TECHNICAL SKILLS- TASK 2

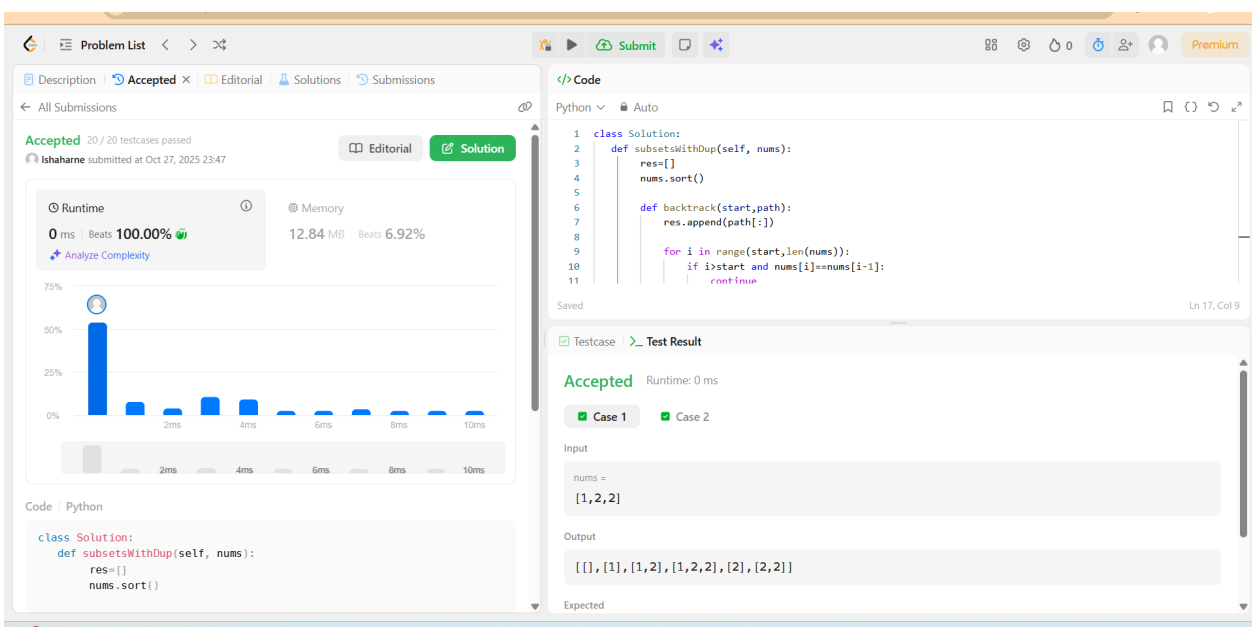
Name: Neer Awasthi

Class: A4 B2

Roll No:20

Leetcode Questions

1



2.

Problem List | Submit | Premium

Description | Editorial | Solutions | Submissions

131. Palindrome Partitioning

Medium | Topics | Companies

Given a string `s`, partition `s` such that every substring of the partition is a **palindrome**. Return *all possible palindrome partitioning of* `s`.

Example 1:

```
Input: s = "aab"
Output: [["a","a","b"],["aa","b"]]
```

Example 2:

```
Input: s = "a"
Output: [["a"]]
```

Constraints:

- `1 <= s.length <= 16`
- `s` contains only lowercase English letters.

Seen this question in a real interview before? 1/5

14K | 228 | 90 Online

Code

Python | Auto

```python
class Solution(object):
    def partition(self, s):
        res = []

        def isPalindrome(sub):
            return sub == sub[::-1]

        def backtrack(start, path):
            if start == len(s):
                res.append(path[:])
                return
```

Saved | Ln 17, Col 8

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

s =
"aab"

Output

[["a","a","b"],["aa","b"]]

Expected

3.

Problem List | Submit | Premium

Description | Editorial | Solutions | Submissions | Run Ctrl

38. Count and Say

Medium | Topics | Companies | Hint

The **count-and-say** sequence is a sequence of digit strings defined by the recursive formula:

- `countAndSay(1) = "1"`
- `countAndSay(n)` is the run-length encoding of `countAndSay(n - 1)`.

