NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD CAMPUS

CS-2009 Design and Analysis of Algorithms - 2025

ASSIGNMENT-02

Instructions:

Programming & Submission Guidelines:

1. **Separate C++ files for each question:**
   - Name your files in the format: Q_<question_number>_<roll_number>.cpp
   - Example: **Q_1_22I-2345.cpp**
2. **Submission on Codeforces:**
   - The **last two questions** must be submitted on **Codeforces** using the provided link in the assignment document.
   - Codeforces submission will close on the deadline, so submit in time.
3. **Report Submission Requirements:**
   - A report must be submitted in **PDF format** along with the code.
   - The report should include:
     - Explanation of the algorithm for the all questions
     - Transition diagrams and tables (if applicable)
     - Time complexity analysis
   - Diagrams can be created either:
     - By hand (scanned and inserted into the report)
     - Using any online tool like **Figma, Canva, or any diagramming software**
4. **Final Submission on Google Classroom:**
   - Submit a ZIP file containing:
     - All five C++ files
     - The report in PDF format
     - ZIP file naming format: A2_<roll_number>.zip Example: A2_22I-2345.zip
   - Submit the ZIP file on **Google Classroom** before the deadline.

Important Notes:

✔ Follow the exact file and ZIP naming conventions to avoid marks deduction.

✔ Ensure all submissions (Codeforces and Google Classroom) are completed before the deadline.

✔ Any missing component (e.g., Codeforces submission, report, C++ files) will lead to deduction or disqualification.

Deadline: March 31th – No extensions will be granted.

Problem - 1

Implementing a d-Ary Heap

Objective: The purpose of this question is to understand and implement a **d-ary heap**, which is a generalization of the binary heap. In a binary heap, each node has at most 2 children. In a **d-ary heap**, each node can have up to **d children**. The value of **d** will be taken as an input, making the implementation flexible for different values of d.

Problem Statement:

You are required to implement a **generic d-ary heap** that supports the following operations:

1. **Heap Construction**: Read a list of numbers from a file and insert them into a d-ary heap.
2. **Insert Operation**: Insert a new element while maintaining the heap property.
3. **Delete Operation**: Remove the root element and restructure the heap accordingly.
4. **Heapify Operation**: Ensure the heap property is maintained after insertion or deletion.
5. **Print Heap**: Display the heap structure in a level-wise format.

Input Format:

1. The first line of input contains an integer d, representing the number of children per node, followed by a heap type (min or max).
2. The subsequent lines contain operations:
    - "ins": This keyword is followed by a space-separated list of integers to be inserted into the heap.
    - "extract": This keyword indicates the extraction of the root element from the heap.
3. The "ins" and "extract" operations can appear multiple times in sequence.

Output Format:

- Display the final heap, with each line representing the elements at a specific level.

Example Test Case:
Input File:

```
2 max
ins
10 20 5 30 15 40 50
extract
extract
extract
ins
45 85 29
```

Expected Output:

```
85
20     45
10   15  5    29
```

Submission Guidelines:

- The program should read input from a file and allow dynamic **d** values.
- The code should be properly commented and modular.
- Implement **extract_min()** or **extract_max()** depending on whether it is a Min-Heap or Max-Heap.
- Submit a handwritten file report explaining your approach and data structures used.

Problem - 2

Enhanced Finite Automata String Matching with Wildcards

The **Finite Automata String Matching Algorithm** is a powerful technique for efficient pattern searching. In this assignment, you will enhance it by implementing **wildcard support (?)**, which can match any single character.

Your task is to:

1. **Build a finite automaton** that can search for multiple patterns within a text.
2. **Support the ? wildcard**, meaning "a?ple" should match both "apple" and "ample".
3. **Process multiple patterns** and search for them in multiple lines of text.
4. **Display search results in a structured format.**

Input Format:

1. The first line contains an **integer p**, the number of patterns.
2. The next p lines each contain a **pattern** that may include a ? wildcard.
3. The next line contains an **integer m**, the number of **text lines** to be searched.
4. The next m lines contain the **text** in which the patterns should be searched.

Output Format:

- For each text line, display **all occurrences** of the given patterns, specifying:
  - **The matched pattern** (including wildcard substitution).
  - **The exact matched word in parentheses**.
  - **The starting index** of each match in the text.
- If a pattern is **not found**, print "No match found".

Example Test Case:

Input File:

```
3
a?ple
banana
g?ape
2
I love eating an Apple and a Banana.
Grapes are my favorite fruit, but I also like banana and apple.
```

Expected Output:

```
Matches in Line 1:
a?ple: apple at index 17
banana: banana at index 29

Matches in Line 2:
g?ape: grape at index 0
banana: banana at index 46
a?ple: apple at index 57
```

Programming Constraints:

1. **Use the Finite Automata approach** (not brute force or KMP).
2. **Support ? as a wildcard**, which can match any single character.
3. **Efficiently process multiple patterns** in a single pass of the text.

