1^1 + 1^1 + 1^1 + 1^1 + 1^1 + 1^1 + 1^1 + 1^1 + 1^1 + 1^1 + 1^1

1^1 + 1^1 + 1^1 + 1^1 + 1^1 + 1^1 + 1^1 + 1^1 + 2^2

1^1 + 1^1 + 1^1 + 1^1 + 2^2 + 2^2

2^2 + 2^2 + 2^2

As we can see, the output should be 3.

Sample Input 2 :

```
1
9
```

Sample Output 2 :

```
1
```

```
#include<bits/stdc++.h>
using namespace std;
int minCount(int n)
{

        int dp[n+1];

        dp[0] = 0;
        dp[1] = 1;
  for (int i = 2; i <= n; ++i)
  {
        dp[i] = INT_MAX;
        for (int j = 1; i-(j*j) >= 0; j++)
        {
                dp[i] = min(dp[i],dp[i-(j*j)]);
        }
        dp[i] +=1;
  }

  return dp[n];

}
int main(){

  // write your code here
  int T;cin>>T;
  while(T--){
     int n; cin>>n;
     cout << minCount(n)<<endl;
  }
  return 0;
}
```

Solution : Same

10-Ass : Hasan and Trip

Send Feedback

Hasan has finally finished his final exams and he decided to go in a trip among cities in Syria.

There are N cities in Syria and they are numbered from 1 to N, each city has coordinates on plane, i-th city is in (Xi, Yi).

Hasan is in first city and he wants to visit some cities by his car in the trip but the final destination should be N-th city and the sequence of cities he will visit should be increasing in index (i.e. if he is in city i he can move to city j if and only if i < j).

Visiting i-th city will increase Hasan's happiness by Fi units (including first and last cities), also Hasan doesn't like traveling too much, so his happiness will decrease by total distance traveled by him.

Help Hasan by choosing a sequence of cities to visit which maximizes his happiness.

Input format:

First line will contain T(number of test case).
First line of each test case will contain an integer N
Next N lines of that test case will contain three space-separated integers Xi, Yi, Fi (coordinates and happiness)
Output format:

For each test Output one number rounded to 6 digits after floating point, the maximum possible happiness in newline, Hasan can get.
Note: If answer is 2 print 2.000000
Constraints:

1 <= T <= 50
1 <= N <= 500
0 <= Xi, Yi, Fi <= 100,000
Sample Input

```
1
3
0 0 1
3 1 1
6 0 9
```

Sample Output

```
4.675445
```

```
1. #include<bits/stdc++.h>
2. using namespace std;
3. double inf = 1e15;
4.
5. double disc(pair<double, double> p1, pair<double, double> p2)
6. {
```

```
    double a, b;
    a = (p2.first - p1.first) * (p2.first - p1.first);
    b = (p2.second - p1.second) * (p2.second - p1.second);
    return sqrt(a+b);
}
void solve(pair<double, double> arr[], double happy[], int n)
{
    double dp[n];
    dp[0] = happy[0];
    for(int i=1;i<n;i++)
    {
        dp[i] = -inf;
        for(int j=0;j<i;j++)
        {
            double x = disc(arr[i], arr[j]);
            dp[i] = max(dp[i], dp[j]-x);
        }
        dp[i] += happy[i];
    }
    cout<<fixed;
    cout<<setprecision(6)<<dp[n-1]<<endl;
    return;
}
int main()
{
    int t;
    cin>>t;
    while(t--)
    {
        int n;
        cin>>n;
        pair<double, double> arr[n];
        double a,b;
        double happy[n];
        for(int i=0;i<n;i++)
        {
            cin>>a>>b;
            arr[i] = make_pair(a,b);
            cin>>happy[i];
        }
        solve(arr,happy,n);
    }
}
```

Solution : Same

11-Ass : Vanya and GCD

Send Feedback

Vanya has been studying all day long about sequences and other Complex Mathematical Terms. She thinks she has now become really good at it. So, her friend Vasya decides to test her knowledge and keeps the following challenge it front of her:

Vanya has been given an integer array A of size N. Now, she needs to find the number of increasing sub-sequences of this array with length ≥1 and GCD=1. A sub-sequence of an array is obtained by deleting some (or none) elements and maintaining the relative order of the rest of the elements. As the answer may be large, print it Modulo 109+7

She finds this task really easy, and thinks that you can do it too. Can you?

