Lab 4

Collaboration level 0 (no restrictions). Open notes.

1 Heaps and Heapsort

1. Is a sorted array a min-heap?
2. Where in a min-heap might the largest element reside (assuming that all elements are distinct)?
3. 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?)
4. **Heap height:** The height of a heap 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). Consider a heap of height h.
   (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 (a) to derive a lower bound for h as function of n, $h = \Omega(..)$.
   (d) Use (b) to derive an $O()$ bound for h as a function of n.
   (e) From (c) and (d) conclude that an n-element heap has height $\Theta(\lg n)$.
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. **Heapsort:** Illustrate the operation of Heapsort on the array $A = [5, 13, 2, 25, 7, 17, 20, 8, 4]$

10. Given a heap with n keys, is it true that you can search for a key in $O(\log n)$ time?
11. How would you search for a given element in a heap, and how long would it take (in the worst case)?
12. Give example of an operation that's supported by binary search trees but not by heaps (could potentially run it on heaps but would not be efficient).
13. What is an operation that's supported more efficiently by heaps than by binary search trees (BSTs)?

2 Quicksort

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 parameters. Quicksort(a, p, r) sorts the part of the array between p and r inclusively. The initial call to sort the entire array $A[0...n-1]$ 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)
```

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

1. Consider the array of $n = 9$ elements

$$A = [3, 6, 1, 5, 8, 2, 4, 1, 3]$$

Assume we call Partition$(A, 0, 8)$. Show the contents of A after it returns. What is the value of q returned?

2. Consider the same array as above:

$$A = [3, 6, 1, 5, 8, 2, 4, 1, 3]$$

Simulate QuickSort$(A, 0, 8)$ and show the whole recursion tree. Show clearly all recursive calls to Quicksort and their parameters.

3. Consider an array with the elements sorted in increasing order:

$$A = [1, 2, 3, 4, 5, 6]$$

(a) Assume we call $\textsc{Partition}(A, 0, 5)$. Show the contents of A after it returns. What is the value of q returned?

(b) What is the run time of Quicksort when called on an array of n elements, sorted in increasing order? Write a recurrence and give a $\Theta()$ bound.

4. Consider an array with the elements sorted in decreasing order:

$$A = [6, 5, 4, 3, 2, 1]$$

(a) Assume we call $\textsc{Partition}(A, 0, 5)$. Show the contents of A after it returns. What is the value of q returned?

(b) What is the run time of Quicksort when called on an array of n elements, sorted in decreasing order? Write a recurrence and give a $\Theta()$ bound.

5. Consider an array with equal elements:

$$A = [2, 2, 2, 2, 2, 5]$$

(a) