OLD MIDTERM
SOLUTIONS

1. (20) These questions concern a max-heap $A$ with auxiliary structures $\text{length}[A]$, $\text{Left}[i]$, $\text{Right}[i]$ and $\text{Parent}[i]$.

(a) (5) Give the max-heap property. (That is, what is the relationship between the values $A[i]$ that one needs for a heap.)


(b) (5) Suppose $\text{length}[A]=100$ and the values $A[i]$ are distinct. What are the possible $i$ for which $A[i]$ is the third largest value? (You must give all possible $i$.)

Solution: $i = 2, 3, 4, 5, 6, 7$ (Error: The problem had mistakenly asked for the second largest – corrected Tuesday)

(c) (10) Describe the algorithm $\text{MAX-HEAP-INSERT}(A, \text{key})$ which adds to heap $A$ a new entry with value $\text{key}$. (Either a description in words or a pseudocode program would be fine.)

Solution: See text

2. (15) These questions concern QUICKSORT. We will take $\text{PARTITION}(A, p, r)$ as given. (This procedure rearranges $A[p \ldots r]$, placing all entries less than $A[r]$ to the left and all entries greater than $A[r]$ to the right. You are not being asked to write it!!)

(a) (5) Give the recursive procedure for $\text{QUICKSORT}[A, p, r]$.

Solution: See text

(b) (10) Let $T(n)$ denote the average time for procedure $\text{QUICKSORT}$ on an array of length $n$. Give a recursive formula for $T(n)$ and (important!) give a description in words of where that recursive formula comes from.

Solution: The recursion is

$$T(n) = n - 1 + \frac{1}{n} \sum_{i=1}^{n} (T(i) + T(n - i))$$

The pivot element is equally likely to be the $i$-th element, $1 \leq i \leq n$. Regardless, there will be $n - 1$ comparisons between the pivot and the other elements. Then we need apply $\text{QUICKSORT}$ on $i - 1$ elements, taking $T(i)$, and on $n - i$ elements, taking $T(n - i)$. 

3. (15) Consider the following procedure (* is multiplication):

\[
\text{FOR } I=1 \text{ TO } N \\
\quad J=I \\
\quad \text{WHILE } J \leq N \\
\quad \quad \text{do } J=2*J \\
\quad \text{END WHILE} \\
\text{END FOR}
\]

(a) (5) How many times is the WHILE step reached for a given \( I \) and \( N \)

**Solution:** After we’ve hit the WHILE step \( t \) times we have \( J = 2^t I \). When \( 2^t I > N \), that is, \( t \geq \lg(N/I) \) the WHILE will stop. So this is at \( \lceil \lg(N/I) \rceil \). As \( t \) starts at zero the number of times is \( \lceil \lg(N/I) \rceil + 1 \). (In grading we didn’t worry about the +1.

(b) (5) Write the total number of times the WHILE step is reached as a summation.

**Solution:** As \( I \) is running from 1 to \( N \) it is

\[
\sum_{I=1}^{N} (1 + \lceil \lg(N/I) \rceil)
\]

(c) (5) Evaluate that sum in \( \Theta \)-land.

**Solution:** It is \( \Theta(N) \). The +1 and the ceiling can each only effect each term by 1 and so the total sum by at most \( 2N \). So basically we are looking at

\[
\sum_{I=1}^{N} \lg(N/I) = \lg(N^N/N!)
\]

Applying Stirling’s Formula (there are other methods) we get \( \lg(N^N/N!) \sim \lg(e^N) \sim N \lg e = \Theta(N) \) so the total sum is \( \Theta(N) \).

4. (20) These questions concern a binary search tree \( T \) in which each node \( x \) has a key value denoted \( \text{key}[x] \). We assume \( \text{Root}[T] \), \( \text{left}[x] \), \( \text{right}[x] \), \( \text{parent}[x] \) are part of the structure.

(a) (5) State the binary-search-tree property. (That is, the condition that the keys are required to fulfill.)

**Solution:** For each \( x \) all \( y \) in the left subtree of \( x \) have \( \text{key}[y] \leq \text{key}[x] \) and all \( z \) in the right subtree of \( x \) have \( \text{key}[z] \geq \text{key}[x] \) and
(b) (5) Suppose the tree has a million records. What is the smallest the height of the tree (recall this is defined as the maximal distance from the root to a leaf) can be? (Give a precise integer!)
Solution: As \(10^6 < 2^{20}\) by a little we could have a tree with height 19 as if we had a full tree with that height it would have a total of \(2^{20} - 1\) nodes.

(c) (5) Describe a procedure \(\text{MIN}\) that returns that node \(x\) with minimal value of \(\text{key}[x]\).
Solution: Start at \(x = \text{Root}[T]\). While \(\text{left}(x) \neq \text{NIL}\), \(x \leftarrow \text{left}(x)\). After the ENDWHILE return \(x\).

(d) (5) Suppose \(T\) has height \(h\). In the (warning: a bit tricky!) best-case, how long would your \(\text{MIN}\) procedure above take. Give a nice picture of \(T\) in that best case.
Solution: Time 1. Maybe the root has no left child, in which case the WHILE loop will stop immediately. One picture (among many) would be the tree being a path going always to the right starting at the root.

5. (25) Prof. Squander decides to do Bucket Sort on \(n\) items with \(n^2\) buckets while his student Ima Hogg decides to do Bucket Sort on \(n\) items with \(n^{1/2}\) buckets. Assume that the items are indeed uniformly distributed. Assume that Ima’s algorithm for sorting inside a bucket takes time \(O(m^2)\) when the bucket has \(m\) items.
Solution: See the website. This was an assignment!

(a) (10) Argue that Prof. Squander has made a poor choice of the number of buckets by analyzing the time of Bucket Sort in his case.

(b) (10) Argue that Ima has made a poor choice of the number of buckets by analyzing the time of Bucket Sort in her case.

(c) (5) Argue that Ima uses roughly the same amount of space as someone using \(n\) buckets.

6. (15) Let \(x_1 x_2 \cdots x_n, y_1 y_2 \cdots y_n\) be two sequences of zeroes and ones. For \(1 \leq a \leq n\) and \(1 \leq b \leq n\) let \(\text{LCS}[a, b]\) denote the length of the longest common subsequence of \(x_1 \cdots x_a\) and \(y_1 \cdots y_b\).

(a) (5) Give an efficient algorithm to find \(\text{LCS}[a, 1]\) for \(1 \leq a \leq n\). How long (in \(\Theta\)-land) does your algorithm take.
Solution: One could just use the full algorithm. But it is easier to initialize \( LCS[0,1] = 0 \). Then FOR \( a = 1 \) to \( n \), IF \( LCS[a-1,1] = 1 \) OR \( x_a = y_1 \) set \( LCS[a,1] = 1 \), ELSE set \( LCS[a,1] = 0 \). (In words, go along until you find \( y_1 \). Until you do the \( LCS \) is 0, once you do it is 1 forevermore. This is a single pass and so takes time \( \Theta(n) \).

(b) (10) Assume \( LCS[a,1] \), \( 1 \leq a \leq n \) and \( LCS[1,b] \), \( 1 \leq b \leq n \) are already known. For \( 2 \leq a,b \leq n \) give a recursion for \( LCS[a,b] \) in terms of \( LCS[a,b-1] \), \( LCS[a-1,b] \) and \( LCS[a-1,b-1] \). Give a clear argument for why this recursion is correct.

Solution: See text.