FIT2004 S2/2016: Assessment questions for week 8

(6 Marks)

DEADLINE: Monday, 12-Sep-2016 10:00:00 AM
LATE PENALTY: 25% per day
CLASS: These questions have to be completed before the deadline. Your demonstrator will check your submission during the lab. He will ask you a series of questions to assess your understanding of this exercise, and gauge how you implemented it. It is required that you implement the programming question strictly using **Python programming language**. Your solution is marked on the performance of your program **and also on your understanding of the solution**. "Forgetting" is not an acceptable explanation for lack of understanding. Demonstrators are not obliged to mark programs that do not run or that crash.
After/befor your demonstrators have interviewed you, you are expected to work on the questions that will be provided to you on the day.
SUBMISSION REQUIREMENT: You will need to submit a zipped file containing your Python program (named autocomplete.py) as well as a PDF file briefly describing your solution and its space and time complexities. The PDF file must give an outline of your solution (e.g., a high level idea of how did you solve it) and the **worst-case** space and time complexity of your solution. The PDF file must also include your solution to the problems related to B-tree. The zipped file is to be submitted on Moodle before the deadline.
Important: You should carefully consider the border cases and make sure that you return correct results for all cases. Although you are not required to include the proof of correctness in your submissions, you are encouraged to formally prove the correctness of your programs. This will help you in identifying and fixing the bugs in your program if present.

1 Task 1: Auto-complete

Alice used to love the auto-complete feature in her phone until the day she sent her thesis advisor an email saying "Thank you for being on my adversary panel". She soon received a reply "I hope you meant the advisory panel ;)". Needless to say, she was embarrassed and wrote a long email (this time disabling the auto-complete feature) explaining that the mistake was caused due to the auto-complete feature. She decided to do something about it and now wants to modify the auto-complete feature in her phone.

The standard auto-complete function in her phone displays up to three words that have the user's entered word as a prefix. E.g., if you are typing "adv", the auto-complete feature shows "adversary", "advisory" and "adventure". She wants to modify the auto-complete feature such that it only shows one word (instead of three) but also shows the definition of the word so that she can make sure that she is using the right word. She also wants to know how many words are there in the dictionary that have this prefix. E.g., if there are 43 words that have "adv" as a prefix, the auto-complete feature must display 43.

She has downloaded a dictionary from the internet that contains English words and their definitions. She has also downloaded the text of all the emails she has ever sent. She has processed the text and has assigned each word in the dictionary a frequency which corresponds to the number of times she ever used the word in her emails. She wants the auto-complete feature to display the word in the dictionary that has the highest frequency among the words that have her entered word as a prefix. For example, assume that the frequencies of "adversary", "advisory", and "adventure" are 100, 200 and 150, respectively. If she types "adv", the modified auto-complete system should display "advisory" with its definition and the number 43 that corresponds to the number of words that have "adv" as a prefix . She implemented a sorting based algorithm to do the prefix matching but it is too slow.

Well, you guessed it right. She has come to you for help. You need to help her in efficient implementation of this feature. You are given a text file "Dictionary.txt" that contains English words, their frequencies and their definitions. You must write a Python program named `autocomplete.py` that reads from the input file and creates a Trie. Once the Trie has been constructed, the program will ask the user to enter a prefix. The program must then display the prefix matched word having the highest frequency along with its definition and the number of words in the dictionary that have this prefix.

1.1 Input

The input is the file "Dictionary.txt" that contains $80,987$ words, their frequencies and their definitions (see Moodle). However, below we illustrate the input and output assuming that the dictionary contains the details of only the following four words.

```
word: align
frequency: 358
definition: To adjust or form to a line; to range or form in line; to bring
into line; to aline.

word: adversary
frequency: 157
definition: One who is turned against another or others with a design to
oppose

word: advisory
frequency: 720
definition: Having power to advise; containing advice; as, an advisory
council; their opinion is merely advisory.

word: adventure
frequency: 696
definition: To try the chance; to take the risk.
```

Although the provided file "Dictionary.txt" is sorted in alphabetical order of words to make it easier for you to manually verify the correctness of your implementation, you cannot assume that the file on which your algorithm will be tested will also be sorted. Your program will be tested on a different Dictionary.

1.2 Output

Once you have created a Trie, your program must ask the user to enter a prefix. It must then display the word with the highest frequency, its definition and the number of words that have the input text as the prefix. If there are more than one words with the highest frequency, you must display the alphabetically smallest word. The program must keep asking the user to enter another prefix and terminate only when the user enters *** instead of the prefix. Below are some examples (you can assume that all the words in the dictionary and the input are in lower case).

```
Enter a prefix: adv
Auto-complete suggestion: advisory
Definition: Having power to advise; containing advice; as, an advisory
council; their opinion is merely advisory.
There are 3 words in the dictionary that have "adv" as a prefix.

Enter a prefix: adve
Auto-complete suggestion: adventure
Definition: To try the chance; to take the risk.
There are 2 words in the dictionary that have "adve" as a prefix.

Enter a prefix:
Auto-complete suggestion: advisory
Definition: Having power to advise; containing advice; as, an advisory
council; their opinion is merely advisory.
There are 4 words in the dictionary that have "" as a prefix.

Enter a prefix: bat
There is no word in the dictionary that has "bat" as a prefix.

Enter a prefix: ***
```

Note that the third prefix entered by the user in the above example is an empty string in which case the most frequent word in the whole dictionary is returned.

