Data Structures & Algorithms for Problem Solving
Assignment-3

Deadline: 25th October 2021, 11:55 pm

Important Points:

1. Only C/C++ is allowed.
2. Submission Format: <RollNo>_<Question_No>.cpp

    Ex: For question 1, 2020201001_Q1.cpp.

    Copy all the codes in a folder with folder name as your roll number and submit the zip file in moodle.

    Ex: 2020201001_A3.zip

    Note: All those submissions which are not in the specified format or submitted after the deadline will be awarded 0 in assignment.
3. STL is not allowed in Q3.
4. Don't use #include <bits/stdc++.h> in any of the questions.

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

1. Spell Checker

Design an efficient spell checker using trie data structure which supports the functionalities mentioned below.

Required Features:

- Spell Check: Check if the input string is present in the dictionary.
- Autocomplete: Find all the words in the dictionary which begin with the given input.
- Autocorrect: Find all the words in the dictionary which are at an edit distance(Levenshtein distance) of at most 3 from the given input.

Input Format:

- First line will contain the number of words in the dictionary.
- Next n lines will contain strings of length atleast 5 and atmost 100.
- Next line will contain two space separated values, first one being an integer ai and second will be a string ti.
  ai = 1 means Spell Check operation needs to be done on ti
  ai = 2 means Autcomplete operation needs to be done on ti
  ai = 3 means Autocorrect operation needs to be done on ti

Output Format:

- Output for spell check operation should be ‘1’ if string is present in the dictionary, otherwise 0.
- Output for autocomplete and autocorrect must first contain the number of words in the answer and then the following lines should be the set of words in lexicographical order.

Note:

- Only trie should be used for storing the words in the dictionary.
- STL is allowed.

Submission Format:

RollNo_Q1.cpp

References:

https://en.wikipedia.org/wiki/Levenshtein_distance

2. City Map

Given a map of cities with roads connecting them, you have to write programs giving the solution for the following problems.

The cities are represented by integers ranging from 0 to n-1, where **n** is the total number of cities. There are **m** number of edges denoting the undirected path between two cities and the time it takes while travelling on that road.

1. Given a city A, print all the shortest path from every other city to this city A. If there are more than one path for two cities, then print lexicographically the smaller path.
   The output should be a list of paths in every new line and a path should be represented as the space separated cities number.
   Eg: if 0 2 3 and 0 1 3 both are the shortest path then 0 1 3 should be printed.
2. Print the k shortest paths. Once again if two paths have same length then print the lexicographically smallest one.
3. Suppose the structure of a city is represented by a grid. You are in the bottom right corner of the grid and have to go to the top left corner. There are some **Q** people travelling with their bikes in the city. You have to print whether you can reach destination without colliding with any of those bikes and if it is possible then print the length of the shortest path you took to reach the destination.
   N.B The 3rd question is intra city not inter city and the bikes can move in all 4 directions.

   Only simple stls like string, vector, sort are allowed. Map, Set, priority queue are not allowed.

3. Puzzle Solver

Problem Statement:

You will be given a 2D grid of alphabets and a list of words. Your task is to find the list of words listed in the list and also present in the grid.

Constraints:

- You can only move in the blocks sharing the adjacent sides.
- You cannot use a single block twice in one word.
- The selection stretch must be contiguous.

Input Format:

- The first line will contain two integers r and c. Denoting the number of rows and columns respectively.
- Next r lines will contain c integers which will represent the row wise data of the grid.
- Next line will be an integer X. denoting the number of words in the list.
- Next X line will contain a single string each in which the i’th line will denote the i’th member of the list.

Output Format:

- Output on a single line, space separated words present in the grid in lexicographically ascending order.

Submission Format:

RollNo_Q3.cpp