REFERRAL TEMPLETE

While asking for a referral, you can simply follow the below template which will contain crisp information and can do the job.  
  
"Hi Koulick,  
I am Aditya Roy, pursuing [B.Tech](http://b.tech/) CSE from ABC college. I have strong problem solving skills and have solved 200 problems at (codeforces/codechef/leetcode) and rated Expert.  
Also I did my internship at X company and have experience in these tech stacks.  
It would be great if you can refer me for the SDE Intern/FTE position at Uber(job\_id) [Give job\_id link].  
  
Thank you! "  
  
Now, just attach your resume along with this message and you are done, thats it.

**Q.1 Dice Combinations**

Your task is to count the number of ways to construct sum n by throwing a dice one or more times. Each throw produces an outcome between 1 and 6.  
For example, if n=3, there are 4 ways:

* 1+1+1
* 1+2
* 2+1
* 3

**Input**  
The only input line has an integer n.  
**Output**  
Print the number of ways modulo 109+7.  
**Constraints** : 1≤n≤106

* int solve(int n,vector<int>& dp){
* if (n==0) return 1;
* if (dp[n]!=-1) return dp[n];
* int ans=0;
* for(int i=1;i<=min((int)(6),n);i++){
* ans = (ans+solve(n-i,dp))%mod;
* }
* return dp[n] = ans;
* }
* int32\_t main()
* {
* FIO;
* int n; cin>>n;
* vector<int> dp(n+1,-1);
* cout<<solve(n,dp)<<endl;
* }

# Q. 2 Minimizing Coins https://cses.fi/problemset/task/1634

Consider a money system consisting of n coins. Each coin has a positive integer value. Your task is to produce a sum of money x using the available coins in such a way that the number of coins is minimal.  
For example, if the coins are {1,5,7} and the desired sum is 11, an optimal solution is 5+5+1which requires 3 coins.  
  
**Input**  
  
The first input line has two integers nn and xx: the number of coins and the desired sum of money.  
  
The second line has nn distinct integers c1,c2,…,cnc1,c2,…,cn: the value of each coin.  
  
**Output**  
  
Print one integer: the minimum number of coins. If it is not possible to produce the desired sum, print −1.  
  
**Constraints**

* 1≤n≤1001≤n≤100
* 1≤x≤1061≤x≤106
* 1≤ci≤1061≤ci≤106

**Example**  
  
Input:  
3 11  
1 5 7  
  
Output:  
3

// https://codeforces.com/blog/entry/70018?locale=ru

int32\_t main()

{

    FIO;

      int n,target; cin>>n>>target;

      vector<int> a(n);

      forn(i,n) cin>>a[i];

      vector<int> dp(target+1,1e9);

      dp[0] = 0; // no. of coins to produce sum 0 is 0;

      for (int i = 1; i <= target; i++) {

        for (int j = 0; j < n; j++) {

          if (i-a[j] >= 0) {

            dp[i] = min(dp[i], dp[i-a[j]]+1);

          }

        }

      }

      cout << (dp[target] == 1e9 ? -1 : dp[target]) << endl;

}

# Q. 3. Coin Combinations I https://cses.fi/problemset/task/1635/

Consider a money system consisting of n coins. Each coin has a positive integer value. Your task is to calculate the number of distinct ways you can produce a money sum x using the available coins.  
  
For example, if the coins are {2,3,5} and the desired sum is 9, there are 8 ways:

* 2+2+52+2+5
* 2+5+22+5+2
* 5+2+25+2+2
* 3+3+33+3+3
* 2+2+2+32+2+2+3
* 2+2+3+22+2+3+2
* 2+3+2+22+3+2+2
* 3+2+2+23+2+2+2

**Input**  
  
The first input line has two integers nn and xx: the number of coins and the desired sum of money.  
  
The second line has nn distinct integers c1,c2,…,cnc1,c2,…,cn: the value of each coin.  
  
**Output**  
  
Print one integer: the number of ways modulo 109+7109+7.  
  
**Constraints**

* 1≤n≤1001≤n≤100
* 1≤x≤1061≤x≤106
* 1≤ci≤1061≤ci≤106

**Example**  
  
Input:  
3 9  
2 3 5  
  
Output:  
8

Solution:

This problem has a very similar implementation to the previous problem.

dp[x] = number of ways to make value x.

We initialize dp[0] = 1, saying the empty set is the only way to make 0.