Report Requirements:
Each student must submit a report containing the following:

1. Introduction
- Explain how the **Enhance Finite Automata String Matching Algorithm** works.
- Describe how **wildcards (?)** were incorporated into the automaton.

2. Transition Table and Diagram
- Draw the **finite state transition diagram** with **transition table** for **a?ple** and **g?ape** of the patterns.

3. Time Complexity Analysis
- Explain the **preprocessing time complexity** for building the transition table.
- Analyze the **searching time complexity** in terms of n (text length) and m (pattern length).

4. Code Explanation
- Provide a brief explanation of your logic and implementation.

Problem - 3

A well-known organization, **PakFlora**, hosts an annual competition that evaluates home and community gardens from various regions of Pakistan. Each garden is assigned a score, represented by **si**, which is a positive integer reflecting its rating. Additionally, every garden is given a unique identifier **ri**, a positive integer serving as its registration number.

a. In an effort to promote inclusivity and reduce competitive pressure, **PakFlora** has decided to award identical trophies to the top k gardens. Given an unsorted list A containing garden pairs, and a positive integer k such that $k \leq |A|$, outline an algorithm (don't write code) that runs in $O(|A|+k\log|A|)$ time to return the registration numbers of the k highest-scoring gardens, resolving any ties arbitrarily.

b. **PakFlora** has decided to take a more objective approach and award a trophy to every garden with a score strictly greater than a reference score x. In a max-heap structure A that holds pairs of garden attributes, where each pair includes a numerical score and a corresponding unique registration number, describe a method and write pseudocode that runs in O(nx) time to extract the registration numbers of all gardens that have scores exceeding a given threshold value x. Here, nx represents the count of gardens that fulfill this scoring criterion.

Problem - 4

Hadi, a production manager at a cutting-edge electronics factory, is responsible for ensuring that every gadget coming off the assembly line meets strict quality standards. Each gadget is assigned a unique quality score, and in an ideal production sequence these scores should appear in non-decreasing order. However, on a particularly hectic day, a production glitch scrambled the order of the gadgets' quality scores.

To assess the situation, Hadi decides to quantify the level of disorder by counting the number of times a gadget with a higher quality score appears before a gadget with a lower score. This count of ordering discrepancies will help him understand how far the production sequence deviated from the ideal.

Your task is to design an efficient algorithm that computes this disorder count for a given production sequence.

Input Format

The first line contains the number of test cases - **t (1 ≤ t ≤ 100)** .

First line of Each test case contains the number of products **n (1 ≤ n ≤ 10^6)**.

The second line contains ***n*** distinct integers representing the quality scores of the gadgets in the order they were produced.

Constraints

t (1 ≤ t ≤ 100) .

n (1 ≤ n ≤ 10^6).

scores (0 < scores < 10^6)

Output Format

Print a single integer representing the total count of ordering discrepancies in the production sequence. If there are none, output "0".

Example Input and Output:

Input:
2
5
5 1 3 7 2
10
8 4 2 1 5 3 7 6 10 9
Output:
5
14

Problem - 5

Ammar owns a mobile phone shop, and he has a big pile of phones waiting to be fixed. Each phone in the pile takes a certain amount of time to repair. Ammar doesn't repair the phones himself. Instead, he sends them to several nearby repair shops. These repair shops all start working at the same time, and each shop can only fix one phone at a time. Also, once a shop finishes fixing a phone, it takes the next phone from the top of the pile. Ammar must send the phones in the order they appear in the pile.

Ammar wants all the phones fixed as quickly as possible. The overall repair time is the time when the last shop finishes fixing its assigned phones. Your task is to help Ammar figure out how to split the pile of phones among the repair shops so that this overall repair time is as short as possible.

Input Format:

- The first line contains number of test cases **t.**
- The first line of each test case contains two numbers: **n** (the number of phones) and **K** (the number of repair shops).
- The second line of each test case has **n** numbers, each representing the time needed to repair a phone in the order they are in the pile (from top to bottom).

Output Format:

- output a single integer representing the minimum possible total repair time.

Constraints:

- 0< n <10^6
- 0< k < 1001
- 1≤ repair time for each phone <10^6

Example Input and Output:
Input:
1
10 3
8 4 2 1 5 3 7 6 10 9
Output:
20

Explanation:
- The first repair shop fixes phones with times: 8, 4, 2, 1, 5 (total time = 20).
- The second repair shop fixes phones with times: 3, 7, 6 (total time = 16).
- The third repair shop fixes phones with times: 10, 9 (total time = 19).

NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES ISLAMABAD CAMPUS