ASSIGNMENT 4

Subject : Tries
TAs: Merve Ozdes, Ahmet Alkılınç, Hayriye Çelikbilek
Programming Lanuage: C++
Due Date: 31.12.2021 23:59

1 Introduction

In this assignment, you are expected to design a Dothraki to English dictionary. The Dothraki language is a constructed fictional language in George R. R. Martin's fantasy novel series A Song of Ice and Fire and its television adaptation Game of Thrones. The primary goal of this experiment is to get you to practice on the data structure called trie (or prefix tree). This special kind of a tree called trie is used to store maps of keys to values where the keys are usually strings. In your case Dothraki words are the keys and English words are the values.

2 Background

Strings can essentially be viewed as the most important and common topics for a variety of programming problems. String processing has a variety of real-world applications too, such as:

- Search Engines
- Genome Analysis
- Data Analytics

All the content presented to us in textual form can be visualized as nothing but just strings.

A Character tree is a special form of tree data structure that is based on the prefix of a string. The prefix of a string is nothing but any n letters, n <=|S| that can be considered beginning strictly from the starting of a string.

A character tree is a special data structure used to store strings that can be visualized like a graph. It consists of nodes and edges. Each node consists of at max 26 children and edges connect each parent node to its children. These 26 pointers are pointers for each of the 26 letters of the English alphabet. A separate edge is maintained for every node.

Strings are stored in a top to bottom manner on the basis of their prefix in a character tree. All prefixes of length 1 are stored at until level 1, all prefixes of length 2 are sorted at until level 2 and so on. Character trees are generally used on groups of strings, rather than a single string. When given multiple strings, we can solve a variety of problems based on them. For example, consider an English dictionary and a single string s, find the prefix of maximum length from the dictionary strings matching the string s. Solving this problem using a naive approach would require us to match the prefix of the given string with the prefix of every other word in the dictionary and note the maximum. This is an expensive process considering the amount of time it would take. Character trees can solve this problem in a much more efficient way. Before processing each Query of the type where we need to search the length of the longest prefix, we first need to add all the existing words into the dictionary. A Character tree consists of a special node called the root node. This node doesn't have any incoming edges. It only contains max 26 outgoing edges for each letter in the alphabet and is the root of the character tree. So, the insertion of any string into a character tree starts from the root node. All prefixes of length one are direct children of the root node. In addition, all prefixes of length 2 become children of the nodes existing at level one. The representation of Trie data structure is given in Figure 1.

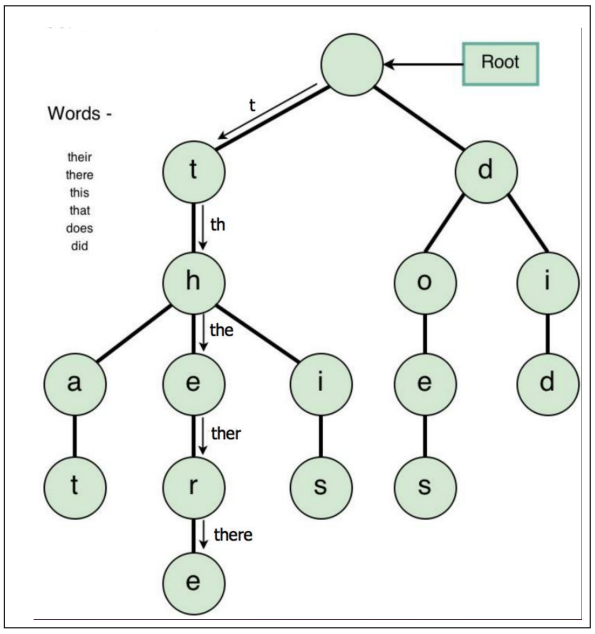

Figure 1: Representation of Trie data structure

3 Experiment

In this experiment, you are expected to write an application that constructs an Dothraki-English dictionary with Trie data structure. The application will take the input.txt file from command line and read its contents. There are some commands for this experiment such as adding, deleting and listing. **Your task is implement these features with min time and space costs.**

