M22CS1.304 Data Structures and Algorithms for Problem Solving

Assignment 4

Deadline: 11:59 pm. November 1, 2022

Important Points:

1. Only C/C++ is allowed.
2. Directory Structure:

```
2021201004_A4
     |____ 2021201004_A4_Q1
               |____ 2021201004_A4_Q1a.cpp
               |____ 2021201004_A4_Q1b.cpp
               |____ 2021201004_A4_Q1c.cpp
     |_____ 2021201004_A4_Q2
               |____ 2021201004_A4_Q2a.cpp
               |____ 2021201004_A4_Q2b.cpp
     |_____ 2021201004_A4_Q3.cpp
     |_____ 2021201004_A4_Q4.cpp
```

   Replace your roll number in place of `2021201004`

3. Submission Format: Follow the above-mentioned directory structure and zip the **`RollNo_A4`** folder and submit **`RollNo_A4.zip`** on moodle.
   Note: All submissions which are not in the specified format or submitted after the deadline will be awarded **0** in the assignment.
4. C++ STL is **not allowed** for any of the questions unless specified otherwise in the question. So "`#include <bits/stdc++.h>`" is not allowed.
5. You can ask queries by posting on moodle.

Any case of plagiarism will lead to a 0 in the assignment or an "F" in the course.

1. Suffix Array

Implement a Suffix Array that is capable of performing the following operations on strings in the most efficient way.

a. Given a string **S** print its minimum lexicographic rotation.

Input Format

A single line that contains a string **S**.

Output Format

A single line containing a string that represents the minimum lexicographic rotation of the input string.

Constraints

1 <= len(S) <= 10^5

Sample Input

```
dcabca
```

Sample Output

```
abcadc
```

Explanation

All possible rotations are "dcabca","cabcad","abcadc", "bcadca","cadcab","adcabc".

Among all lexicographically minimum is "abcadc" .

Expected Time Complexity:- O(nlogn).

b. Given an integer **K** and a string **S**, print the length of the longest substring that appears in the text at least K times.

Input Format

The first line contains a single integer **K.**

The Second line contains a string **S**.

Output Format

A single line containing a single integer representing the length of the longest substring.

If no such substring exists, print **-1**.

Constraints

1 <= len(S) <= 10^5

1 <= K <= len(S)

Sample Input

```
dcabca
2
```

Sample Output

```
2
```

Explanation

"ca" is a substring that appears twice and its length is 2, so the answer is 2.

Expected Time Complexity:- O(nlogn).

c. Given a string **S,** Your task is to find out the longest substring that is also a palindrome. If multiple solutions exist, print the lexicographically smallest palindrome.

Input Format

A single line that contains a string **S**.

Output Format

A single line containing a string that represents the longest substring that is a palindrome.

Constraints

1 <= len(S) <= 10^5

Sample Input

```
dcabca
```

Sample Output

```
a
```

Explanation

Only length 1 substring is palindromic, and among them "`a`" is lexicographically smallest, hence the answer is "`a`".

Expected Time Complexity:- O(nlogn).

Note: For all the above parts, the input string **S** contains ASCII lowercase letters only.

2. Unordered Map

Aim: To learn how Hashing works and the importance of the Hash Function.

a. Functions to implement:

insert(key, value) - insert the key-value pair. If the key already exists, update the value corresponding to the given key.
erase(key) – erase if the key is present otherwise do nothing.
find(key) – returns true if the key exists otherwise returns false.
map[key] – returns the value mapped to the key. If the key does not exist, return the default value of the datatype as the mapped value to the key.

Note:

- Your unordered map should be generic i.e. it should be able to handle **(int, float, char, double, string)** data types.
- You have the liberty to choose between open addressing and chaining to handle hash collisions.

Input Format

First Line will contain a single integer **q** denoting the number of queries.
The next lines for each query follow the following pattern:
The first line will contain a single integer denoting the type of operation.
[1 -> insert, 2 -> erase, 3 -> find, 4 -> map]
The second line will contain inputs based on the type of operation.
Type 1: 2 strings denoting the key, and value to be inserted respectively.
Type 2: 1 string denoting the key to be erased.
Type 3: 1 string denoting the key to be found.
Type 4: 1 string denoting the key on which operation must be applied.

Output Format

Type 1: print nothing.
Type 2: print nothing.
Type 3: 1 if your output is **true**, 0 if your output is **false**.
Type 4: print the value associated with the key.

b. Using the unordered map implemented in the last part, solve the following question:

Problem: You are given a array of size **N.** Your task is to count the number of distinct elements in every subarray of size **K.**

Input Format

The first line contains two integers **N and K** where **N** denotes the size of the array and **K** denotes the size of the subarray.

The second line contains N space-separated integers denoting the value of the individual elements in the array.