Run-length encoding (RLE) is a string compression method that works by replacing consecutive identical characters (repeated 2 or more times) with the concatenation of the character and the number marking the count of the characters (length of the run). For example, to compress the string `"3322251"` we replace `"33"` with `"23"`, replace `"222"` with `"32"`, replace `"5"` with `"15"` and replace `"1"` with `"11"`. Thus the compressed string becomes `"23321511"`.

Given a positive integer `n`, return *the* n^{th} *element of the* ***count-and-say*** *sequence.*

Example 1:

```
Input: n = 4

Output: "1211"

Explanation:

countAndSay(1) = "1"
countAndSay(2) = RLE of "1" = "11"
```

5K | 345 | 54 Online

Code

Python | Auto

```python
class Solution:
    def countAndSay(self, n):
        if n == 1:
            return "1"

        res = "1"
        for _ in range(n - 1):
            temp = ""
            count = 1
            for i in range(1, len(res)):
                if res[i] == res[i - 1]:
```

Saved | Ln 19, Col 19

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

n =
1

Output

"1"

Expected

4.

Problem List | Submit | Premium

Description | Editorial | Solutions | Submissions

47. Permutations II

Medium | Topics | Companies

Given a collection of numbers, `nums`, that might contain duplicates, return *all possible unique permutations* ***in any order***.

Example 1:

```
Input: nums = [1,1,2]
Output:
[[1,1,2],
 [1,2,1],
 [2,1,1]]
```

Example 2:

```
Input: nums = [1,2,3]
Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

Constraints:

- `1 <= nums.length <= 8`
- `-10 <= nums[i] <= 10`

9K | 67 | 67 Online

Code

Python | Auto

```python
class Solution(object):
    def permuteUnique(self, nums):
        nums.sort()
        result = []
        picked = [0 for _ in nums]

        def backtrack(current):
            if len(current) == len(nums):
                result.append(current[:])
                return

            for i in range(len(nums)):
                if picked[i] == 1:
                    continue
                if i > 0 and nums[i] == nums[i - 1] and picked[i - 1] == 0:
                    continue

                picked[i] = 1
```

Saved | Ln 26, Col 1

Testcase | Test Result

Accepted Runtime: 4 ms

Case 1 | Case 2

Input

nums =
[1,1,2]

5.

5.

Description | Editorial | Solutions | Submissions

22. Generate Parentheses

Medium | Topics | Companies

Given `n` pairs of parentheses, write a function to *generate all combinations of well-formed parentheses.*

Example 1:

```
Input: n = 3
Output: ["((()))","(()())","(())()","()(())","()()()"]
```

Example 2:

```
Input: n = 1
Output: ["()"]
```

Constraints:

- `1 <= n <= 8`

Seen this question in a real interview before? 1/5

Yes | No

Code

Python | Auto

```python
class Solution(object):
    def generateParenthesis(self, n):
        res = []

        def backtrack(openN, closeN, path):

            if openN == closeN == n:
                res.append(path)
                return


            if openN < n:
                backtrack(openN + 1, closeN, path + "(")


            if closeN < openN:
                backtrack(openN, closeN + 1, path + ")")
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

n =

Hackerearth Questions:

1.

A binary palindrome

13666 50% 30 ★★★★★ 46 votes Basic Programming, Recursion, Recursion and Backtracking, C++ Share

Details | Submissions | Discussion | Similar Problems | Editorial

Problem

You are given a number N. In one operation, you can either increase the value of N by *1* or decrease the value of N by *1*.

Determine the minimum number of operations required (possibly zero) to convert number N to a number P such that binary representation of P is a palindrome.

Note: A binary representation is said to be a palindrome if it reads the same from left-right and right-left.

Input format

- The first line contains an integer T denoting the number of test cases.
- For each test case, the first line contains an integer N.

Output format

For each test case in a new line, print the minimum number of operations required.

Constraints

$1 \le T \le 10^5$
$0 \le N \le 2 \times 10^9$

Sample Input	Sample Output
2 5 9	1 0

Time Limit: 1
Memory Limit: 256
Source Limit:

Explanation

For first test case

- If we decrease the value of *N* by *1*. Then *N = 5*, whose binary representation is *101* which is a palindrome.
- Hence, minimum *1* operation is required.

For second test case

- *N = 9*, has binary representation *1001* which is a palindrome.
- Hence, *0* operations is required.

Enter your code or Upload your code as file. Save Python 3.8 (python 3.8.10)

```python
def is_binary_palindrome(x):
    b = bin(x)[2:]
    return b == b[::-1]

t = int(input())

for _ in range(t):
    n = int(input())

    steps = 0
    while True:

        if n - steps >= 0 and is_binary_palindrome(n - steps):
            print(steps)
            break
        if is_binary_palindrome(n + steps):
            print(steps)
            break
        steps += 1
```

12:6 vscode

Test against custom input ▾ Compile & Test code Submit code

Log ID: 284152297 / Oct 28, 2025 11:00 PM IST (Asia/Kolkata)

Problem

This will the last contest organised by the **Cypher's senior Team**. So they want to make a new team which is capable of framing good questions. Now they know from experience that good team can only be created if all the team members are friends with each other and all of them are from the same section.
That is why they picked Top **N** students from the best section of LPU and interviewed each one of them and asked them about the persons they were friends with.
Now akgarhwal compiled the answer of all the candidate's and made a list such that each item in this contains two students **A** and **B** who are not friends with each other. The list will contain **K** such pairs.
Now your task is to create a team which have the maximum number of members such that all the team members are friends. Since there can be multiple answers for the same problem you have to print the name which comes first Lexicographically.

INPUT FORMAT:

- The first line of input contains two integer N and K, denoting the number of students and number of lines in the list, respectively.
- Each of next N lines contains a string describing the name of ith student of the section.
- Next, K Lines will contain two string A and B,the students who are not friends with each other.

OUTPUT FORMAT:

- The first line will contain an integer denoting the Numbers of students in the desired team.
- Print the name of the selected member of the team in lexicographical order separated by white space.

Constraint:

1<= **N** <= 16

1<=|**name**|<=50

0<= **K** <= *(N*(N+1))/2*

Code:

```
#include <stdio.h>

#include <string.h>


#define MAXN 16


int main() {

  int N, K;    scanf("%d

%d", &N, &K);


   char names[MAXN][51];

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

scanf("%s", names[i]);
```

```
    int notFriend[MAXN][MAXN] = {0};

    for (int i = 0; i < K; i++) {         char
A[51], B[51];         scanf("%s %s", A,
B);         int idxA = -1, idxB = -1;
for (int j = 0; j < N; j++) {             if
(strcmp(names[j], A) == 0) idxA = j;
if (strcmp(names[j], B) == 0) idxB = j;
        }
        if (idxA != -1 && idxB != -1) {
notFriend[idxA][idxB] = 1;
notFriend[idxB][idxA] = 1;
        }
    }

    int maxSize = 0;
    char bestTeam[MAXN][51];
int totalSubsets = 1 << N;

    for (int mask = 1; mask < totalSubsets; mask++) {
        int valid = 1;
int count = 0;
       char tempTeam[MAXN][51];
       int idxList[MAXN];
```

```
        for (int i = 0; i < N; i++) {            if
(mask & (1 << i)) {
strcpy(tempTeam[count], names[i]);
                idxList[count++] = i;
            }
        }

        for (int i = 0; i < count && valid; i++) {
for (int j = i + 1; j < count && valid; j++) {
if (notFriend[idxList[i]][idxList[j]]) valid = 0;
            }
        }

        if (valid) {             for (int i = 0; i < count - 1;
i++) {                 for (int j = i + 1; j < count; j++) {
if (strcmp(tempTeam[i], tempTeam[j]) > 0) {
                        char t[51];
strcpy(t, tempTeam[i]);
strcpy(tempTeam[i], tempTeam[j]);
strcpy(tempTeam[j], t);
                    }
                }
            }

            if (count > maxSize) {
maxSize = count;                 for (int i = 0;
i < count; i++)
```

```
strcpy(bestTeam[i], tempTeam[i]);         }
else if (count == maxSize && count > 0)
{
              int cmp = 0;
for (int i = 0; i < count; i++) {
                  cmp = strcmp(tempTeam[i], bestTeam[i]);
                   if (cmp < 0) {                        for
(int j = 0; j < count; j++)
strcpy(bestTeam[j], tempTeam[j]);
                       break;
} else if (cmp > 0)
break;
               }
           }
       }
   }

   printf("%d\n", maxSize);
for (int i = 0; i < maxSize; i++)
{        printf("%s",
bestTeam[i]);        if (i !=
maxSize - 1) printf(" ");
   }
   printf("\n");

   return 0;
```

}

Submission ID: 121790613

RESULT: Partially accepted Refer judge environment

Score	Time (sec)	Memory (KiB)	Language
90	0.13246	308	C

Input	Result	Time (sec)	Memory (KiB)	Score	Your output	Correct output	Diff
Input #1	Accepted	0.011935	2	10			
Input #2	Accepted	0.010257	2	10			
Input #3	Wrong answer	0.018096	2	0			
Input #4	Accepted	0.009664	2	10			
Input #5	Accepted	0.008867	2	10			
Input #6	Accepted	0.020268	2	10			
Input #7	Accepted	0.034634	308	10			
Input #8	Accepted	0.008937	2	10			
Input #9	Accepted	0.009806	2	20			

3

Code :

```
#include <stdio.h>

int countBought(int prices[], int n, long long wealth) {
    int count = 0;     for (int
i = 0; i < n; i++) {         if
(wealth >= prices[i]) {
wealth -= prices[i];
count++;
        }
    }
```

```
    return count;
}

int main() {
    int n, k;
    scanf("%d %d", &n, &k);

    int prices[n];     long
long sum = 0;     for (int i
= 0; i < n; i++) {
scanf("%d", &prices[i]);
sum += prices[i];
    }

    long long low = 0, high = sum, ans = -1;

    while (low <= high) {         long long
mid = (low + high) / 2;         int bought =
countBought(prices, n, mid);

        if (bought == k) {
ans = mid;
low = mid + 1;            }
else if (bought < k) {
low = mid + 1;           }
else {             high =
mid - 1;
```

```
        }
    }

    printf("%lld\n", ans >= 0 ? ans : 0);
return 0;
}
```

RESULT: Partially accepted — Refer judge environment

Score	Time (sec)	Memory (KiB)	Language
0	0.16874	572	C

Input	Result	Time (sec)	Memory (KiB)	Score	Your output	Correct output	Diff
Input #1	Accepted	0.009495	2	10			
Input #2	Accepted	0.010036	2	10			
Input #3	Accepted	0.012033	2	10			
Input #4	Accepted	0.010013	2	10			
Input #5	Wrong answer	0.009488	2	0			
Input #6	Wrong answer	0.01755	2	0			
Input #7	Wrong answer	0.017464	2	0			
Input #8	Wrong answer	0.028149	536	0			
Input #9	Accepted	0.028264	572	10			
Input #10	Accepted	0.026251	2	10			

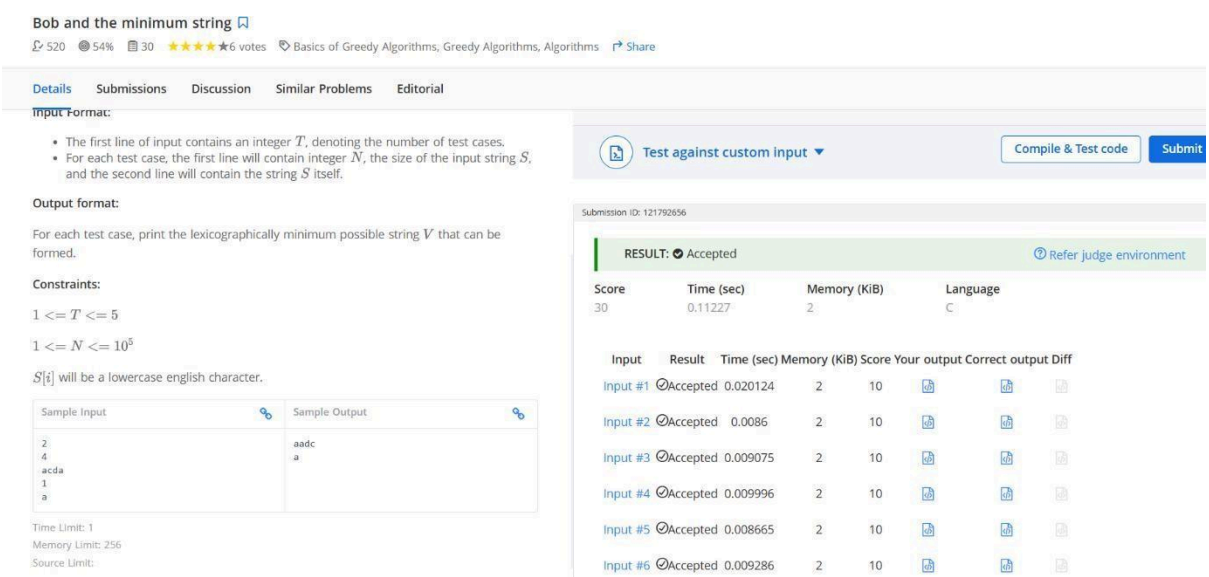
Q.3

Code:

```
#include <string.h>

#define MAX 100000
```

```
int main() {
    int t;
    scanf("%d", &t);
while (t--) {
        int n;         char
s[MAX];
scanf("%d", &n);
scanf("%s", s);
char
minSuffix[MAX];
minSuffix[n - 1] =
s[n - 1];         for (int
i = n - 2; i >= 0; i--) {
if (s[i] < minSuffix[i +
1])
minSuffix[i] = s[i];
else
minSuffix[i] =
minSuffix[i + 1];
        }

        char stack[MAX],
result[MAX];         int top = -1,
resIndex = 0;

        for (int i = 0; i < n; i++) {
stack[++top] = s[i];
```

```
            while (top >= 0 && (i == n - 1 || stack[top] <= minSuffix[i + 1])) {
result[resIndex++] = stack[top--];
            }
        }

        while (top >= 0)
result[resIndex++] = stack[top--];

        result[resIndex] = '\0';
printf("%s\n", result);
    }
    return 0;
}
```



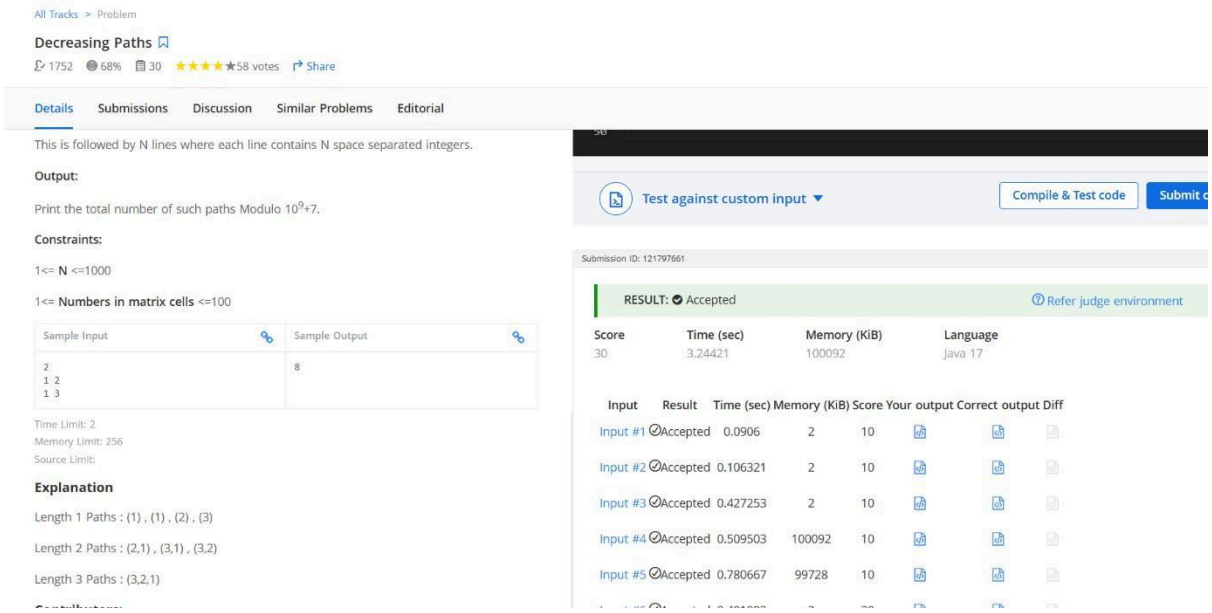
Q.4.