Structure of Commands:

- **insert(k,v):** should insert a key-value pair (k, v) into the trie; if the key k already existed the corresponding value should be replaced with v. **Key** corresponds to Dothraki word and **value** corresponds to English word.
- **search(k):** search with the given key and return the value
- **delete(k):** delete the key and its value
- **list:** should return a printable representation of the trie with keys sorted lexicographically

3.1 Insert command: insert(k,v)

- The application will read the given Dothraki word (k) character by character and add them to the tree. With the last character of the Dothraki word store the English equivalent (v) of that word.
- If the first character of the Dothraki word (k) is not referenced by the root node, the word will be added to the tree starting from the root node.
- If the first n character of the Dothraki word (k) exists on the tree, a branch occurs on the n^{th} node for the last characters
- The node which is the last character of the Dothraki word has to hold the English equivalent for the given Dothraki word.

- If there is a Dothraki word same as the given word (k) and their values (v) are also same, the application will give an output that “xxx already exist”. If the key k already existed and values are different, then the corresponding value should be replaced with v and "xxx was updated".

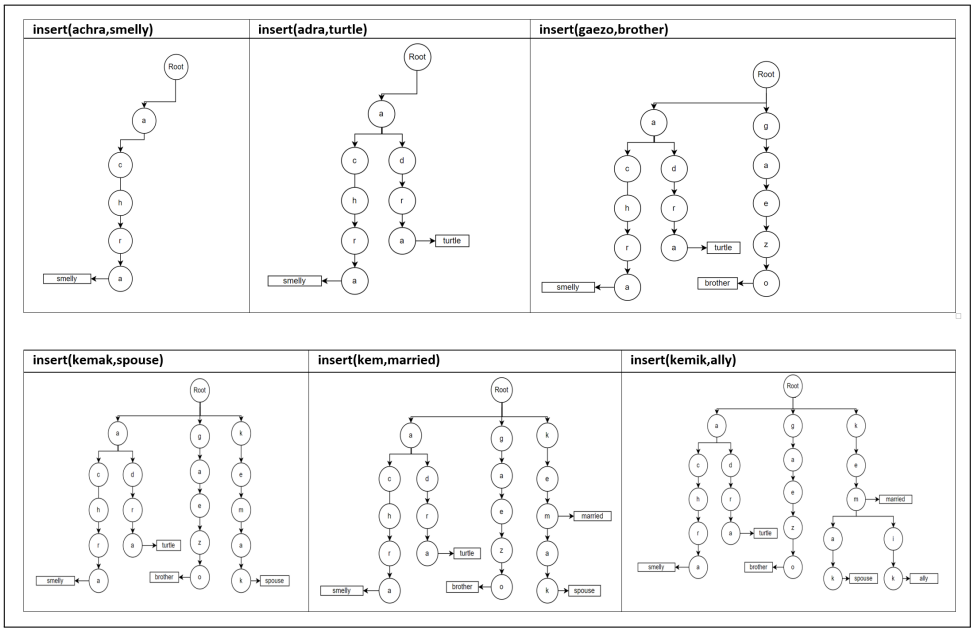

Figure 2: Example of insert command

3.2 Search command: search(k)

- The application will read the given Dothraki word (k) and according to the questioned word it will return an information.
- If the first character of the k is not referenced by the root node, the application will give an output that “no record”.
- If the first n character of the k exists on the tree, but the remainder is not, the application will give an output that “incorrect Dothraki word”.
- If all characters of the k exist on the tree, but the last character has no English equivalent, the application will give an output that “not enough Dothraki word”.
- If all characters of the k exist on the tree and the last character has its English equivalent, the application will give an output that “The English equivalent is xxx”.

search(mekis)	search(adra)	search(gaez)	search(kemok)	search(kemak)	search(kem)
“no record”	“The English equivalent is turtle”	“not enough Dothraki word”	“incorrect Dothraki word”	“The English equivalent is spouse”	“The English equivalent is married”

Figure 3: Example of search command

3.3 Delete command: delete(k)

- With the given k, the application will delete the key and its value from the tree.

