ESO207A – Data Structures and Algorithms

Indian Institute of Technology Kanpur

Programming Assignment 4

Important guidelines for submission

- It is only through the assignments that one learns the most about the algorithms and data structures. You are advised to refrain from searching for a solution on the net or from a notebook.
- Refrain from collaborating with the students of other groups or your friends. If any evidence is found that confirms copying, the penalty will be very harsh. Refer to the website at the link: https://cse.iitk.ac.in/pages/AntiCheatingPolicy.html regarding the departmental policy on cheating.
- **The assignment is to be submitted on Gradescope. You must submit one C file named as *assign4.c* for questions 1,2 and 3. This is EXTREMELY IMPORTANT. The code is checked with an autograder, and a different file name will fail to pass the test cases.**
- **You must submit a .txt file for Question 4 named as assign4.txt.**
- **DEADLINE** for this assignment submission is 11 PM Friday (11th April).
- In case of any issue related to this assignment, send an email at geetika@cse.iitk.ac.in or at nsandeep21@iitk.ac.in

Tasks to be done

1 The Game of Numbers

1.1 Problem Statement

One fine day, bored out of your mind, you called your two closest friends— Ram and Shyam to play a game. The rules of game are simple.
The Rules:
1. Pick Your Number:

- Each of you had to choose a number between 1 and 10^9.
- The order is strict: Ram goes first, then you, then Shyam.
- No two players could have the same number. If Ram picked 7, then you had to pick something other than 7.

2. Your other friend:

- After the numbers were locked in, one of your other friend would display n ($\leq 10^6$) numbers.
- The winner? Whoever locked number occurred more than one-third of the total numbers in the display.
- You have bribed your friend so that you could see the sequence of numbers beforehand.

Your Mission:
After seeing the numbers, you had to decide:
Can you always win, no matter what Ram and Shyam pick?

- If YES, you had to choose two numbers so that:
- If Ram took one, you'd pick the other.
- If Ram ignored both, you could pick either.
- If NO (meaning your victory depended on their choices, or you just couldn't guarantee a win).

1.2 Input Format

- The first line contains an integer t - total number of test cases.
- Then for each test case:
  - The first line contains an integer n - the total number to be displayed.
  - The second line contains n numbers each denoted by $arr[i]$ separated by whitespace.

1.3 Output Format

- For each test case:
  - The first line contains an "YES"/"NO" - (in capital letters).
  - If the first line is "YES" then second line contains two number separated by whitespace in increasing order.

1.4 Constraints

- $1 \leq t \leq 10$
- $1 \leq n \leq 10^6$
- $1 \leq arr[i] \leq 10^9 \ \forall i$

1.5 Example1

Input

```
1
5
1 1 2 2 3
```

Output

```
YES
1 2
```

Explanation Both 1 and 2 occur 2 times which is greater than 5/3(n/3) times. If Ram choose 1 then you can choose 2. Note that here Ram will also win which is fine, but it is guaranteed that you will always win in this case.

1.6 Example2

Input

```
2
8
1 1 2 2 4 3 1000000 1888972
9
3 3 3 3 3 2 87 9 10
```

Output

```
NO
NO
```

Explanation For first test case no number occured more than $n/3$ times.
For second test case only 3 occured more than $n/3$ times. Therefore if Ram choose 3 you will loose, so your win depends on his choice.

1.7 Hints

- Think about numbers that appear way more often than others... and how you can exploit that.
- Preferred time complexity - $O(n)$ and space complexity - $O(1)$ excluding the space required to store the initial array.

2 Bridge Detection in an Undirected Graph

Problem Statement

You are given an undirected connected graph with **n** vertices and **m** edges. Your task is to determine, for each edge, whether removing it would disconnect the graph. In other words, identify all **bridges** in the graph.

A bridge (or cut-edge) is an edge that, if removed, increases the number of connected components in the graph.

Input Format

- The first line contains two integers **n** and **m** — the number of vertices and edges in the graph.
- The next **m** lines each contain two integers **u** and **v** — representing an undirected edge between vertex **u** and vertex **v**.

Output Format

Print all the bridges that, if removed, can change the number of connected components.

- For each bridge, print a line containing two integers **u** and **v**, where $u < v$, representing the edge.
- Print all bridges in increasing order, sorted first by **u**, then by **v**.
- If there are no bridges, print a single line with `-1`.

Constraints

- $1 \leq n \leq 10^5$
- $0 \leq m \leq 10^5$
- $0 \leq u, v \leq n - 1$
- There are no parallel edges or self-loops.
- The graph is connected.

Example

Input

```
5 5
0 1
0 2
2 3
2 4
3 4
```

Output

```
0 1
0 2
```

Explanation

The given graph is connected.
The edges $(0, 1)$ and $(0, 2)$ are bridges, since removing them would increase the number of connected components in the graph.

3 Merging K sorted arrays

3.1 Problem Statement

You are given k sorted integer arrays. Your task is to efficiently merge them into a single sorted array. Since the arrays are already individually sorted, design an algorithm that minimizes the total time complexity of merging.

3.2 Input Format

- The first line contains a single integer k — the number of arrays.
- Each of the next k lines contains:
  - An integer n_i — the number of elements in the i-th array.
  - Followed by n_i integers sorted in non-decreasing order.

3.3 Output Format

Print all elements from the merged array in non-decreasing order, space-separated. Duplicates are allowed.

3.4 Constraints

- $1 \leq k \leq 1000$
- $1 \leq n_i \leq 1000$
- Total number of elements across all arrays $\leq 10^5$
- $-10^9 \leq a[i] \leq 10^9$

3.5 Example

Input

```
3
3 1 4 5
2 2 6
1 0
```

Output

```
0 1 2 4 5 6
```

Explanation

```
First array is 1 4 5
Second array is 2 6
third array is 0
so combined 0 1 2 4 5 6
```

3.6 Hints

- Time complexity of the optimal solution should be $\mathcal{O}(n \log k)$, where n is the total number of elements across all arrays.

4 Sorting

4.1 Problem Statement

This is a subjective problem and you are free to use Google, GenAI tools like ChatGpt etc. We know that quicksort has a worst-case time complexity of $O(n^2)$, while mergesort has a worst-case complexity of $O(nlogn)$. Does this mean mergesort is always faster in practice? For this analysis, disregard all assumptions made in the course's introductory lectures. Conduct a thorough examination to determine which algorithm performs better under what circumstances and why. (Word limit: 400 words)

4.2 Key Points to think Upon

- Unstable vs stable sort.
- in-place and out-of place memory usage.
- How caching effect affect the performance.
- What we generally ignore when comparing Big(O) notations.
- What will happen in real-world scenarios.

4.3 Recommended resource

- https://stackoverflow.com/questions/680541/quick-sort-vs-merge-sort
- How standard c++ and python libraries implement sorting and why? Introsort vs Timsort

Test Cases

Your solution will be evaluated against multiple hidden test cases. These test cases are not visible to you during the submission period. To receive full marks, your code must produce outputs that exactly match the expected results for all test cases.. You will not see your score until the deadline is over when we grade your code using autograder, and publish the result. You may resubmit as many times as you want, but only the final submission will be graded.