Code:

```
 import java.util.Scanner;
```

```
public class DecreasingPaths {

    static int N;     static int[][] mat;
static long[][] dp;     static final int
MOD = 1000000007;
    static int[] dx = {1, -1, 0, 0};
static int[] dy = {0, 0, 1, -1};

    static long dfs(int x, int y) {
if (dp[x][y] != -1) return dp[x][y];
long count = 1;         for (int dir =
0; dir < 4; dir++) {            int nx =
x + dx[dir];            int ny = y +
dy[dir];
            if (nx >= 0 && nx < N && ny >= 0 && ny < N && mat[nx][ny] <
mat[x][y]) {                count = (count + dfs(nx, ny)) % MOD;

            }
        }
        dp[x][y] = count;
return count;
    }

    public static void main(String[] args) {
        Scanner sc = new
Scanner(System.in);         N =
```

```
sc.nextInt();         mat = new int[N][N];
dp = new long[N][N];

        for (int i = 0; i < N; i++) {
for (int j = 0; j < N; j++) {
mat[i][j] = sc.nextInt();
dp[i][j] = -1;
            }
        }

        long total = 0;         for (int i =
0; i < N; i++) {             for (int j = 0; j
< N; j++) {                  total = (total +
dfs(i, j)) % MOD;
            }
        }

        System.out.println(total);
        sc.close();
    }
}
```

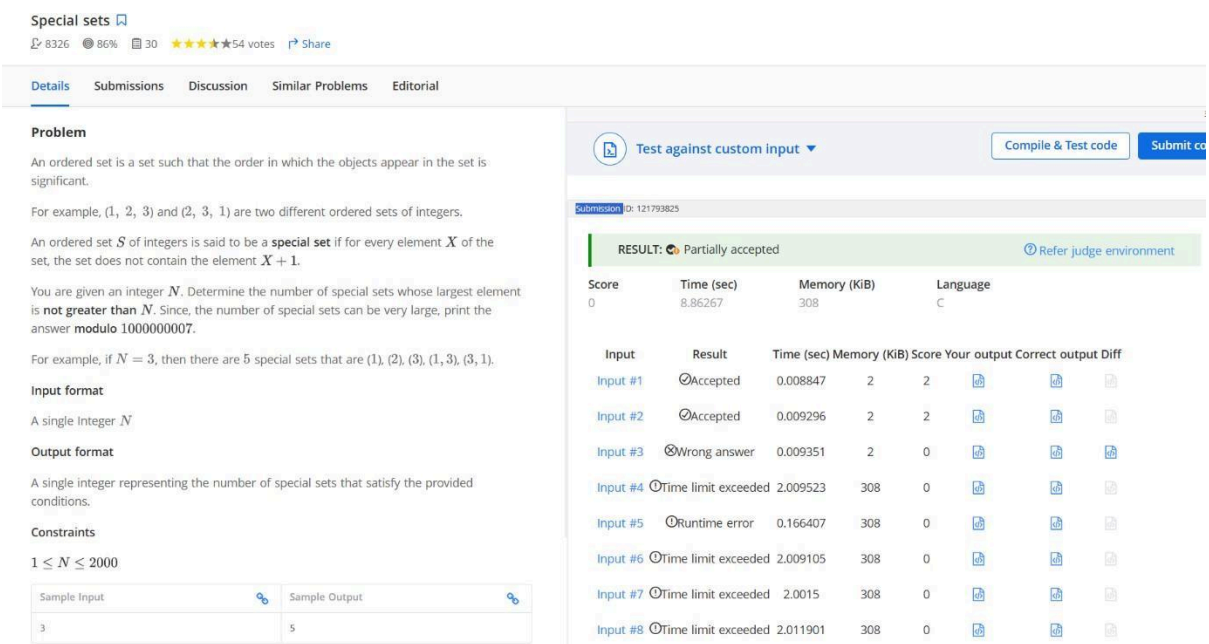


Q.5

Code:

```
#include <stdio.h>

#include <stdlib.h>

#include <math.h>


#define MOD 1000000007


int used[20];

int n; long long

ans = 0;


void dfs(int depth, int prev, int count)

{     if (count > 0) ans = (ans + 1) %

MOD;

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

{         if (!used[i]) {
```

```
            if (prev != -1 && abs(prev - i) == 1)
                continue;
            used[i] = 1;
dfs(depth + 1, i, count + 1);
used[i] = 0;
        }
    }
}

int main() {     scanf("%d", &n);
ans = 0;     for (int i = 0; i <= n;
i++) used[i] = 0;     dfs(0, -1, 0);
    printf("%lld\n", ans % MOD);
    return 0;
}
```



Special sets

8326 86% 30 ★★★★★54 votes Share

Details Submissions Discussion Similar Problems Editorial

Problem

An ordered set is a set such that the order in which the objects appear in the set is significant.