- If the first character of the k is not referenced by the root node, the application will give an output that "no record".
- If the first n character of the k exists on the tree, but the remainder is not, the application will give an output that "incorrect Dothraki word".
- If all characters of the k exist on the tree, but the last character has no English equivalent, the application will give an output that "not enough Dothraki word".
- If all characters of the k exist on the tree, and the last character has the English equivalent, the application will delete all nodes which are not connected to another Dothraki word. Then it will give an output that "xxx deletion is successful".

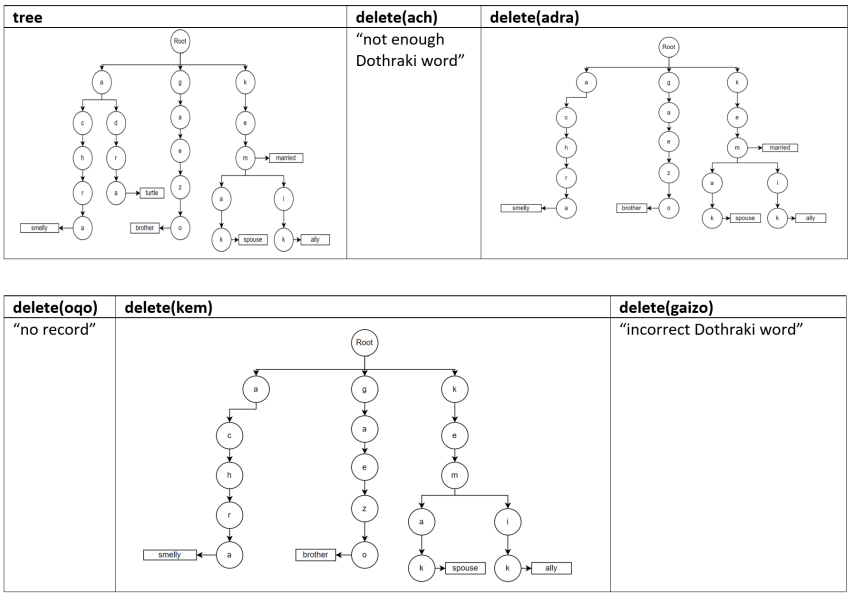

Figure 4: Example of delete command

3.4 List command: list

- The application will list the all Dothraki words with their values by preorder traversal.
- After the first level of the tree, each branch will be displayed with a new tab. If there is no branch, no need a new tab. If a branch have different values, you can display with one tab. Following example is given for this case (Only output of list command is shown).

```
insert(at,one)                -a
insert(atak,first)     --->       -ale(some)
insert(ataki,first)               -at(one)
insert(ale,some)                  -atak(first)
list                              -ataki(first)
```