Like in "Minimizing Coins", we loop over the possibilities for last coin added. There are dp[x-v] ways to make x, when adding a coin with value v last. This is since we can choose any combination for the first coins to sum to x-v, but need to choose v as the last coin. Summing over all the possibilities for v gives dp[x]

    int n,s; cin>>n>>s;

      vector<int> a(n);

      forn(i,n) cin>>a[i];

      int t[s+1];

      memset(t,0,sizeof(t)); //  IMP

      t[0]=1;

      for(int i=1;i<=s;i++){

          for(int j=0;j<n;j++){

              if (a[j]<=i){

                  t[i] =(t[i]+t[i-a[j]])%mod;

              }

          }

      }

      cout<<t[s]<<endl;

# Q 4. Coin Combinations II DOUBT https://cses.fi/problemset/task/1636

Consider a money system consisting of n coins. Each coin has a positive integer value. Your task is to calculate the number of distinct *ordered* ways you can produce a money sum xx using the available coins.  
  
For example, if the coins are {2,3,5}{2,3,5} and the desired sum is 99, there are 33 ways:

* 2+2+52+2+5
* 3+3+33+3+3
* 2+2+2+32+2+2+3

**Input**  
  
The first input line has two integers nn and xx: the number of coins and the desired sum of money.  
  
The second line has nn distinct integers c1,c2,…,cnc1,c2,…,cn: the value of each coin.  
  
**Output**  
  
Print one integer: the number of ways modulo 109+7109+7.  
  
**Constraints**

* 1≤n≤1001≤n≤100
* 1≤x≤1061≤x≤106
* 1≤ci≤1061≤ci≤106

**Example**  
  
Input:  
3 9  
2 3 5  
  
Output:  
3

#include <bits/stdc++.h>

using namespace std;

int main() {

int mod = 1e9+7;

int n, target;

cin >> n >> target;

vector<int> x(n);

for (int&v : x) cin >> v;

vector<vector<int>> dp(n+1,vector<int>(target+1,0));

dp[0][0] = 1;

for (int i = 1; i <= n; i++) {

for (int j = 0; j <= target; j++) {

dp[i][j] = dp[i-1][j];

int left = j-x[i-1];

if (left >= 0) {

(dp[i][j] += dp[i][left]) %= mod;

}

}

}

cout << dp[n][target] << endl;

}

# Q 5. Removing Digits

# https://cses.fi/problemset/task/1637/

You are given an integer n. On each step, you may subtract one of the digits from the number.  
  
How many steps are required to make the number equal to 0?  
  
**Input**  
  
The only input line has an integer nn.  
  
**Output**  
  
Print one integer: the minimum number of steps.  
  
**Constraints**

* 1≤n≤1061≤n≤106

**Example**  
  
Input:  
27  
  
Output:  
5  
  
Explanation: An optimal solution is 27→20→18→10→9→027→20→18→10→9→0.

int solve(int n,vector<int>& dp){

    if(n<=0) return 0;

    else if(dp[n]!=-1) return dp[n];

    vector<int> digits;

    int temp=n;

    while(temp){

        digits.push\_back(temp%10);

        temp/=10;

    }

    int ans=INT\_MAX;

    for(int i=0;i<digits.size();i++){

        if (digits[i]==0) continue;

        ans = min(ans,1+solve(n-digits[i],dp));

    }

    return dp[n]=ans;

}

// https://codeforces.com/blog/entry/70018?locale=ru

int32\_t main()

{

    FIO;

      int n;cin>>n;

      vector<int> dp(n+1,-1);

      cout<<solve(n,dp)<<endl;

}

TABULAR SOL.

    FIO;

      int n;

      cin >> n;

      vector<int> dp(n+1,1e9); // Initialising with INT\_MAX if (digit==0) then it will stuck in infinite loop

      dp[0] = 0;

      for (int i = 0; i <= n; i++) {

        for (char c : to\_string(i)) {

          dp[i] = min(dp[i], dp[i-(c-'0')]+1);

        }

      }

      cout << dp[n] << endl;

# Q 6. Array Description https://cses.fi/problemset/task/1746/

You know that an array has n integers between 1 and m, and the absolute difference between two adjacent values is at most 1.  
  
Given a description of the array where some values may be unknown, your task is to count the number of arrays that match the description.  
  
**Input**  
  
The first input line has two integers nn and mm: the array size and the upper bound for each value.  
The next line has nn integers x1,x2,…,xnx1,x2,…,xn: the contents of the array. Value 00 denotes an unknown value.  
**Output**  
Print one integer: the number of arrays modulo 109+7109+7.  
  