Output Format

A single line that contains **N - K + 1** elements denoting the number of distinct elements in the individual subarray.

Constraints

$1 <= N <= 10^4$

$1 <= K <= N$

Sample Input

```
7 4
1 2 1 3 4 2 3
```

Sample Output

```
3 4 4 3
```

Explanation

First window is `{1, 2, 1, 3}`, count of distinct numbers is 3.

The second window is `{2, 1, 3, 4}`, count of distinct numbers is 4.

The third window is `{1, 3, 4, 2}`, count of distinct numbers is 4.

The fourth window is {3, 4, 2, 3}, and the count of distinct numbers is 3.

3. Shortest Path

You are given an undirected and weighted graph, denoting a city. Each node represents a city block. The edges represent a road connecting the blocks. Their weight represents the length of the road. Some blocks contain a police station. For each city block, find the distance from the nearest police station.

Input Format

The first line contains **N, M,** and **K,** denoting the number of city blocks, the number of roads, and the number of city blocks that contain a police station.

The following **M** lines contain **u, v,** and **w** each, denoting that there is a road between **u** and **v** of length **w.**

The next line contains **K** space-separated integers (K_i), denoting a list of city blocks that contain a police station.

Constraints

City blocks are numbered from 1 to **N.**

1 <= **N, M, K** <= 10^4

1 <= **u, v,** K_i <= N

1 <= **w** <= 100

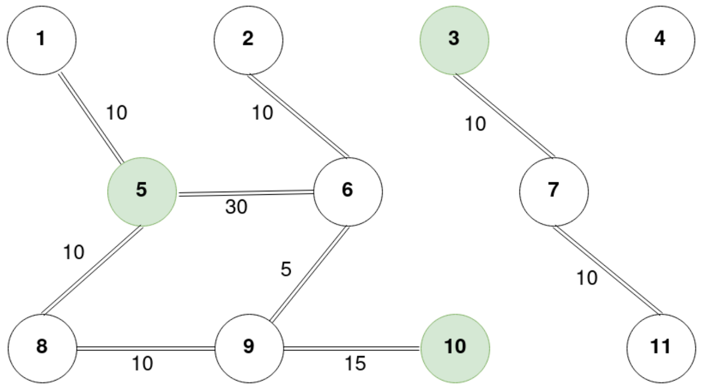
Output Format

Print N space-separated integers denoting the minimum distance of a police station from the i^{th} city block (for 1 <= i <= N). If the city block cannot reach any police station, print **-1** for the block.

Sample Input

```
11 9 3
1 5 10
2 6 10
3 7 10
5 6 30
5 8 10
6 9 5
7 11 10
8 9 10
9 10 15
3 5 10
```

The graph corresponding to the above input is shown below.



Sample Output

```
10 30 0 -1 0 20 10 10 15 0 20
```

4. Elections

Elections are around the corner. There are **n** parties competing with each other in the election. Since the competition is high, they are trying to make alliances with each other. You are given m pairs where each pair represents 2 parties that are forming an alliance.

The formation of alliances is transitive i.e. If Party A forms an alliance with Party B and Party B forms an alliance with Party C, then by the transitive nature of alliances, Party A is having an alliance with Party C i.e. Party A, B & C are in one alliance group.

The size of the alliance group is the number of parties in that group.

For each alliance formed, your task is to print the number of alliance groups and the size of the largest alliance group.

An alliance group can contain a single party.

Input Format

The first input line has two integers **n** and **m**: the number of parties and the number of alliances. The parties are numbered **1,2,…,n**.

Then, there are m lines describing the alliance. Each line has two integers **a** and **b** where **a & b** represents the 2 parties forming an alliance.

Output Format

In each of the m lines separated by a new line, you have to print 2 integers **a & b** where **a** represents the number of alliance groups and **b** represents the size of the largest alliance group.

Constraints

$1 <= n <= 10^5$

$1 <= m <= 2 * 10^5$

1 <= a,b <= n

Sample Input

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

Sample Output

```
4 2
3 3
2 3
```

Explanation

Initially, there are 5 alliance groups.

Firstly, **1 - 2** make an alliance so now there will be 4 alliance groups namely:- (1,2), (3), (4), (5) and the size of the largest alliance group will be 2 because (1,2) alliance groups have 2 parties contained in it.

Secondly, **1 - 3** make an alliance so now there will be 3 alliance groups namely:- (1,2,3), (4), (5) and the size of the largest alliance group will be 3 because (1,2,3) alliance groups have 3 parties contained in it.

Thirdly, **4 - 5** make an alliance so now there will be 2 alliance groups namely:- (1,2,3), (4,5) and the size of the largest alliance group will be 3 because the (1,2,3) alliance group has 3 parties contained in it.