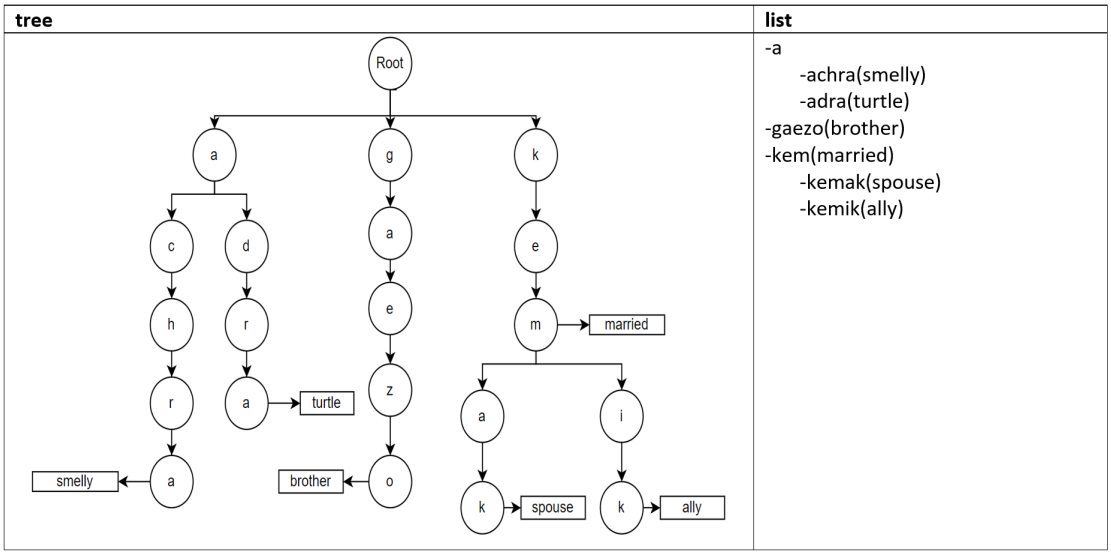

Figure 5: Example of list command

4 Inputs and Outputs

For this assignment you have one input file that contains commands. You are expected to produce output.txt file according to input.txt file. The format of input and output files are shown below. The input.txt file of the example given in Figures above is as follows.

```
insert(achra,smelly)
insert(adra,turtle)
insert(gaezo,brother)
insert(kemak,spouse)
insert(kem,married)
insert(kemik,ally)
insert(gaezo,brother)
search(mekis)
search(adra)
search(gaez)
search(kemok)
search(kemak)
search(kem)
list
delete(ach)
delete(adra)
delete(oqo)
delete(kem)
delete(gaizo)
delete(kem)
list
```

The output of this assignment which are created according to the input.txt file should be as shown below. Please create your output file according to the format given to you.

```
"achra" was added
"adra" was added
```

```
"gaezo" was added
"kemak" was added
"kem" was added
"kemik" was added
"gaezo" already exist
"no record"
"The English equivalent is turtle"
"not enough Dothraki word"
"incorrect Dothraki word"
"The English equivalent is spouse"
"The English equivalent is married"
-a
    -achra(smelly)
    -adra(turtle)
-gaezo(brother)
-kem(married)
    -kemak(spouse)
    -kemik(ally)
"not enough Dothraki word"
"adra" deletion is successful
"no record"
"kem" deletion is successful
"incorrect Dothraki word"
"no record"
-achra(smelly)
-gaezo(brother)
-kem
    -kemak(spouse)
    -kemik(ally)
```

5 Grading and Evaluation

- Your work will be graded over a maximum of 100 points.
- Your total score will be partial according to the grading policy stated below.

Taking input and output files from command line	10p
Insertion	20p
Deletion	20p
Searching	20p
Listing	20p
Code design, clean and readable code, algorithmic perspective, comments	10p

- **Your code will be tested on dev platform with different inputs. And one output file will be expected as output file. If your program cannot compile on dev server, you cannot get any points. Please be sure that your program can be compile on dev server.**
  - Upload your files to your server account (dev.cs.hacettepe.edu.tr)
  - Compile your code (g++ -o Main *.cpp)
  - Run your program (./Main input.txt output.txt)
  - Control your output (output.txt).

Usage example:
>g++ -o Main *.cpp
>./Main input1.txt output.txt

Notes

- Do not miss the deadline.
- Save all your work until the assignment is graded.
- The assignment must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.
- Write READABLE SOURCE CODE block
- You can ask your questions via Piazza (https://piazza.com/hacettepe.edu.tr/fall/bbm203) and you are supposed to be aware of everything discussed in Piazza.
- You will use online submission system to submit your experiments. https://submit.cs.hacettepe.edu.tr/ No other submission method (email or etc.) will be accepted. Do not submit any file via e-mail related with this assignment.
- File hierarchy must be zipped before submitted (Not .rar, only .zip files are supported by the system). You must submit your work with the file hierarchy stated below:

<student_id.zip>/(Required)

→src/(Required)

→Main.cpp(Required)

→*.cpp(optional)

→*.h(optional)

Policy

All work on assignments must be done **individually** unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, 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 the pseudocode) **will not be tolerated**. In short, turning in someone else's work (from internet), 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.