MIDTERM (with solutions)

Please answer all 6 questions, worth 140 points in all. Read carefully. Keep your scratch work neat — they may come in useful. This is a closed book exam, but you may refer to a prepared 8"X11" 2-sided sheet of notes. Go to it!

**Problem 1** TRUE or FALSE [5 points each]

Minus 3 points for wrong answers. Brief justification is needed for full credit.

(a) Let $T$ be an array that represents a min-heap (i.e., the minimum item is at the root of the heap). If we reverse the elements of the array, we obtain a max-heap.

**ANSWER:** False. Suppose $T = [1, 2, 4, 3]$, which represents a min-heap. But $[3, 4, 2, 1]$ is not a max-heap.

(b) A binary tree with $n$ nodes has $n - 1$ edges.

**ANSWER:** True: each node, except the root, has an edge to its parent, and this accounts for all the edges.

(c) The sum $S(n) = \sum_{i=1}^{n}(i/\lg i)$ is $O(\lg(n!))$.

**ANSWER:** False: $S(n) = \Theta(n^2/\lg n)$ since it is a polynomial sum. But $\lg(n!) = \Theta(n \lg n)$.

**Problem 2** SHORT QUESTIONS [10 points each]

(a) What is a primitive 16th root of unity modulo $M = 2^{64} + 1$?

**ANSWER:** $2^8$ is a primitive 16th root of unity mod $2^{64} + 1$. In general, $2^{2L/K}$ is a primitive $K$th root of unity mod $2^L + 1$.

(b) Give the domain transformation that transforms $T(n) = T(\sqrt{n}) + n$ to standard form. Also, state the standard form (but don't solve it).

**ANSWER:** Change the $n$-domain to $N$-domain where $N = \lg \lg n$. If $t(N) = T(n)$ then we have $t(N) = t(N - 1) + 2^{2^N}$.

(c) Apply the Master Theorem to solve the following recurrence: $T(n) = 6T(n/3) + n^2$. You must justify the relevant case of the Master Theorem used.

**ANSWER:** If the recurrence is $T(n) = f(n) + a \cdot T(n/b)$ then case (+1) of the Master theorem is where the function $f(n)$ satisfies the regularity condition $af(n/b) \le c \cdot f(n)$ for some constant $c < 1$. In our case, the regularity conditionis satisfied with the constant $c = 2/3$: $6(n/3)^2 = (2/3) \cdot n^2 = c \cdot f(n)$.

(d) In Skip Lists, we construct a hierarchy of lists $L_0 \supseteq L_1 \supseteq L_2 \supseteq, \ldots, \supseteq L_m = \emptyset$ by a randomized process so that $m$ is expected to be $O(\lg n)$. Why don't simply pick every other element in the list $L_i$ to put into $L_{i+1}$?

**ANSWER:** This deterministic property cannot be maintained when arbitrary elements are inserted/deleted dynamically.

(e) Suppose we relax the definition of a "complete binary tree" as follows: if level $i$ is not full and it has $k$ nodes, then level $i+1$ has at most $k/4$ nodes. Give an upper bound on the height of such a tree if it has $n$ nodes.

**ANSWER:** There are at most $\lg(n)$ levels that are full. There are at most $\log_4(n)$ levels that are not full. Hence height is at most $\lg(n) + \log_4(n) = O(\log n)$. CAN YOU GIVE A SHARPER BOUND?

**Problem 3** HEAPS [20 points]

Here is an array $T = [6, 4, 7, 9, 8, 10, 2, 5, 1, 3]$ with 10 keys.

(a) Draw it as a complete tree. This is not yet a min-heap.

(b) Construct a min-heap from this array using the bottom-up method. Show the intermediate

heaps (drawn as complete binary trees) after each stage of the construction. A "stage" is when we heapify all subtrees at a given level.
(c) Show the result after we remove the minimum item.
**ANSWER:** (a) Omitted, but the root is 6. (b) After stage 1, the array is $[6, 4, 7, 1, 3, 10, 2, 5, 9, 8]$. Note: although we only show an array here, you are supposed to draw this array as a binary tree. After stage 2, the array is $[6, 1, 2, 4, 3, 10, 7, 5, 9, 8]$. Finally, the array is $[1, 3, 2, 4, 6, 10, 7, 5, 9, 8]$.
(c) After removing the min, we get $[2, 3, 7, 4, 6, 10, 8, 5, 9]$.