**Constraints**

1≤n≤1051≤n≤105

1≤m≤1001≤m≤100

0≤xi≤m0≤xi≤m

**Example**  
Input:  
3 5  
2 0 2  
Output:  
3  
  
Explanation: The arrays [2,1,2][2,1,2], [2,2,2][2,2,2] and [2,3,2][2,3,2] match the description.

Solution:       int n,m; cin>>n>>m;

      vector<int> a(n);

      forn(i,n) cin>>a[i];

      int t[n+1][m+2];

      memset(t,0,sizeof(t));

      for(int i=1;i<=n;i++){

          for(int x=1;x<=m;x++){

              if (i==1){

                  if(a[i-1]==0 || a[i-1]==x) t[i][x]=1;

                  else t[i][x]==0;

              }

              else{

                     if (a[i-1]==0 || a[i-1]==x){

                          t[i][x] = ((t[i-1][x-1] + t[i-1][x])%mod + t[i-1][x+1])%mod;

                      }

                    else{

                          t[i][x]=0;

                      }

              }

          }

      }

      int ans=0;

      for(int i=1;i<=m;i++){

          ans =(ans + t[n][i])%mod;

      }

      cout<<ans<<endl;

No. of arrays ending at pos I with x

Dp[i,x] = dp[i-1, x-1] + dp[i-1, x] + dp[i-1,x+1]

2 cases: if (a[i]==0 or a[i]==x) // in both these cases ith element ends with x

Dp[i,x] = dp[i-1, x-1] + dp[i-1, x] + dp[i-1,x+1]

Else

Dp[I,x] = 0; // cannot end with x

i==1 case handled separately.

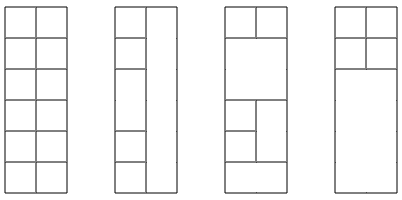
M+2 cols size is to handle x=m case

[**https://www.youtube.com/watch?v=d1H5JylYG4I&list=PLb3g\_Z8nEv1h1w6MI8vNMuL\_wrI0FtqE7&index=10**](https://www.youtube.com/watch?v=d1H5JylYG4I&list=PLb3g_Z8nEv1h1w6MI8vNMuL_wrI0FtqE7&index=10)

# Q7. Counting Towers DOUBT

**https://cses.fi/problemset/task/2413**

Your task is to build a tower whose width is 22 and height is nn. You have an unlimited supply of blocks whose width and height are integers.  
  
For example, here are some possible solutions for n=6n=6:



Given nn, how many different towers can you build? Mirrored and rotated towers are counted separately if they look different.  
  
**Input**  
The first input line contains an integer tt: the number of tests.  
After this, there are tt lines, and each line contains an integer nn: the height of the tower.  
  
**Output**  
For each test, print the number of towers modulo 109+7109+7.  
  
**Constraints**

1≤t≤1001≤t≤100

1≤n≤1061≤n≤106

**Example**  
  
Input:  
3  
2  
6  
1337  
Output:  
8  
2864  
640403945

 w(t){

        int n; cin>>n;

        int t[n+2][2];

        t[n+1][1]=t[n+1][0] = 1; // base case

        for(int i=n;i>=2;i--){

            int op1 = (t[i+1][1] + t[i+1][0])%mod;

            int op2 = t[i+1][0];

            int op3 = (2\*t[i+1][0])%mod;

            int op4 = t[i+1][1];

            t[i][0] = ((op1+op2)%mod + op3)%mod; // i-1 unlinked

            t[i][1] = (op1 + op4)%mod;            // i-1 linked

        }

        int ans= (t[2][0] + t[2][1])%mod;

        cout<< ans<<endl;

    }

## **Q8. Edit Distance (1639)**

## When we calculate dp[i][k], there are four possibilities to consider for the rightmost operation. We check all of them and take the cheapest one.

1. We deleted character a[i-1]. This took one operation, and we still need to change a[:i-1] to b[:k]. So this costs 1 + dp[i-1][k] operations.

2. We added character b[k-1] to the end of a[:i]. This took one operation, and we still need to change a[:i] to b[:k-1]. So this costs 1 + dp[i][k-1] operations.

3. We replaced a[i-1] with b[k-1]. This took one operation, and we still need to change a[:i-1] to b[:k-1]. So this costs 1 + dp[i-1][k-1] operations.

4. a[i-1] was already equal to b[k-1], so we just need to change a[:i-1] to b[:k-1]. That takes dp[i-1][k-1] operations. This possibility can be viewed as a replace operation where we don't actually need to replace a[i-1].

