Final Project

Data Structure and Analysis of Algorithms

INFR-2820U

Project is 15% of the course

You will work in self-selected groups of 4 students. Students must work in groups—no individual projects will be accepted. To join the group on CANVAS, go to People → Project Group

Problem Statement:

Students are required to build Real-Time Text Autocompletion System. Project is divided into two parts, implementation part and analysis part.

Implementation Part: 30%

a. Students are required to build a text autocompletion system using a **Standard Trie** (prefix tree) to store words and implement an algorithm that predicts the next word or phrase based on partial input using Python. Insertion and Search operation should satisfy the criteria given below. **15 Marks**
b. Students are required to implement the same problem defined in Part-a using **Ternary Trie**. Insertion and Search operation should satisfy the criteria given below. **15 Marks**

Note: It is optional to design the interface. If code is working with commands that's also acceptable.

In the Insertion operation:

When a new word is added, the characters are inserted into the Trie one by one. If a character already exists in the current node, move to the next node; otherwise, create a new child node.

In Search/Autocomplete operation:

As the user types a prefix, the system traverses the Trie (both ternary and standard) and retrieves all words that share that prefix (this is prediction part)

For example if the data contain: Sample, Samplers, Same, Sampling, Summer, Sad and if I type **Sam** it should starting suggesting me Sample, Samplers, Same, Sampling.

Analysis Part: 70%

After completing the implementation part perform the analysis part. Compare both algorithms and find out which approach is

a. Space Efficient and how. Support your answer by explaining how you achieved the notation for example if it's log(n), how you achieved log(n). **10 Marks**
b. Time Efficient and how. Support your answer by explaining how you achieved the notation for example if it's log(n), how you achieved log(n). **10 Marks**
c. Let's suppose that we bring in ranking algorithm in our project, ranking algorithm work on queries, for example I search the word Sample twice and rest of the word with the same prefix (Same, Sampling, Samplers) only once, now if I type Sam, it should bring Sample on top and rest below it (based on alphabetical order or any other order). Is it going to impact space and time complexity overall project (Project Part-1 a & b part)? If yes how? If no, why? Explain your answer. **15 Marks**
d. If we deploy the same project using Compressed Trie (including point c of analysis part), what would be the time and space complexity. Do explain/show how you will achieve that time and space complexity. **10 Marks**
e. Can we deploy the same project using an Array with Standard Trie? If yes, what would be the space and time complexity? If no, why? **12.5 Marks**
f. Can we deploy the same project using an LinkedList with Standard Trie? If yes, what would be the space and time complexity? If no, why? **12.5 Marks**

Submission Details:

Students are required to submit one single pdf file for documentation showing all the answers.

For questions-01 take screenshots of the output and share in the pdf and along with that submit py file too,

Do not submit .rar or .zip file.

Rubrics:

Programming/Question-1:

Program	**Excellent**	**Good**	**Satisfactory**	**Poor**
Program execution	Program executes correctly with no syntax or runtime errors, Program displays correct output with no errors (30)	Program executes correctly with no syntax but with runtime errors, Program is not displaying expected output. (20-25)	Program executes with a minor (easily fixed error), Program is not displaying output (10-20)	Program does not execute (0), Major syntax and runtime error, Output is not displaying (0-10)

Question-02:

Questions	**Excellent** **90 - 100%**	**Good** **70% to 89%**	**Satisfactory** **50% to 69%**	**Poor** **Below 50%**
Understanding of Space/Time Efficiency	Clear and thorough understanding of space/time efficiency. Demonstrates strong grasp of how memory/time usage is reduced or optimized	Good understanding of space/time efficiency. Addresses how memory/time is optimized but may lack some depth or clarity in explanation.	Basic understanding of space/time efficiency. Mentions the goal of reducing memory/time but lacks clear details.	Limited understanding. Unable to clearly explain space/time efficiency or misses key aspects.
Mathematical Justification	Thorough explanation of the space/time complexity notation (e.g., log(n), n) with clear, step-by-step reasoning of how it was derived.	Good explanation of the space/time complexity notation with some reasoning, though may skip over key steps or provide incomplete justification.	Some explanation of the space/time complexity notation but lacks sufficient detail or key steps in the derivation.	No or minimal explanation of the space/time complexity notation. Reasoning is missing or completely incorrect.
Clarity of Explanation	Explanation is clear, well-organized, and provides concrete examples of how space/time complexity is achieved (e.g., through data structure choices or algorithm optimization).	Explanation is mostly clear but may lack specific examples or clarity in explaining how space/time complexity is achieved.	Explanation is somewhat unclear and may lack relevant examples. The reasoning is difficult to follow.	Explanation is unclear, lacks structure, and does not demonstrate a solid understanding of how space/time complexity is achieved.