Input Format:

First line will contain T(number of test case), each test consists of two line.
The first line contains a single integer N denoting size of array A.
The next line contains N space separated integers denoting the elements of array A

Output Format:

Print the required answer Modulo 10^9+7 for each test case in new line

Constraints:

1 <= T <= 50
1 <= N <= 200
1 <= A[i] <= 100

Sample Input

```
1
3
1 2 3
```

Sample Output

```
5
```

```
#include<bits/stdc++.h>
using namespace std;
int mod = 1e9 + 7;
int solve(int arr[],int n)
{
    int maxVal = INT_MIN;
    int dp[n+1][105];

    for(int i=0;i<n;i++)
    {
        for(int j=0;j<=100;j++)
        {
```

```
            dp[i][j] = 0;
        }
    }

    for(int i=0;i<n;i++)
    {
        maxVal = max(arr[i], maxVal);
        dp[i][arr[i]] = 1;
    }

    for(int i=1;i<n;i++)
    {
        for(int j=0;j<i;j++)
        {
            if(arr[i] > arr[j])
            {
                for(int k=1;k<=maxVal;k++)
                {
                    int x = __gcd(k,arr[i]);
                    dp[i][x] += dp[j][k];
                    if(dp[i][x] >= mod)
                        dp[i][x] -= mod;
                }
            }
        }
    }

    int ans = 0;
    for(int i=0;i<n;i++)
    {
        ans += dp[i][1];
        if(ans >= mod)
            ans -= mod;
    }
    return ans;
}

int main(){

    // write your code here
    int t;
    cin>>t;
    while(t--)
    {
        int n;
        cin>>n;
        int arr[n];
        for(int i=0;i<n;i++)
        {
```

```
            cin>>arr[i];
        }
        cout<<solve(arr,n)<<endl;
    }
    return 0;
}
```

Solution : Same

12-Ass : Roy and Coin Boxes

Send Feedback

Roy has N coin boxes numbered from 1 to N.

Every day he selects two indices [L,R] and adds 1 coin to each coin box starting from L to R (both inclusive).

He does this for M number of days.

After M days, Roy has a query: How many coin boxes have at least X coins.

He has Q such queries.

Input Format:

First line will contain T (number of test case), format of each test case follows
First line contains two space separated integers N and M (N - number of coin boxes, M - number of days).
Each of the next M lines consists of two space separated integers L and R. Followed by integer Q - number of queries.
Each of next Q lines contain a single integer X.

Output Format:

For each query of each test case output the result in a new line.

Constraints:

1 <= T <= 10
1 ≤ N ≤ 10000
1 ≤ M ≤ min(10000, N)
1 ≤ L ≤ R ≤ N
1 ≤ Q ≤ 10000
1 ≤ X ≤ N
Sample Input

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

2
Sample Output

6
0
0

4

```
#include<bits/stdc++.h>
using namespace std;
int main(){

    // write your code here
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    int t;
    cin>>t;
    while(t--)
    {
        int n,m,q,l,r;
        cin>>n>>m;
        int start[n+1], end[n+1], coins[n+1], ans[n+1];
        for(int i=0;i<=n;i++)
        {
            start[i] = 0;
            end[i] = 0;
            coins[i] = 0;
            ans[i] = 0;
        }
        for(int i=0;i<m;i++)
        {
            cin>>l>>r;
            start[l]++; //noting the times the starting box was kth box
                                    end[r]++; //noting the times the ending box was kth box
        }
        int temp = 0;
        for(int i=1;i<=n;i++)
        {
            temp += start[i];
            coins[i] = temp;
            temp -= end[i];
        }

        //now calculate how many boxes have exact i coins
        for(int i=1;i<=n;i++)
        {
            ans[coins[i]]++;
        }
```

```
41.         /*now calculate how many boxes have atleast i coins which is same as
42.          number of boxes having atleast i+1 coins + number of boxes having exact i coins
43.          Note: box with atleast max coins is same as box with exact k coins
44.          In other words we can take max as n and start iterating from n-1*/
45.         for(int i=n-1;i>0;i--)
46.         {
47.             ans[i] = ans[i]+ans[i+1];
48.         }
49.         cin>>q;
50.         while(q--)
51.         {
52.             cin>>temp;
53.             cout<<ans[temp]<<endl;
54.         }
55.     }
56.     return 0;
57. }
```

13-Ass : Alyona and Spreadsheet

Send Feedback

During the lesson small girl Alyona works with one famous spreadsheet computer program and learns how to edit tables.

Now she has a table filled with integers. The table consists of n rows and m columns. By ai, j we will denote the integer located at the i-th row and the j-th column. We say that the table is sorted in non-decreasing order in the column j if ai, j ≤ ai + 1, j for all i from 1 to n - 1.

Teacher gave Alyona k tasks. For each of the tasks two integers l and r are given and Alyona has to answer the following question: if one keeps the rows from l to r inclusive and deletes all others, will the table be sorted in non-decreasing order in at least one column? Formally, does there exist such j that ai, j ≤ ai + 1, j for all i from l to r - 1 inclusive.