**The complexity is O(|a|⋅|b|)O(|a|⋅|b|).**

string s1,s2;

        cin>>s1>>s2;

        int t[s1.length()+1][s2.length()+1];

        for(int i=1;i<=s1.length();i++) t[i][0] = i;

        for(int i=1;i<=s2.length();i++) t[0][i] = i;

        t[0][0]=0;

        for(int i=1;i<=s1.length();i++){

            for(int j=1;j<=s2.length();j++){

                if(s1[i-1] != s2[j-1]){

                    t[i][j] = 1+min({t[i-1][j],t[i-1][j-1], t[i][j-1]});

                }

                else{

                    t[i][j] = t[i-1][j-1];

                }

            }

        }

        cout<<t[s1.length()][s2.length()]<<endl;

# Q9. Rectangle Cutting

**https://cses.fi/problemset/task/1744/**

Given an a×b rectangle, your task is to cut it into squares. On each move you can select a rectangle and cut it into two rectangles in such a way that all side lengths remain integers. What is the minimum possible number of moves?  
  
**Input**  
The only input line has two integers aa and bb.  
  
**Output**  
Print one integer: the minimum number of moves.  
  
**Constraints**

1≤a,b≤5001≤a,b≤500

**Example**  
Input:  
3 5  
Output:  
3

       int a,b; cin>>a>>b;

       int t[a+1][b+1];

       memset(t,0,sizeof(t));

       for(int i=1;i<=a;i++){

           for(int j=1;j<=b;j++){

               if(i==j){ t[i][j]=0; continue; }

               int op1=INT\_MAX, op2=INT\_MAX;

               for(int k=1;k<=i-1;k++){

                   op1 = min(op1, 1+t[k][j]+t[i-k][j]);

               }

               for(int k=1;k<=j-1;k++){

                   op2 = min(op2, 1+t[i][k]+t[i][j-k]);

               }

               t[i][j] = min(op1, op2);

           }

       }

       cout<<t[a][b]<<endl;

# Q 10. Removal Game

[**https://cses.fi/problemset/task/1097/**](https://cses.fi/problemset/task/1097/)

There is a list of nn numbers and two players who move alternately. On each move, a player removes either the first or last number from the list, and their score increases by that number. Both players try to maximize their scores.  
  
What is the maximum possible score for the first player when both players play optimally?  
  
**Input**  
  
The first input line contains an integer nn: the size of the list.  
  
The next line has nn integers x1,x2,…,xnx1,x2,…,xn: the contents of the list.  
  
**Output**  
Print the maximum possible score for the first player.  
  
**Constraints**

* 1≤n≤50001≤n≤5000
* −109≤xi≤109−109≤xi≤109

**Example**  
Input:

4  
4 5 1 3

Output:  
8

The trick here is to see that since the sum of the two players' scores is the sum of the input list, player 1 tries to maximize score1−score2 while player 2 tries to minimize it.

dp[l][r] = differencescore1−score2 considering the game played only on interval [l, r].

If the interval contains only one element (l = r), then the first player must take that element. So dp[i][i] = x[i].

Otherwise, player 1 can choose to take the first element or the last element. If he takes the first element, he gets x[l] points, and we are left with the interval [l+1,r], but with player 2 starting. score1−score2 on interval [l+1,r] is just dp[l+1][r] if player 1 starts. Since player 2 starts, it is -dp[l+1][r]. Thus, the difference of scores will be x[l]-dp[l+1][r] if player 1 chooses the first element. Similarly, it will be x[r]-dp[l][r-1] if he chooses the last element. He always chooses the maximum of those, so dp[l][r] = max(x[l]-dp[l+1][r], x[r]-dp[l][r-1]).

In this problem dp[l][r] depends on dp[l+1][r], and therefore we need to compute larger l before smaller l. We do it by looping through l from high to low. r still needs to go from low to high, since we depend only on smaller r (dp[l][r] depends on dp[l][r-1]). Note that in all the other problems in this editorial, dp only depends on smaller indices (like dp[x] depending on dp[x-v], or dp[i][x] depending on dp[i-1][x]), which means looping through indices in increasing order is correct.

We can reconstruct the score of player 1 as the mean of, the sum of all input values, and score1−score2score1−score2.

