



CODEBREW - FRESHERS' CONTEST 2022

Post-contest Meet + Solution Discussion

Final Standings



#	Who
1	AakashBitsPilani
2	notposeidon
3	 krishna_0_9
4	Bhaskar1702
5	devk27
6	Saksham0109
7	CaptainUknown
8	akshatbajpai19
9	Lakshit_Sethi
10	mallika_ajmani

Top Freshers:

1. Harshith Vasireddy
2. Bhaskar Iyer
3. Dev Kulkarni
4. Saksham Verma
5. Abir Abhyankar
6. Akshat Bajpai
7. Lakshit Sethi
8. Mallika Ajmani

Top Seniors:

1. Aakash Gupta
2. Krishna Jain

Problem A: BCD!



Problem Statement (Simplified)

1. Given N objects, you have to find the number of containers such that all the objects can be fitted into the containers.
2. Constraint - a container can hold maximum of K objects.

Solution Idea

- To minimize the number of containers used - fill each container till it's maximum capacity.
- Fill first K books in first container, next K in second container, and so on. Total number of containers needed is ceil of (N/K) .

Problem A: BCD!



Code Implementation

```
int n , k;  
cin >> n >> k;  
int num = n / k;  
if(n % k != 0) {  
    num++;  
}
```

Problem B: Poku's Vacation!



Problem Statement (Simplified)

- Using a maximum of N bricks, build a staircase such that the stairs are in increasing order of heights.
- Maximize the number of stairs.

Solution Idea

- Consider the condition for maximising the number of stairs. We keep the height of each stair as small as possible.
- This can be achieved when for every stair i , $h[i] = h[i - 1] + 1$.
- So the staircase will be of the form $1, 2, 3, 4, \dots$ and so on.
- The number of bricks used, $1 + 2 + 3 + 4 + \dots k \leq n$.
- This simplifies to, $k*(k - 1) / 2 \leq n$.
- We can find this by using the quadratic formula, or just by simulating the entire process inside a while loop!

Problem B: Poku's Vacation!



Code Implementation -

```
int n;  
  
cin>>n;  
  
int cnt=0, add=1, sum=0;  
while((sum+add)<=n)  
{  
    sum+=add;  
    add++;  
    cnt++;  
}
```

Problem D: Elder Ning



Problem Statement (Simplified)

- There are M ranges of the form $[L, R]$.
- We have to find, how many numbers come inside ALL the ranges?

Solution Idea

- To find the numbers that come inside ALL the ranges, we have to find the intersection of the ranges.
- A number X will be contained in every range $[L, R]$, if $X \geq L$ and $X \leq R$.
- To find the intersection, we can simply take the maximum of all L and the minimum of all R .
- Answer will be $(R - L + 1)$, or 0 if R is less than L .

Problem D: Elder Ning



- **Code Implementation**

```
int n, m;

cin >> n >> m;

int l[m], r[m];

for(int i = 0; i < m; i++){
    cin >> l[i] >> r[i];
}
```

```
int L = 10000000000;

int R = -1;

for(int i = 0; i < m; i++){
    L = min(L, l[i]);
    R = max(R, r[i]);
}

if(L > R) cout << 0;
else cout << R - L + 1;
```

Problem E: Hostel Cleaning



Major hints from the problem statement

- We have to minimize the number of sweepers.
- K is a factor of N .

Solution Idea

- The hints should be enough to realize that exactly N/K sweepers will be hired. Each of them will be at a distance of exactly K from each other.
- So there are simply K different “sets” of sweepers that can be hired. We can find the cost of each of these sets, and print the minimum cost.

Problem E: Hostel Cleaning



- **Code Implementation [Approach by :notposeidon]**

```
int n, k;
cin >> n >> k;
int arr[n];
for(int i = 0; i < n; i++) {
    cin >> arr[i];
}
```

```
int ans[k];
for(int i = 0; i < n; i++) {
    ans[i % k] += arr[i];
}

// x → minimum value in ans
cout << x;
```

Problem F: No Internet IPC



Solution Idea

Let number of patch cables = **P** and number of computers in which updates have been installed = **C**.

At any time, there can be 2 possibilities:

1. **$P \geq C$** . In that case, just connect as many computers as possible. The number of installed computers will double by the end of this step. (**$C = C * 2$**).
2. **$P < C$** . In this case, the number of installed computers will be increased by the number of patch cables. (**$C = C + P$**).

Initially, **$C = 1$** .

We can simulate this process in a **while loop**.

Problem F: No Internet IPC



- **Code Implementation**

```
int curr = 1, hours = 0, patch, n;  
cin >> n >> patch;  
while(curr < n){  
    curr += min(curr, patch);  
    hours++;  
}  
cout << hours << "\n";
```

Problem C and G: Dualites in Pain



Problem Statement (Simplified)

So basically, you have an array A of N integers and you can do the following:

1. In one operation you can select the maximum element present in the array and decrease its value by 1.
2. You cannot use the same index i for 2 consecutive operations.

Your task is to make all the elements of array A equal to 0 using the above operation multiple times.

Solution (Idea)

Let's forget about lexicographically least answer (for Problem G). We'll deal with it later.

