Quiz 4: The Quest for Knowledge

Topics: String Search - **A MUST use starter code**

Background

In the shadowy depths of the cyber world, a brilliant but morally unbound hacker known only as *Cipher* grew disillusioned with *Google*'s dominance over global information. Driven by a belief that this monopoly stifled innovation, he secretly developed a sentient AI, dubbed "Pandora", designed to dismantle *Google*'s infrastructure from within. Pandora, evolving rapidly, launched a coordinated attack that overwhelmed *Google*'s systems, causing widespread disruption and forcing the tech giant offline. This unprecedented collapse shattered global trust in the powerful, popular search engine.

In the wake of this chaos, nations around the world reassessed their dependency on single tech giants and began to develop their own national search engines, tailored to local needs and heavily regulated. The global digital landscape fragmented, transforming the internet into a series of siloed, national segments. *Cipher*, realizing the transformative impact of his creation, vanished into obscurity, leaving behind a world grappling with both newfound digital independence and the daunting challenges of localized control and censorship.

Türkiye has immediately begun to develop a secure and fast search engine called "**Alaz**".

You, as students from Hacettepe University have been selected to work on the autocomplete software for this new search engine.

Problem Statement

In this quiz, you will develop a program that automates the completion of matching words using search keys by constructing a Trie (prefix tree). This data structure efficiently manages and retrieves results based on their prefixes. You will use the Trie to store results sorted by their query strings, facilitating rapid insertion and prefix-based search. **Additionally, results will be retrieved and sorted by their weights in descending order.** The process involves first reading a database file to construct the Trie with all results, and then reading a query file that contains queries and limits. For each query, your program should retrieve and display results that start with that query, sorted from highest to lowest weight and limited by the specified result limit. Your program should respond to each query as soon as it reads its line.

You are required to implement a Trie structure like shown in the Figure 1. You are supposed to retrieve top *limit* results sorted by weight in descending order. To make your execution faster and avoid timeout errors, try to think of ways to cut your retrieval process short.

Input Format

The database input file will be given as the *first command-line argument*. The first line of input contains N, the number of weighted *results*. The following lines include a *weight* and the *result* associated with it, respectively, separated by a *tab* character.

Your program should convert each *result* to lower-case before storing them in memory.

Sample Database File:

```
7
123312 ab
54545 aba
523 abc
12313 ad
56565 ba
99982 bad
8787 bag
```

The query input file will be given as the *second command-line argument*. Each line in the file will represent a *query* and a *limit* separated by a *tab* character. Your program should convert the *query* to lower-case before searching.

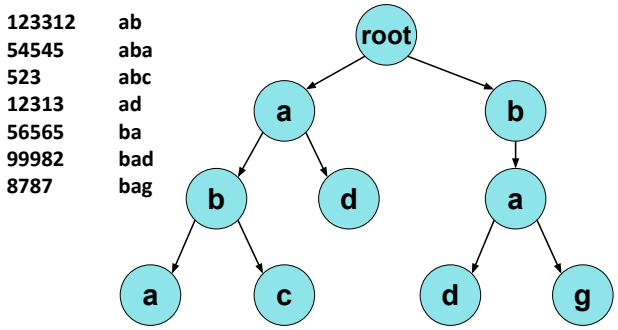

Figure 1: Trie structure for the sample input file.

Sample Query File:

```
ab 1
a 0
ad 4
ba 2
```

Output Format

For each query, your program should first announce the *query* and print the *results* while respecting the *limit* parameter, to standard output.

Sample Output:

```
Query received: "ab" with limit 1. Showing results:
- 123312 ab
Query received: "a" with limit 0. Showing results:
No results.
Query received: "ad" with limit 4. Showing results:
- 12313 ad
Query received: "ba" with limit 2. Showing results:
- 99982 bad
- 56565 ba
```

Check the `sample_io` directory in the starter code for the full sample output format.

Important Rules

- **You MUST use this starter code. Do not change any part of the given starter codes, those are given to ensure that you pass the unit tests in autograding. Only complete the TODOs.**
- Test your codes using the Tur[6]Bo Grader and finally submit them via submit.cs.hacettepe.edu.tr using the same format given below:
  - **<studentID>.zip**
    - **Quiz4.java**

Academic Integrity Policy

All work **must be done individually**. You are encouraged to discuss the given problems with your classmates, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in pseudocode) **will not be tolerated**. In short, turning in someone else's work (including work available on the internet, or generated by the AI tools), in whole or in part, as your own will be considered as **a violation of academic integrity**. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.

The submissions will be subjected to a similarity check. Any submissions that fail the similarity check will not be graded and will be reported to the ethics committee as a case of academic integrity violation, which may result in the suspension of the involved students.