The complexity is O(n2)

    int n; cin>>n;

    int a[n];

    forn(i,n) cin>>a[i];

    int t[n][n];

    for(int i=n-1;i>=0;i--){

        for(int j=i;j<n;j++){

            if (i==j){

                t[i][j]=a[i];

            }

            else{

                t[i][j] = max(a[i]-t[i+1][j], a[j]-t[i][j-1]);

            }

        }

    }

    int diff = t[0][n-1];

    int s=0;

    forn(i,n) s+=a[i];

    cout<<(s+diff)/2<<endl;

# Q.11 Two Sets II

**https://cses.fi/problemset/task/1093/**

Your task is to count the number of ways numbers ,2,…,n can be divided into two sets of equal sum.  
  
For example, if n=7n=7, there are four solutions:

* {1,3,4,6}{1,3,4,6} and {2,5,7}{2,5,7}
* {1,2,5,6}{1,2,5,6} and {3,4,7}{3,4,7}
* {1,2,4,7}{1,2,4,7} and {3,5,6}{3,5,6}
* {1,6,7}{1,6,7} and {2,3,4,5}{2,3,4,5}

**Input**  
  
The only input line contains an integer nn.  
  
**Output**  
  
Print the answer modulo 109+7109+7.  
  
**Constraints**

* 1≤n≤5001≤n≤500

**Example**  
  
Input:  
7  
  
Output:  
4

    int n; cin>>n;

    int sum=(n\*(n+1))/2;

    if (sum%2) {cout<<"0"<<endl; return 0; }

    sum/=2;

    // count no.of ways to make sum/2;

    int t[n+1][sum+1];

    for(int i=0;i<=n;i++) t[i][0]=1;

    for(int j=1;j<=sum;j++) t[0][j]=0;

    for(int i=1;i<=n;i++){

        for(int j=1;j<=sum;j++){

            if(i <= j){

                t[i][j] = (t[i-1][j] + t[i-1][j-i])%mod;

            }

            else{

                t[i][j] = t[i-1][j];

            }

        }

    }

    // diving by 2 after performing mod operation, giving wa. Other way is multiply it with mod inverse of 2 wrt 1e9+7(got its value using mod inverse calcultor)

    cout<<(t[n][sum]\*500000004)%mod<<endl;

# Q12. Projects

# https://cses.fi/problemset/task/1140

There are nn projects you can attend. For each project, you know its starting and ending days and the amount of money you would get as reward. You can only attend one project during a day.  
  
What is the maximum amount of money you can earn?  
  
**Input**  
  
The first input line contains an integer nn: the number of projects.  
  
After this, there are nn lines. Each such line has three integers aiai, bibi, and pipi: the starting day, the ending day, and the reward.  
  
**Output**  
  
Print one integer: the maximum amount of money you can earn.  
  
**Constraints**

* 1≤n≤2⋅1051≤n≤2⋅105
* 1≤ai≤bi≤1091≤ai≤bi≤109
* 1≤pi≤1091≤pi≤109

**Example**  
  
Input:  
4  
2 4 4  
3 6 6  
6 8 2  
5 7 3  
  
Output:  
7

On day i, maybe we just did nothing, so we earn what we earned on day i-1, i.e dp[i-1]. Otherwise, we just finished some project. We earned some money doing the project, and use dp[start of project] to know how much money we could have earned before starting the project.

struct cell{

    int s,e,r;

    cell(int end, int start, int reward){

        e = end; s = start; r = reward;

    }

};

bool compare(cell& a, cell& b){

    if (a.e==b.e) return a.s < b.s;

    return a.e < b.e;

}

int getLargestj(vector<cell>& a, int lo,int hi, int s){

    int ans=-1;

    while(lo<=hi){

        int mid=lo+(hi-lo)/2;

        if (a[mid].e >= s){

            hi = mid-1;

        }

        else{

            ans=mid;

            lo = mid+1;

        }

    }

    return ans;

}

// https://codeforces.com/blog/entry/70018?locale=ru

int32\_t main()

{

    FIO;

    int n; cin>>n;

    vector<cell> a;

    forn(i,n){

        int x,y,z; cin>>x>>y>>z;

        cell t(y,x,z);

        a.push\_back(t); // endtime, starttime, reward

    }

    // sort according to end point

     sort(a.begin(),a.end(),compare);

//   forn(i,a.size()) cout<<a[i].s<<" "<<a[i].e<<" "<<a[i].r<<endl;

    int t[n+1];

    t[0]=0;

    t[1]=a[0].r;

    for(int i=2;i<=n;i++){

        int j = getLargestj(a,0,i-2,a[i-1].s);

        if (j!=-1)

            t[i] = max(t[i-1], a[i-1].r+t[j+1]);

        else

          t[i] = max(t[i-1], a[i-1].r);

    }

    // forn(i,n+1) cout<<t[i]<<" ";

    cout<<t[n]<<endl;

}