Alyona is too small to deal with this task and asks you to help!

Input Format:

First line of input will contain T(number of test case), each test case is described as.
The first line of the each test case contains two positive integers n and m the number of rows and the number of columns in the table respectively.
Each of the following n lines contains m integers. The j-th integers in the i of these lines stands for ai, j.

The next line of the input contains an integer k, the number of task that teacher gave to Alyona.

The i-th of the next k lines contains two integers li and ri

Output Format:

For each test case, print "Yes" to the i-th line of the output if the table consisting of rows from li to ri inclusive is sorted in non-decreasing order in at least one column. Otherwise, print "No".

Constriants:

1 <= T <= 10

1 <= N, M <= 20000
1 <= N*M <= 20000
1 <= arr[i][j] <= 10^9
1 <= K <= 10000
1 <= l <= r <= N

Sample Input :

1
3 11
5 1 3 4 5 1 5 5 3 3 2
5 8 2 10 1 9 8 4 4 3 4
15 6 9 2 7 1 3 13 7 7 5
1
1 3

Sample Output :

Yes

```
#include<bits/stdc++.h>
using namespace std;
vector<int> solve(vector<vector<int>> &mat, int n, int m)
{
    vector<int> dp(n, 0);
    int index[n][m];
    for(int i=0;i<m;i++)
    {
        index[0][i] = 0;
        for(int j=1;j<n;j++)
        {
            if(mat[j][i] >= mat[j-1][i])
                index[j][i] = index[j-1][i];
            else
                index[j][i] = j;
        }
    }
    for(int i=0;i<n;i++)
    {
        dp[i] = i;
        for(int j=0;j<m;j++)
        {
            dp[i] = min(dp[i], index[i][j]);
        }
    }
    return dp;
}

int main(){

    // write your code here
    int t;
```

```
    cin>>t;
    while(t--)
    {
        int n,m;
        cin>>n>>m;
        vector<vector<int>> mat(n,vector<int>(m,0));
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<m;j++)
            {
                cin>>mat[i][j];
            }
        }
            vector<int> dp = solve(mat, n, m);
            int q, l, r;
            cin>>q;
            while(q--)
            {
                cin>>l>>r;
                l--;
                r--;
                if(dp[r] <= l)
                    cout<<"Yes\n";
                else
                    cout<<"No\n";
            }
    }


    return 0;
}
```

Solution :

14-Ass : Angry Children

Send Feedback

Bill Gates is on one of his philanthropic journeys to a village in Utopia. He has N packets of candies and would like to distribute one packet to each of the K children in the village (each packet may contain different number of candies). To avoid a fight between the children, he would like to pick K out of N packets such that the unfairness is minimized.

Suppose the K packets have (x1, x2, x3,....xk) candies in them, where xi denotes the number of candies in the ith packet, then we define unfairness as

```
unfairness=0;
for(i=0;i<n;i++)
   for(j=i;j<n;j++)
```

unfairness+=abs(xi-xj)
abs(x) denotes absolute value of x.

Input Format:

First line will contain T(number of test cases), and each test case consists of two lines.
The first line contains two space-separated integers N and K.
The second line will contain N space-separated integers, where Ith integer denotes the candy in the Ith packet.

Output Format:

For each test case print a single integer which will be minimum unfairness in newline.
Constraints

1 <= T <= 10
2<=N<=10^5
2<=K<=N
0<= number of candies in each packet <=10^6
Sample Input

1
7 3
10 100 300 200 1000 20 30
Sample Output

40
Explanation

Bill Gates will choose packets having 10, 20 and 30 candies.So unfairness will be |10-20| + |20-30| + |10-30| = 40. We can verify that it will be minimum in this way.

```
#include<bits/stdc++.h>
using namespace std;

long long int solve(int arr[], int n, int k)
{
    sort(arr,arr+n);
    vector<long long> sum(n+1,0);
    long long curr=0,ans;
    sum[0] = 0;
    for(int i=0;i<n;i++)
    {
        sum[i+1] = sum[i] + arr[i];
    }
    for(int i=0;i<k;i++)
    {
        curr += (1ll*i*arr[i]-sum[i]);
```

```
    }
    ans = curr;
    for(int i=k;i<n;i++)
    {
        curr = curr-2ll*(sum[i]-sum[i-k+1]) + 1ll*(k-1)*(arr[i]+arr[i-k]);
        ans = min(ans,curr);
    }
    return ans;
}


int main(){

    // write your code here
     int t;
    cin>>t;
    while(t--)
    {
        int n,k;
        cin>>n>>k;
        int arr[n];
        for(int i=0;i<n;i++)
        {
            cin>>arr[i];
        }
        cout<<solve(arr,n,k)<<endl;
    }
    return 0;
}
```