Algorithm Analysis Homework 6

April 17, 2023 (Due April 29, Tuesday class)

1. When we discussed construction of Huffman tree, we used a priority queue based on a heap. Note that the frequencies and their sums that occur in Huffman tree construction can be arbitrarily large. Suppose that we are working on an application that uses only a finite number of priorities, say, 1 to 10. (Unix processes has 40 priorities between -20 to 19, and Windows has 32 priorities between 0 and 31. Smaller number represents a higher priority.)

   (a) If items A and B have the same priority and A was pushed into the queue before B was, does a priority queue based on heap *always* pops A before B? If you think it does, explain why. If not, give a counterexample.

   (b) If we have only finite number of priorities, we can make a priority queue with which push and pop operates in constant time *and* among the items of the same priority the one pushed first pops first. Explain how to make such a prioroty queue. (Hint: Consider a hash table.)

2. (a) Download the file "`anna-karenina.txt`" from the homework folder. Write a C program `count` that counts and prints the number of each ASCII character that appears in the input text. For example, the program can take input from `stdin`, and the command "`./count < anna-karenina.txt`" can output as follows:

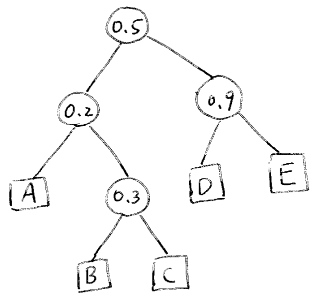
   ```
   .
   .
   .
   count[97]=119804
   count[98]=19902
   count[99]=33921
   count[100]=68057
   count[101]=186523
   count[102]=30983
   count[103]=33029
   count[104]=104878
   count[105]=103974
   count[106]=1416
   .
   .
   .
   ```

   Make your program as fast as possible. On my old laptop computer with 1.6 Ghz Intel CPU, when run against the whole text of Anna Karenina, it takes less than 1 second.

   (b) The size of the file `anna-karenina.txt` is about 2MB. Suppose that you compress the file using the Huffman code obtained by the counting. Estimate the size of the resulting compressed file.

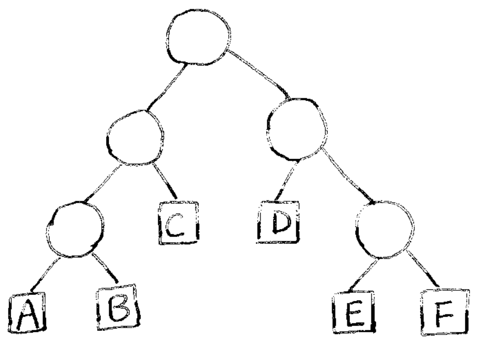
   Hint: You don't have to actually construct the Huffman code. Think about the sum of the numbers in the inner nodes of the Huffman code tree that we constructed in class.

3. (a) For a given real number $r \in (0, 1)$, we traverse the following tree as if we search for the key r in a binary search tree and output the corresponding letter when we reach a terminal node. When r is equal to the number in an inner node, then follow the left edge. For example, if $r = 0.35$, then C is output, and if $r = 0.9$, then D is output. If an input r is a random number uniformly distributed in $(0, 1)$, what is the probability for getting each letter?



$\Pr(A) =$
$\Pr(B) =$
$\Pr(C) =$
$\Pr(D) =$
$\Pr(E) =$

(b) We want to use the following tree and a random number r in $(0, 1)$, as in (a), to produce letters according to the following probability: $\Pr(A) = 0.1$, $\Pr(B) = 0.35$, $\Pr(C) = 0.15$, $\Pr(D) = 0.05$, $\Pr(E) = 0.25$, and $\Pr(F) = 0.1$. What are the values of the nodes of the tree?



(c) When we produce letters by the trees in (a) and (b), each inner node corresponds to a comparison. Construct a tree to produce letters according to the probabilities given in (b), with which the minimum number of comparisons are required *on average*. What is the minimum average number of comparisons?

Hint: Use Huffman's algorithm.