Week 4: Lab

Collaboration level 0 (no restrictions). Open notes.

1. Consider a heap of height h, where the height is defined as the number of edges on the longest root-to-leaf path (for e.g., a heap of 1 element has height=0, a heap of 2 or 3 elements has height=1, and so on).

   (a) What is the minimum number of elements in the heap, as a function of h?

   (b) What is the maximum number of elements in the heap, as a function of h?

   (c) Use these to derive an $O()$ and $\Omega()$ bound for h as function of n, and argue that an n-element heap has height $\Theta(\lg n)$.

2. Where in a min-heap might the largest element reside, assuming that all elements are distinct?

3. Is an array that is in sorted order a min-heap?

4. Argue that the leaves in a heap of n elements are the nodes indexed by $\lfloor n/2 \rfloor + 1, ..., \lfloor n/2 \rfloor + 2, ..., n$. (Hint: what is the parent of the last element?)

5. **Heap Insert:** Assume you call HEAP-INSERT (A, 4) on the following min-heap $A = [3, 4, 9, 5, 10, 15, 12, 8, 20, 11, 12]$. What is the resulting array?

6. **Heap Delete-Min:** Assume you call DELETE-MIN (A) on the following min-heap $A = [3, 4, 9, 5, 10, 15, 12, 8, 20, 11, 12]$. What is the resulting array?

7. **Heapify:** Assume you call HEAPIFY(A, 3) on the following min-heap $A = [2, 4, 6, 10, 5, 2, 4, 12, 15, 9, 6, 4, 5, 5, 6]$. What is the resulting array?

8. **Buildheap:** Assume you call Buildheap(A) on the following array. $A = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]$. What is the resulting array A after Buildheap is finished?

9. Illustrate the operation of Heapsort on the array

$$A = [5, 13, 2, 25, 7, 17, 20, 8, 4]$$

10. How would you implement a function that searches for a given element in a heap, and how long would it take in the worst case?

11. Given a node x in a min heap, with children nodes l and r, what does the heap property tell us about $x.key, l.key, r.key$?

12. Is it true that a heap of n elements can be built in $O(n)$ time?

13. Given a heap with n keys, is it true that you can search for a key in $O(\log n)$ time?

14. Give example of an operation that's supported more efficiently by heaps than by binary search trees (BSTs).

15. Give example of an operation that's supported by BSTs but not by heaps.

Below is the pseudocode for Quicksort that we saw in class. As usual with recursive functions on arrays, we see indices p and r as arguments. Quicksort(a, p, r) sorts the part of the array between p and r inclusively. The initial call (that is, to sort the entire array) is Quicksort$(A, 0, n-1)$.

```
QUICKSORT($A, p, r$)
IF $p < r$ THEN
    q=PARTITION($A, p, r$)
    QUICKSORT($A, p, q-1$)
    QUICKSORT($A, q+1, r$)
```

```
PARTITION($A, p, r$)
$x = A[r]$
$i = p - 1$
FOR $j = p$ TO $r - 1$ DO
    IF $A[j] \leq x$ THEN
        $i = i + 1$
        Exchange $A[i]$ and $A[j]$
Exchange $A[i+1]$ and $A[r]$
RETURN $i + 1$
```

16. **Quicksort Partition:** Assume you call $Partition(A, 0, 9)$ on the array $A = [3, 6, 1, 5, 8, 2, 4, 1, 3]$. What is the array after Partition is finished, and what is the value of q returned?

17. What is the running time of QUICKSORT when all elements of arrary A have the same value

18. Suppose we modify the deterministic version of quicksort so that, instead of selecting the last element as the pivot, we chose the element at index $\lfloor n/2 \rfloor$, that is, an element in the middle of the sequence. What is the running time of this version of quicksort on a sequence that is already sorted? What kind of sequence would cause this version of quicksort to run in $\Theta(n^2)$ time? Show an example of $n = 10$ elements or so that would trigger worst-case.

19. Argue that Quicksort is not *stable* by showing a small example.

20. Which of the following sorting algorithms are stable?

    - Bubblesort
    - Insertion Sort
    - Selection Sort
    - Mergesort
    - Heapsort