BASIC ALGORITHMS MIDTERM

Open book. Open Notes. NO Websearch. Maximal Score: 80.

1. (20) In a BST *assume* z has parent p, left child w, no right child, and that z is the right child of p. Consider the operation $DELETE[z]$.
   (a) (5) Give $DELETE[z]$ in this case.
   `Solution:`Reset $Right[p] \leftarrow z$ and $\pi[w] \leftarrow p$
   (b) (5) Draw two nice pictures illustrating the changing parts of the tree before and after the $DELETE[z]$ operation.
   `Solution:`

```
          p                          becomes      p

                    z

               w                                       w
```

   Everything above p, below w and elsewhere remains the same.
   (c) (10) Further assume BST has $desc[v]$ which tells, for each vertex v, the number of descendents (counting v itself) of v. Assume z has p, w as above. Extend $DELETE[z]$ so that it *updates* all $desc[v]$ that change value. Your updating should take time $O(H)$.
   `Solution:`p and all of its ancestors have one less descendent, all other values stay the same. Therefore:
   TEMP $\leftarrow p$
   WHILE $TEMP \neq NIL$
   $desc(TEMP) - -$
   $TEMP \leftarrow \pi(TEMP)$
   ENDWHILE

2. (20) Let $A[1 \cdots N]$ be an unsorted array. Consider the following algorithm for creating a maxheap B with this data. Initialize by setting $length[B] = n$, $heapsize[B] = 1$ and $B[1] = 1$. Then
   FOR I = 2 TO N
   INSERT[B,A[I]]; ENDFOR
   (a) (5) How long (in Θ-land) does the INSERT step take as a function of I. Brief reason, please!
   `Solution:`$O(\lg I)$ as the heap has size I at that time.

(b) (10) Give the total time for the algorithm as $\Theta(g(n))$ for a nice $g(n)$. Give full arguments for both upper and lower bounds.
`Solution:`We get $\sum_{i=1}^{n} \lg i$. An upper bound is $n \lg n$. A lower bound by the halving method is $(n/2)\lg(n/2) \sim (n/2)\lg n$ (Stirling does even better) so the anwer is $\Theta(n \lg n)$.

(c) (5) Is this a good method to create a maxheap? Brief reason please.
`Solution:`No, as the method given in class is $\Theta(n)$ time.

3. (10) Dr. Stingy creates a Hash Table of size n (initially empty) with doubly linked lists to register vaccine applicants. n^3 people register.

(a) (5) How much time (in Θ-land) does the registration of the n^3 people take. Brief reason please.
`Solution:`$Theta(n^3)$. Each insertion takes $O(1)$ as the length of the list is immaterial.

(b) (5) William Gates arrives to get his vaccine, but he hasn't registered. How long will it take (on average) to determine that he hasn't registered. Brief reason please.
`Solution:`$\Theta(n^2)$. The average list has $n^3/n = n^2$ length and we have to go through the list Gates hashes to in order to be certain he isn't there.

4. (10) *Assume* the existence of an algorithm $QT[A, p, r]$ which produces an i, $p \leq i \leq r$, such that $A[i]$ is precisely the first quartile of the values $A[p \cdots r]$. (That is, a quarter of the $A(j)$ are $\leq A[i]$, the rest are $> A[i]$.

(a) (5) Write a variant $VQ[A, p, r]$ of quicksort that uses QT and sorts $A[p \cdots r]$.
`Solution:`The problem as stated was ambiguous. I'll here assume that QT also partitions the values so that when $j < i$ we have $A[j] \leq A[i]$, and when $j > i$ we have $A[j] > A[i]$.
IF $p < r$ (* else just one value, do nothing *)
$q \leftarrow QT[p, r]$
$QT[p, q-1]$
$QT[q+1, r]$

(b) (5) Further, assume QT takes $4n$ comparisons when applied to n data points. Let $T(n)$ be the total number of comparisons

for your $VQ[A, p, r]$. Give a recursion (don't worry about initial values) for $T(n)$. (Note: Do *not* attempt to solve the recursion!)
`Solution:`$T(n) = 4n + T(n/4) + T(3n/4)$ as the recursive calls are to sizes $n/4$, $3n/4$. (Technically with floors and ceilings, but no points deducted on that.)

5. (10) Suppose that in implementing the Huffman code we weren't so clever as to use Min-Heaps. Rather, at each step we found the two letters of minimal frequency and replaced them by a new letter with frequency their sum. (That is, use the "standard" method to find the minimum of a set of numbers and apply it twice.) How long (reasons, please!) would that algorithm take, in Θ-land, as a function of the initial number of letters n.
`Solution:`When there are i elements left it takes $O(i)$ to find the minimum, $O(i)$ for the second minimum, $O(1)$ to insert the sum, for a total of $O(i)$. Here i ranges from n down to 2 so the total time is $\sum_{i=2}^{n} O(i)$ which is $O(n^2)$.

6. (10) Let two strings X, Y in the English alphabet both begin with q. Give a *logical argument* why there is a longest common subsequence of X, Y which uses the first q in both sequences. (An example will help!)
`Solution:`There are two cases. If a common sequence used neither of the initial q we could add q to the left of both sequences, getting a longer one. e.g.: qaDbOcG and qxyDzwOrsG have DOG but they have the longer qDOG. In the second case, suppose wlog the first letter in the first sequence is used but not the first letter in the second sequence. Then we could replace the first q used in the second sequence with the first letter (also q) in the second sequence. e.g: qaDbOcG and qoqDbOcG have common qDOG using qo**q**DbOcG but also using **q**oqDbOcG.