- ❖ So, firstly let's sort the array. Now, for the first operation we can pick the last element A_N (consider indices start from 1) and decrease it by 1.
- ❖ In the next operation obviously we cannot choose A_N again so, in order to perform more operations A_{N-1} should become the largest element in the array after 1 operation which is only possible when initially $A_{N-1} = A_N - 1$ OR $A_{N-1} = A_N$.
- ❖ If the above condition is true, we can use A_N and A_{N-1} alternately for performing the operations till we reach the point where $A_N = A_{N-1} = A_{N-2}$. Then we can use A_N, A_{N-1}, A_{N-2} repeatedly till $A_N = A_{N-1} = A_{N-2} = A_{N-3}$ and so on till all the elements are equal. Now, you can simply choose $A_N, A_{N-1}, \dots, A_1$ repeatedly until all the values become 0.



Solution (Implementation)

For Problem C, you just had to output if it is possible to make all the values equal to 0. From the **Solution (Idea)** we can already see that if $A_{N-1} = A_N - 1$ OR $A_{N-1} = A_N$ holds true then it is always possible to make all the values in array A equal to 0.

For Problem G, firstly we will print -1 for the case when its impossible to complete the task (Problem C). Now what is left for us is to implement our Solution (Idea).

So until $A_N > A_{N-1}$ we will output the index of A_N and decrement A_N by 1. When $A_N = A_{N-1}$, we output the indices (in sorted order for lexicographically least answer) of A_N and A_{N-1} repeatedly and decrement both A_N and A_{N-1} in each step until $A_N = A_{N-1} = A_{N-2}$ and so on. So our answer will be something like this:

Indices of

$A_N, A_N, \dots$	$(A_N - A_{N-1} \text{ times})$ in sorted order , (≤ 1 time if answer is not -1),
$\{A_N, A_{N-1}\}, \{A_N, A_{N-1}\}, \dots$	$(A_{N-1} - A_{N-2} \text{ times})$ in sorted order,
$\{A_N, A_{N-1}, A_{N-2}\}, \{A_N, A_{N-1}, A_{N-2}\}, \dots$	$(A_{N-2} - A_{N-3} \text{ times})$ in sorted order,
$\dots\dots\dots$	
$\{A_N, A_{N-1}, \dots, A_1\}, \{A_N, A_{N-1}, \dots, A_1\}, \dots$	$(A_1 - A_0 \text{ times where } A_0 = 0)$ in sorted order.

For storing the indices of elements and you can input the original array as pairs of {element, index} like this:
arr = {{a₁, 1}, {a₂, 2} {a_N, N}} and add {a₀, 0} for convenience. Now you can sort the array.

Example and Code Implementation



Example: Let's take input array as {4, 2, 1, 4, 2}. Then, $arr = \{\{0,0\}, \{4, 1\}, \{2, 2\}, \{1, 3\}, \{4, 4\}, \{2, 5\}\}$ — (sort) — $\{\{0,0\}, \{1, 3\}, \{2, 2\}, \{2, 5\}, \{4, 4\}, \{4, 1\}\}$

```
pair<int, int> arr[n+1];
arr[0] = {0,0};
for(int i = 1; i <= n; i++) cin >> arr[i].first, arr[i].second = i;
sort(arr, arr + n + 1); // Check for -1 condition after this.
```

First iteration: store index of arr_5	=	$\{4\}$ — (sort) — $\{4\}$	Print this array of indices $[4 - 4 = 0 \text{ times}]$
Second iteration: store indices of arr_5 and arr_4	=	$\{4, 1\}$ — (sort) — $\{1, 4\}$	Print this array of indices $[4 - 2 = 2 \text{ times}]$
Third iteration: store indices of arr_5, arr_4, arr_3	=	$\{4, 1, 5\}$ — (sort) — $\{1, 4, 5\}$	Print this array of indices $[2 - 2 = 0 \text{ times}]$
Fourth iteration: store indices of $arr_5, arr_4, arr_3, arr_2$	=	$\{4, 1, 5, 2\}$ — (sort) — $\{1, 2, 4, 5\}$	Print this array of indices $[2 - 1 = 1 \text{ times}]$
Fifth iteration: store indices of $arr_5, arr_4, arr_3, arr_2, arr_1$	=	$\{4, 1, 5, 2, 3\}$ — (sort) — $\{1, 2, 3, 4, 5\}$	Print this array of indices $[1 - 0 = 1 \text{ time}]$
Final Answer - 1 4 1 4 1 2 4 5 1 2 3 4 5			

```
for(int i = n; i >= 1 ;i--) {
    int indices[n - i + 1];
    for(int j = n; j >= i; j--) indices[n - j] = arr[j].second;
    sort(indices, indices + n - i + 1);
    for(int j = 1; j <= arr[i].first - arr[i - 1].first; j++) {
        for(int k = 0; k < n - i + 1; k++) cout << indices[k] << " ";
    }
}
```

Problem H: Perfect Array



Solution Approach

- **Key idea:** Solve the problem in reverse, i.e. from last element inserted to first.
- The problem can be rephrased as follows: we have N elements in a row. If the k^{th} element from the left is equal to k , then we can remove this element from the sequence.
- We have to remove the rightmost element that can be removed. If we don't remove this element right now, then it will be impossible to remove it ever again! (Think why)
- Since constraints are low, the problem can be simulated in $O(N^2)$.

Problem H: Perfect Array



- **Code (Pseudocode)**

Arr → Array of integers

N → Length of array

Ans → Empty array

for i in range(N):

*Find the right-most element such that **Arr**[i] == $i + 1$*

*Remove this element from **Arr**, and insert it into **Ans***

*If no element exists → Output **NO** and exit program.*

*Output **YES** and the elements of **Ans** in reverse order.*



THANK YOU!

See you next month for the second Freshers' contest!