CSC172 2nd Midterm 2013 Answers

Please write your name on the bluebook. There are 75 points (minutes). Two sides of handwritten notes allowed. Stay cool and please write neatly.

1 Heaps: 20 pts.

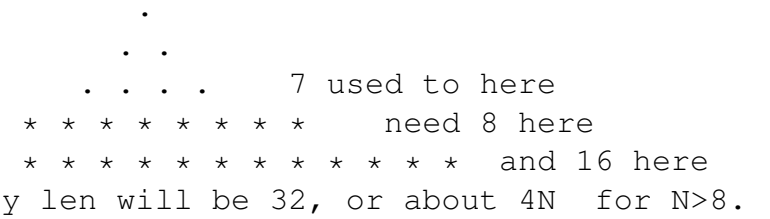
A (10 min). The array implementation of a complete binary tree of N elements uses N array positions 1 to N. Consider incomplete trees: how large must the array be to deal with the following (5 min) cases:

A.1. A binary tree of N nodes that has two extra levels (slightly unbalanced).
A.2. The worst case binary tree (it forces the largest heap array) of size N.
A.3. A binary tree with deepest node at $3 \log N$.

B. (10 min.) In a binary MaxHeap, consider the minimum item in the heap: Show

B.1. it must be at one of the leaves.
B.2. every leaf must be examined to find it.
B.3. there are exactly $\lceil N/2 \rceil$ leaves.

Ans. A.1. Ignoring the technical $\pm$ 1 for levels vs. depth, a simple diagram shows we need up to 4N (multiplicative constant). It looks something like



Analytically, we need $2^{\log N+2}$.
Or: $2^{\lceil logN \rceil+1}$.
That evaluates to 4N, a multiple of the number of meaningful elements.

A.2. Here, no need to get that weird "depth" involved. A worst-case binary tree with N elements (entries) has N levels, which means $2^N - 1$ nodes, plus one if you like for the Array[0] position, get 2^N. So if N = 3, need array of length 8, an exponential function of length.

A.3. Again there's the unfortunate fact that the deepest node in a tree with L levels is at depth L-1 (root is at depth 0). Taking the question literally, we'd get $2^{3 \log N+1}$
or $2^{3 \log N}$ if we ignored the subtlety of how to define "depth". Since $3 \log N$ is $logN^3$, we get a polynomial function: required length is N^3 or $2N^3$.

B.1. If not it'd be above some smaller-valued node and would not obey the PO Tree property.

B.2. The minimum item can be put into any leaf by choosing the order of insertions.

B.3. Clearly true in complete (full, totally balanced, $N = 2^k$ type tree. There are 2^{k-1} leaves, which is $\lceil N/2 \rceil$. The first leaf added to a parent keeps the number of leaves constant. The second adds a leaf, but that can only be done for 2^{k-1} old leaves, now parents. This exactly doubles the number of leaves, creates a full tree as above, and thus leaves (har) the assertion true.

2 Sorts: 15 pts

A (5 min). What is the running time of heapsort for presorted input? (The input is in ascending order. Use a Minheap).

B. (10 min) What is the running time of shellsort (number of swaps) for reverse-ordered input? Assume increments of (N/2, N/4, ..., 2, 1).

Ans

A. buildheap does no swapping but the deletemins are are still a log N proposition, so NLogN.

B Ans. Shellsort is just a bunch of insertion sorts. For a given increment I, there will be I subarrays to sort by insertion, each of length N/I. We know that insertion sort requires time $O(m^2)$ to sort a reverse-sorted array of length m. Here, m will be (N/I) for eachsub array. Thus one subarray will cost $N/I)^2$ to sort. There are I sub arrays, so the total cost will be $I * (N/I)^2 = n^2/I$. But that is the cost just for a single gapSize. The total time for all of the iterations must be $N^2/(N/2) + N^2/(N/4) + N^2/(N/8) + ... + N^2/2 + N^2/1 = 2N + 4N + ... + N^2/2 + N^2/1$. If we factor out an N, we get $N(2 + 4 + ... + N/2 + N)$. In parens is sum of powers of 2 from 2 to N, which is approximately equal to 2N. Therefore, the total cost is $N(2N) = 2N^2 = O(N^2)$.

3 Searching: 20 pts

A (15 pts): We have a graph of V vertices with costs on its E edges. Compare and contrast Depth First Search and Breadth First Search for:

A1 (5 pts): Data structures and programming concepts required to *implement the search* (not implement the graph).

A2 (5 pts): Worst-case Big-Oh **space** (in terms of V or E or both) required to search the graph. Justify your answer: what is the worst case graph for DFS and BFS?

A3 (5 pts): Worst-case Big-Oh **time** (in terms of V or E or both) required to search the graph.

B (5 pts): Is uniform cost search more like DFS or BFS? What data structure is natural for its implementation?

Ans. A1. DFS: stack and (or) recursion. BFS: queue and iteration

A2 Although we pay (depth-limited) DFS in the hope it only needs about O(logN) space (as many recursive calls as the depth), in the Worst Case the depth is N... e.e. a binary tree in which all nodes have only a left child.

Thus O(V) for each type of search. worst for dfs is list, worst for BFS is also a list, or depending on the edges, a clique... or anything!

A3 BFS: O(V+E) or O(E): linear DFS: O(E), also linear in size of graph.

B: it's like BFS, priority queue is natural.

4 Recurrence and Loop Analysis: 10 pts

Note: given in class earlier. Here's some C-ish pseudocode.

This function takes three parameters: 1: A character array. 2: The starting index of a substring (subarray). 3: The ending index of the substring.
Array indices start at 0. A sample (and top-level) call: `disturb('abcd', 0, 3);`

```
disturb(char a[], int i, int n) {
   int j;
   if (i == n)
     printf("%s\n", a);
   else
   {  for (j = i; j <= n; j++)
       {  swap(a[i], a[j]); //swap 2 chars
          disturb(a, i+1, n);
          swap(a[i], a[j]);  //(un)swap 2 chars
 } }   }
```

A (5 pts): Do loop analysis to derive a recurrence equation for the growth rate (as a function of N) for the algorithm `disturb`. Solve it by inspection.

B (5 pts): In one word, what is `disturb` computing?

Ans:

One of the more common errors was to compute the loop cost by adding the T(n-1) costs of the body, rather than multiplying each by n.

A. T(n) = n*T(n-1) +O(1)
T(0) = 1,
so T(n) = n!

C. permutations, which the n! should remind us of.

5 Hashing: 5 pts

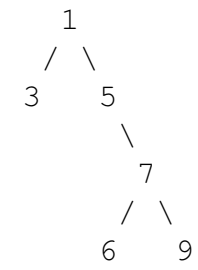
Cubic probing uses the probing sequence $hash(x)+i^3, \quad i = 1, 2, ...$ Does this method improve on quadratic probing's secondary clustering behavior? Why or why not?

Ans. no improvement. Secondary clustering is caused by any repeated pattern of probes to resolve a collision. So it's endemic unless some more randomization is applied, like secondary hashing. True the cubes grow faster, but they're modded out like the squares and always repeat themselves for any collision.

6 Disjoint Sets: 15 min.

A (10 pts): Perform the following unions on the set 0,...,7 using `union by size`: (several right answers)
U(0,1); U(4,5); U(2,3); U(6,7); U(0,3); U(5,2).

B (5 pts): Show the effect of find(9) performed on the following tree, using path compression.



Ans. A. Lots of right answers depending on how pick children and parents. My answer looks like the two-tree forest: Lots of people forgot the little 6-7 tree in the union.

B. 3,5,7,9 all become direct children of root 1.