M22CS1.304 Data Structures and Algorithms for Problem Solving

Assignment 3

Deadline: 11:59 pm. October 03, 2022

Important Points:

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

```
2021202003_A3
    |_____ 2021202003_A3_Q1.cpp
    |_____ 2021202003_A3_Q2.cpp
    |_____ 2021202003_A3_Q3.cpp
```

   Replace your roll number in place of `2021202003`

3. Submission Format: Follow the above mentioned directory structure and zip the **`RollNo_A3`** folder and submit **`RollNo_A3.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 the moodle.

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 two space separated integers **n, q** which represents the number of words in the dictionary and the number of queries to be processed respectively.
- Next **n** lines will contain a single string **s** which represents a word in the dictionary.
- Next **q** lines will contain two space separated values, First one will be an integer $\mathbf{a_i}$ and second will be a string $\mathbf{t_i}$.
    - $\mathbf{a_i = 1}$ means Spell Check operation needs to be done on $\mathbf{t_i}$.
    - $\mathbf{a_i = 2}$ means Autcomplete operation needs to be done on $\mathbf{t_i}$.
    - $\mathbf{a_i = 3}$ means Autocorrect operation needs to be done on $\mathbf{t_i}$.

Output Format:

For each query print the result in a new line.

- **Spell check**: Print **‘1’** if string is present in the dictionary, otherwise **‘0’**.
- **Autocomplete** & **Autocorrect:** Print the number of words in the first line. The following lines will be the set of words in lexicographical order.

Constraints:

1 <= **n** <= 1000
1 <= **q** <= 1000
1 <= **len(s)** <= 100
1 <= **len(t_i)** <= 110

Sample Input:

```
10 4
consider
filters
filers
entitled
tilers
litter
dames
filling
grasses
fitter
1 litter
1 dame
2 con
3 filter
```

Sample Output:

```
1
0
1
consider
5
filers
filters
fitter
litter
tilers
```

Note:

- Only trie should be used for storing the words in the dictionary.
- You are allowed to use `vector` for this problem

References: Levenshtein Distance

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.

Task:

Given a file containing a large unsorted list of integers (Will not fit in your usual Laptop RAM) as input, generate an output file with non-descending sorted list of given integers.

Input format:

- Input.txt contains space separated integers.
- Program should take two command line arguments.
- First command line argument is the input file path.
- Second command line argument is the output file path.
- For example: You have a directory named **`data`** in your current working directory inside which you have kept the **`input.txt`** file and you wish to create **`output.txt`** inside the **`data`** directory itself. Then, run the program using the following command.
  `./a.out ./data/input.txt ./data/output.txt`
- Input files can be arbitrarily large.

Output Format:

- Print the following details in the terminal window:
  - Number of integers in a temporary file.
  - Number of temporary files created.
  - Total time taken by the program upto 2 decimal places.
- Output file should contain integers in non-descending sorted order in a new line.

Evaluation Parameters:

- Correctness.
- Time and space complexity of the algorithm.
- Efficient use of data structures.

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.

Note: You are allowed to use `vector` and inbuilt `sort` for this problem

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 present in the list and also present in the grid.

Note:

- 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** characters which will represent the row wise data of the grid (characters are space separated).
- Next line will be an integer **X**. denoting the number of words to be searched.
- Next **X** line will contain a single string **s** which denotes the word to be searched in the puzzle.

Output Format

- Print the number of words in the first line.
- Print words present in the grid in lexicographically ascending order each in a new line.

Constraints:

1 <= **r, c** <= 10

1 <= **X** <= $3 * 10^4$

1 <= **len(s)** <= 10

- All the strings are **unique**.
- Grid contains only lowercase english characters.
- Strings contain lowercase english characters.

Sample Input

```
4 4
o a a n
e t a e
i h k r
i f l v
6
oath
pea
eat
rain
hklf
hf
```

Sample Output

```
4
eat
hf
hklf
oath
```