DS final project – A search engine simulation

111060026 劉孟瑋

**1 Implementation**

* 1. **Main code**

The main code of the program uses the data structure of trie. Trie is a structure that saves data into a tree. It helps lessen the time of searching the whole paragraph by using tree traversal instead of for.

* 1. **Reading Equation**

The reading of the equation uses the post order traversal. Then the program reads through the post order equation. If the program hits a word that starts with “‘a’ to ‘z’” or “’A’ to ‘Z’”, it checks the next symbol and does operations accordingly. If the word is not “‘a’ to ‘z’” or “’A’ to ‘Z’”, it means it must be “and” or “or” or “exclude”.

* 1. **Queries**
     1. **Exact Search**

Running through the trie until the word that we wanted comes to an end. If the trie is not at an end or cannot find the same word, the word does not exist. If the word exists, the program pushes the words that are used in the paragraph into a vector.

* + 1. **Prefix Search**

Running through the trie that contains the prefix word. Like exact search, if the word cannot be found, that means there is no word that starts with it. Then, we search through all the possibilities in the trie with the prefix.

* + 1. **Suffix Search**

The thought is the same as prefix. However, we make a reversed trie and search it by using prefix search.

* + 1. **Wildcard Search**

First find words using prefix search, then we run the word with the pattern. If the pattern is ‘\*’, we can ignore the letters in the word. If the pattern matches, the program returns true and the paragraph that uses the word is pushed into vector. If the first word of the pattern is ‘\*’, go the other way around.

* + 1. **And, Or, Exclude**

Since every answer is returned by a vector and the program uses the post order traversal when we see a “and” or “or” or “exclude”. We can simply do operations to the first and second vector we got. After doing the operations, we push the answer back to the first vector.

1. **Optimization**
   1. **Saving the data from paragraphs**

There are a lot of words that should be saved in the trie tree. At first, I thought that making a trie tree for every paragraph. However, it is obvious that it will take too much memory. After checking my code again, I found out that I can save which paragraph uses the word by saving the words that are used in the paragraph through a vector at the end of the word. The following picture is the saving of alternated trie.

|  |  |
| --- | --- |
|  |  |
| Picture 1: The words & paragraphs | Picture 2: Making into altered trie |

1. **Challenges and Conclusion**
2. Sometimes it is hard to think up an algorithm by oneself. Therefore, finding reference on the Internet is needed. Asking others about their thoughts is important as well.
3. Debugging is hard since there are around 600~700 lines in the code.
4. GitHub is a good tool for version control.
5. **Reference**

Reference through chatting: My friends

Trie: <https://www.geeksforgeeks.org/trie-insert-and-search/>

Read folder: <https://cplusplus.com/forum/beginner/10292/>

Find algo: <https://cplusplus.com/reference/algorithm/find/>

Suffix: <https://www.geeksforgeeks.org/find-strings-that-end-with-a-given-suffix/>

Prefix: <https://www.geeksforgeeks.org/auto-complete-feature-using-trie/>

Wildcard: <https://www.geeksforgeeks.org/wildcard-pattern-matching/>