**Problem 4** SUMMATION TECHNIQUE [15 points]
Show that $H_n \to \infty$ as $n \to \infty$. HINT: For any integer $k \geq 1$, if $n = 2^k$ then $H_n \geq k/2$. Break up the terms of $H_n$ into $k$ groups.
**ANSWER:** Assume $n = 2^k$ and break up the terms of $H_n$ into $k$ groups $G_0, G_1, \ldots, G_{k-1}$, where the $i$-th group has $2^i$ consecutive terms. Thus $G_0 = 1$ and $G_1 = (1/2) + (1/3)$ and $G_2 = (1/4) + (1/5) + (1/6) + (1/7)$. But each term in $G_i$ is at least $2^{-i-1}$ and there are $2^i$ terms. Hence the sum of the terms in $G_i$ is at least $1/2$. Hence $H_n \geq k/2$.

**Problem 5** INDUCTION PROOF [5+15 points]
(a) Let the inorder and preorder traversal of a binary tree $T$ with 10 nodes be $(a, b, c, d, e, f, g, h, i, j)$ and $(f, d, b, a, c, e, h, g, j, i)$, respectively. Draw the tree $T$.
**ANSWER:** To avoid drawing trees, I will give a linear representation of trees using parentheses: write "$r(L, R)$" for the binary tree with root $r$ and whose left and right subtrees are linearly represented by $L$ and $R$, respectively. If $r$ has no children, we just write "$r$", and if $r$ has only one child, we may write "$r(L, )$" or "$r(, R)$". Returning to our question, the tree $T$ has the form $f(\cdots, \cdots)$ (i.e., $f$ is the root). But what are the left and right subtrees? Expanding again, we see $T = f(d(\cdots, \cdots), h(\cdots, \cdots))$. Eventually, we get $T = f(d(b(a, c), e), h(g, j(i, )))$.
(b) Assume that $T$ is a binary tree with $n$ distinct nodes. Show by induction on $n$ that, given the inorder $\mathrm{In}(T)$ and preorder $\mathrm{Pre}(T)$ listing of the nodes of $T$, we can reconstruct the tree $T$. HINT: if $r$ is the root of $T$ and the left and right subtrees of $T$ are $T_L$ and $T_R$ (respectively), then $\mathrm{In}(T) = (\mathrm{In}(T_L), r, \mathrm{In}(T_R))$ and $\mathrm{Pre}(T) = (r, \mathrm{Pre}(T_L), \mathrm{Pre}(T_R))$.
**ANSWER:** Note that the issue here is how to deduce $\mathrm{Pre}(T_L)$ and $\mathrm{Pre}(T_R)$ from $\mathrm{Pre}(T)$, and similarly for $\mathrm{In}(T)$. In other words, even though we know that $\mathrm{Pre}(T) = (r, \mathrm{Pre}(T_L), \mathrm{Pre}(T_R))$, we do not know where $\mathrm{Pre}(T_L)$ ends in the string $\mathrm{Pre}(T)$! All that we need is the length of $\mathrm{Pre}(T_L)$, and this we can obtain from $\mathrm{In}(T_L)$.

Here then is the proof: The result is trivial for $n = 0$ and $n = 1$. Assume $n > 1$. Let the left and right subtree of $T$ be $T_L$ and $T_R$. We know the root $r$ of $T$ by looking at $\mathrm{Pre}(T)$. Hence $\mathrm{In}(T) = \mathrm{In}(T_L) r \mathrm{In}(T_R)$. This gives us $\mathrm{In}(T_L)$ and $\mathrm{In}(T_R)$. Hence, we know know the number of nodes in $T_L$ and $T_R$. It follows that we can deduce $\mathrm{In}(T_L)$ and $\mathrm{In}(T_R)$ from $\mathrm{In}(T)$. Now, by induction, we can construct $T_L$ and $T_R$ from their inorder and preorder lists. Hence $T$ can be reconstructed.

**Problem 6** RECURRENCES [20 points]
Solve the following recurrence by using domain and range transformations: $T(n) = \sqrt{n}T(\sqrt{n}) + n$. HINT: this is similar to problem 2(b).
**ANSWER:** By the domain transformation, $N = \lg\lg n$ and $t(N) = T(n)$, we get $t(N) = 2^{2^{N-1}} t(N-1) + 2^{2^N}$. By the range transformation, $s(N) = t(N)/2^{2^N - 1}$, we get $s(N) = s(N - 1) + 2$. Thus $s(N) = 2N$. Thus $t(N) = 2N \cdot 2^{2^N - 1} = N \cdot 2^{2^N}$. Finally, $T(n) = t(N) = n \lg\lg n$.

NOTES: Note that this is somewhat unlike the Master recurrence where $T(n) = bT(n/b) + n$

ought to give the solution $\Theta(n \log n)$. The reason is that "$b$" in our problem is not a constant, but depends on $n$. What about the recurrence $T(n) = \sqrt{n}T(\sqrt{n}) + f(n)$ where $f(n) = n \lg n$? or, $f(n) = n \lg n \lg n$?