CS251 - Homework 5: Strings II

Out: April 15, 2016 @ 9:00 pm
Due: April 22, 2016 @ 9:00 pm

Name: Joel Van Auken

Read this before start: Not all questions are multiple choice, some of them may require an explanation or simulating an algorithm. In those questions give as much information you consider is required for the solution. Read carefully each question in order to identify what is being asked.

**** For all questions, any answer without an explanation (even though it is correct) will be graded with 0 points. ****

1 (10 pts). Find the shortest string that is **not** in the language represented by the regular expression $\mathbf{a^*(ab)^*b^*}$. The alphabet is {a, b}. (Hint : the empty string is not a valid answer, the answer has one or more characters.)

The string "ba" as there is never a 2 character string with an "a" after a "b" in the language.

2 (10 pts). Find a regular expression corresponding to the language of all strings over the alphabet { a, b } that contain exactly two a's.

b*ab*ab* This could only make a string that contains exactly 2 a's regardless of the number or position of b's.

3 (10 pts). Consider a standard trie constructed from a chemistry book of 500 pages, each page having 2000 words (on average !). Word lengths range from 1 letter to 1909 (There are long chemical names, and sometimes the author forgets to put hyphen to separate the groups. If you don't believe me check this : https://en.wikipedia.org/wiki/Longest_word_in_English).
We are interested in searching English words (so, our alphabet includes standard English alphabet, ignoring case, and numbers 0-9, totalling 36).
We want to search the word "methylhydroxide". What will be the most number of comparisons to search the word?

a) 68724
b) 15
c) 36
d) 540

The most number of comparisons needed is the length of the word. This case would be when there is another word that uses "methylhydroxide" as a prefix, or there is another word that has the prefix "methylhydroxid" as the prefix.

4. (10 points) How many tree nodes are there in a standard trie constructed from the following three words : albert, albany, albeit

a) 11
b) 12
c) 7
d) 9

12 nodes total, including the empty root node.

5 (10 pts). We have an alphabet {A, B, C, D, E}. Consider the following encoding for the alphabet:

A = 011
B = 10
C = 11
D = 00
E = 001

No, this is NOT a valid Huffman encoding because the code for D is a prefix for the code of E. (00 and 001)

Is the above encoding a valid huffman encoding? Explain.

_____

_____

6. (10 points) Which **one or ones** of these following characteristics describes Dynamic Programming?

a) We construct **optimal** solutions from subproblems bottom up.
b) The solutions to the subproblems can be computed **independently**.
c) The solution is continually improved by **local improvements**.
d) The problem is divided into subproblems.

The subproblems overlap, so they cannot be solved independently.

7. (10 points) Which **one or ones** of these following characteristics describes Divide-and-conquer?

a) We construct **optimal** solutions from subproblems bottom up.
b) The solutions to the subproblems can be computed **independently**.
c) The solution is continually improved by **local improvements**.
d) The problem is divided into subproblems.

The solutions (local improvements) are combined to form the final solution. They don't overlap, so they can be solved independently.

8. (10 points) A suffix trie is constructed from the word “sososo”. What will be the number of nodes in the trie (in uncompressed form) ? Also draw the tree.

a) 10
b) 14
c) 18
d) 22

Including the empty root node. Tree attached at end.

9. (10 points) Consider the same suffix trie, but in compressed form. What will be the number of nodes in the compressed suffix trie? Also draw the tree.

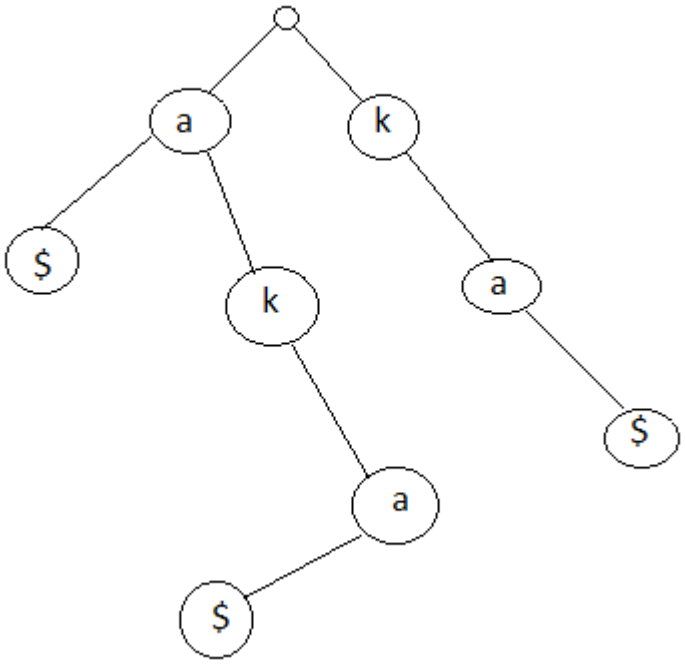
a) 9
b) 10
c) 11
d) 12

Including the empty root node. Tree attached at end.

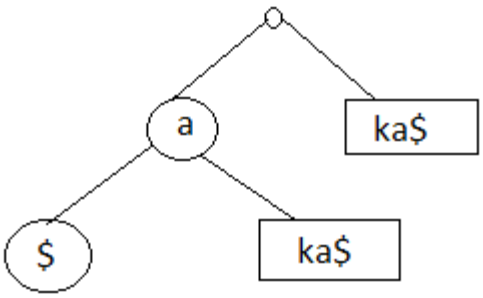
Hint for 8 and 9 : Take the example string “aka”.
The suffix strings constructed will be :

1. aka$
2. ka$
3. a$

Where $ is a special character used to denote the end of a string.
The uncompressed suffix trie will be the following:



In compressed format the suffix trie will be the following:



10. (10 points) With regards to searching for a pattern of size *m* in a text of size *n*, Boyer-Moore (BM) and Knuth-Morris-Pratt (KMP) algorithms obtain a big-Oh search-time related to *n* and *m*. Trie-based methods obtain a big-Oh search-time related to *m* (and

independent of the text size *n*). What is the main difference between trie-based methods and BM/KMP that enables this performance improvement? Your answer should be 1-2 sentences at most.

The main difference between BM/KMP and trie-based methods is the fact that trie-based methods compress a lot of the text down. So if you have multiple words that are the same, trie-based methods are essentially searching all of those spots at once.

Submit Instructions:
The homework must be turned in by the due date and time using the turnin command. Follow the next steps:

1. Please make sure your submission is legible! (No cellphone pictures please! All ITAP labs have scanners for you to use)
2. Login to data.cs.purdue.edu (you can use the labs or a ssh remote connection).
3. Make a directory named with your username and copy your solution (in pdf format) there. (**Important:** Such pdf file should be the only one contained within the folder).
4. Go to the upper level directory and execute the following command:
   turnin -c cs251 -p hw5 your_username
   (Important: previous submissions are overwritten with the new ones. Your last submission will be the official and therefore graded).

Verify what you have turned in by typing **turnin -v -c cs251 -p hw5**
(**Important:** Do not forget the -v flag, otherwise your submission would be replaced with an empty one). If you submit the wrong file you will not receive credit.