Data Structures & Algorithms for Problem Solving
Assignment-4

Deadline: 15th November 2021, 11:55 pm

Important Points:

1. Only C/C++ is allowed.
2. C++ STL is **not allowed** for any of the questions. So #include <bits/stdc++.h> is not allowed.

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

PROBLEM 1: Implementation of suffix array

AIM: Implement a Suffix Array that is capable of performing following operations on Strings in a most efficient way.

1. Given a string S print its minimum lexicographic rotation. **O(nlogn)**
2. Given an integer K, print the length of the longest substring that appears in the text at least K times. If no such substring exists, print -1. **O(nlogn)**
3. Given a string S determine its longest substring that is also a palindrome. In the case of multiple solutions, print the lexicographically smallest palindrome. **O(nlogn)**

EXAMPLE:

Input String: S = "dcabca"

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

   Among all lexicographically minimum is "abcadc" .

2. If K=2 than since "ca" is a substring that appears twice and its length is 2, so the answer is 2
3. Since only length 1 substring are palindromic, and among them "a" is lexicographically smallest, hence the answer is "a".

INPUT FORMAT: You will be given a large string S (length <= 10 ^5). Print the corresponding output for each case Q1a, Q1b and Q1c.

Constraints:

- 1 <= String length <= 10^5
- String consists of either Lower/Upper Case Alphabet and Numeric digits.

Submission Format: For each subpart implement a different code. Submit it as rollnumber_Q1a.cpp, rollnumber_Q1b.cpp, rollnumber_Q1c.cpp.

We will run each file separately.

PROBLEM 2: External Sorting

AIM: External Sorting is a class of algorithms used to deal with massive amounts of data that do not fit in memory.The question aims at implementing one such type: K-Way merge sort algorithm to sort a very large array. This algorithm is a perfect example of the use of divide and conquer where with limited resources large problems are tackled by breaking the problem space into small computable subspaces and then operations are done on them.

Input Constraints:

1. A file containing a large unsorted list of integers (Will not fit in your usual Laptop RAM).
2. Do not use any in-built data structures.

Output: A file containing non-Descending sorted list of the given integers

Evaluation parameters :

1. Time and Space Complexity of the algorithm
2. Efficient use of Data-Structures

Input Format: Your code should take two arguments.

- First is the name of the input file.
- Second is the name of the output file.
- Example Format: If your input file is at ./data/input.txt And if you need your output file at ./data/ named output.txt.**For c++, code should be of format rollnumber_Q3.cpp compiled file should accept two arguments ./a.out "./data/input.txt" "./data/output.txt"**

Generation of unsorted file:

To generate the unsorted file, a python script is uploaded along with this pdf. It contains all the instructions required to run it.

Submission format: Submit it as rollnumber_Q2.cpp

PROBLEM 3: Trie Implementation

AIM: Given an array A of N numbers, you will be given q queries. Each query will contain a single integer x. You have to then find the maximum xor of x with any number in A.

Constraints:

- 1 <= N, q <= 10 ^ 5
- 1 <= A[i] <= 10 ^ 12

INPUT:

First Line contains N and q .

Second line contains N space separated integers.

Next q lines contain q queries of single integer

Example:

3 2

1 2 3

4

5

A = {1, 2, 3}

Case 1: x = 4 Maximum xor of x is with 3, therefore answer is 4 xor 3 = 7

Case 2: x = 5 Maximum xor of x is with 2, therefore answer is 5 xor 2 = 7

Submission Format: Submit it as rollnumber_Q3.cpp