For example, $(1,\ 2,\ 3)$ and $(2,\ 3,\ 1)$ are two different ordered sets of integers.

An ordered set S of integers is said to be a **special set** if for every element X of the set, the set does not contain the element $X+1$.

You are given an integer N. Determine the number of special sets whose largest element is **not greater than** N. Since, the number of special sets can be very large, print the answer **modulo** 1000000007.

For example, if $N=3$, then there are 5 special sets that are (1), (2), (3), $(1,3)$, $(3,1)$.

Input format

A single Integer N

Output format

A single integer representing the number of special sets that satisfy the provided conditions.

Constraints

$1 \le N \le 2000$

Sample Input	Sample Output
3	5

Test against custom input ▾ Compile & Test code Submit co

Submission ID: 121793825

RESULT: Partially accepted Refer judge environment

Score	Time (sec)	Memory (KiB)	Language
0	8.86267	308	C

Input	Result	Time (sec)	Memory (KiB)	Score	Your output	Correct output	Diff
Input #1	Accepted	0.008847	2	2			
Input #2	Accepted	0.009296	2	2			
Input #3	Wrong answer	0.009351	2	0			
Input #4	Time limit exceeded	2.009523	308	0			
Input #5	Runtime error	0.166407	308	0			
Input #6	Time limit exceeded	2.009105	308	0			
Input #7	Time limit exceeded	2.0015	308	0			
Input #8	Time limit exceeded	2.011901	308	0			

6.

All Tracks > Basic Programming > Recursion > Recursion and Backtracking > Problem

Number of divisors

12651 696 30 ★★★★★ 93 votes Backtracking, Basic Programming, Number theory, Recursion Share

Details Submissions Discussion Similar Problems Editorial

Print the sum of these divisors.

Note: **k** is a prime number.

Input format

- The first line contains an integer T representing the number of test cases that will follow.
- Each test case consists of one line containing two integers n and k.

Output format

The output must contain the answer for each test case on a different line.

Each answer consists of a single integer.

Constraints

$T \leq 300000$
$1 \leq n \leq 1000000000$
$2 \leq k \leq 1000000000$

Sample Input	Sample Output
4 10 3 10 2 10 5 1000000000 97	41 36 43 494897959532893312

Time Limit: 1
Memory Limit: 64
Source Limit:

Explanation

In the first test case, f (x) from 1 to 10 is [1, 2, 1, 4, 5, 2, 7, 8, 1, 10], sum of which is 41.

In the second test case, f (x) from 1 to 10 is [1, 1, 3, 1, 5, 3, 7, 1, 9, 5].

Enter your code or Upload your code as file. Saved Python 3.8 (python 3.8.10)

```python
import sys
input = sys.stdin.readline

T = int(input())
for _ in range(T):
    n, k = map(int, input().split())

    if k == 1:
        print(n * (n + 1) // 2)
        continue

    total = 0
    while n > 0:
        q = n // k
        r = n - q * k
        total += (r + 1) * (n - r // 2)
        n //= k

    print(total)
```

7.

Problem List Submit 00:00:00

Description Editorial Similar Problems Submissions Discussion

Java 8 Auto

N-Queens

87% Success 18241 Attempts 20 Points 1s Time Limit 256MB Memory 1024 KB Max Code

Given a chess board having $N \times N$ cells, you need to place N queens on the board in such a way that no queen attacks any other queen.

Input:
The only line of input consists of a single integer denoting N.

Output:
If it is possible to place all the N queens in such a way that no queen attacks another queen, then print N lines having N integers. The integer in i^{th} line and j^{th} column will denote the cell (i, j) of the board and should be 1 if a queen is placed at (i, j) otherwise 0. If there are more than way of placing queens print any of them. If it is not possible to place all N queens in the desired way, then print "Not possible" (without quotes).

Constraints:
$1 \leq N \leq 10$.