1.3 Complexity/implementation requirement

The construction cost of the Trie must be $O(T)$ where T is the total number of characters in the dictionary. Let M be the length of the prefix entered by the user and N be the total number of characters in the the word with the highest frequency and its definition. Your program must return the response in $O(M + N)$. Note that this is optimal because the size of the output string is $O(M + N)$ and no algorithm can achieve a better complexity. Note that N may be equal to M (i.e., the word is the same as the prefix and the definition is empty).
Important: You must implement your own Trie and are not allowed to use publicly available implementations.

2 Task 2: B-Tree Insertion and Deletion

Assume that the minimum degree t is 3 for the B-trees shown in the figures.

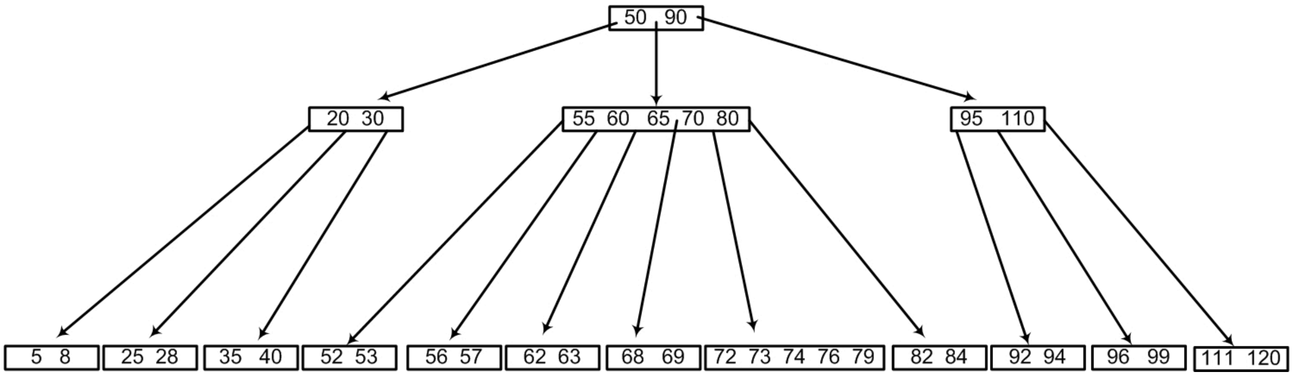

Figure 1: Insert 75 in this B-tree

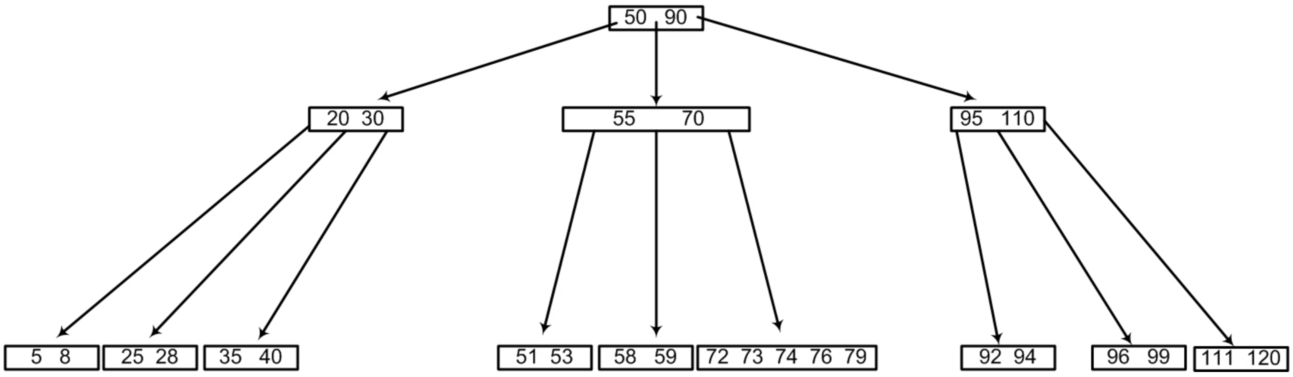

Figure 2: Delete 59 from this B-tree

Task 2a. Show how the B-tree in Fig. 1 will be updated when 75 is inserted in it.
Task 2b. Show how the B-tree in Fig. 2 will be updated when 59 is deleted from it.

For each task, you need to show all the steps of B-tree update. Be careful and do not insert/delete from the wrong B-tree (look at the captions of the figures).