Examples

```java
import java.util.*;

public class Main {
    static int N;
    static int[][] board;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        N = sc.nextInt();
        board = new int[N][N];

        if (solveNQueens(0)) {
            printBoard();
        } else {
            System.out.println("Not possible");
        }
    }

    // Try to place queens row by row
    static boolean solveNQueens(int row) {
        if (row == N) {
```

Results

Accepted

Refer judge environ

Time	Memory	Language
0.085s	83408KiB	Java 8

Input

```
4
```

Output

```
0 1 0 0
0 0 0 1
1 0 0 0
0 0 1 0
```

8.

X Subarrays

55% Success | 1039 Attempts | 30 Points | 1s Time Limit | 256MB Memory | 1024 KB Max Code

You are given an array A of length N sorted in non-decreasing order and an integer X. Find the number of subarrays such that the sum of the minimum and maximum element of that subarray is less than or equal to X.

Input Format:

- The first line contains an integer T, which denotes the number of test cases.
- The first line of each test case contains two integers, N and X.
- The next line of each test case contains N space-separated integers, elements of array A.

Output Format:

For each test case, print the number of subarrays such that the sum of the minimum and maximum element of that subarray is less than or equal to X.

Constraints:

$1 \le T \le 10$

$1 \le N \le 10^5$

$1 \le A[i], X \le 10^5$

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();  // number of test cases

        while (t-- > 0) {
            int n = sc.nextInt();  // size of array
            int x = sc.nextInt();  // target sum
            int[] arr = new int[n];
            for (int i = 0; i < n; i++) arr[i] = sc.nextInt();

            long count = countSubarrays(arr, n, x);
            System.out.println(count);
```

Results

Log ID: 284117539 / 27/10/2025,18:34:29

Accepted

Refer judge environment

Time | Memory | Language

9.

Problem List

Submit 00:00:00

Description | Editorial | Similar Problems | Submissions | Discussion

Make Palindrome

55% Success | 5815 Attempts | 20 Points | 1s Time Limit | 256MB Memory | 1024 KB Max Code

There is a string that consists of lowercase english alphabets and we have to make the string a palindrome
For this, we have 2 types of operations
1. Re-arrange the string however you want (this operation is free)
2. Add a new character to the end of the string (this operation will cost 1Re)

Your task is to calculate the minimum amount of money to make the string a palindrome

Input Format:
The first line contains one integer T the number of test cases
The first line of every test case consists of N, the length of the string
The second line of every test case contains string S

Output Format:
For every test case output one integer, the minimum amount of money required to make the string a palindrome

Constraints:

Java 8 | Auto

```java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt(); // number of test cases

        while (t-- > 0) {
            int n = sc.nextInt();
            String s = sc.next();

            int cost = minCostToMakePalindrome(s);
            System.out.println(cost);
        }
    }
```

Results

Log ID: 284117681 / 27/10/2025.18:36:50

Accepted | Refer judge environment

Time	Memory	Language
0.050s	83192KiB	Java 8

10.

All Tracks > Problem

Details | Submissions | Discussion | Similar Problems | Editorial

Problem

Given an array of integers where each element represents the max number of steps that can be made forward from that element. Write a function to return the minimum number of jumps to reach the end of the array (starting from the first element). If an element is 0, they cannot move through that element. If the end isn't reachable, return -1.

Examples:

Input: arr[] = {1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9}

Output: 3 (1-> 3 -> 9 -> 9)

Explanation: Jump from 1st element

to 2nd element as there is only 1 step,

now there are three options 5, 8 or 9.

If 8 or 9 is chosen then the end node 9

can be reached. So 3 jumps are made.

Input: arr[] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}

Enter your code or Upload your code as file. | Save | Java 8 (openjdk 1.8.0_241)

```java
import java.io.*;
import java.util.*;

public class TestClass {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        PrintWriter wr = new PrintWriter(System.out);
         int N = Integer.parseInt(br.readLine().trim());
         String[] arr_P = br.readLine().split(" ");
         int[] P = new int[N];
         for(int i_P = 0; i_P < arr_P.length; i_P++)
         {
             P[i_P] = Integer.parseInt(arr_P[i_P]);
         }

         long out_ = alien_attack(N, P);
         System.out.println(out_);

         wr.close();
         br.close();
    }
    static long alien_attack(int N, int[] P){
```

35:1 vscode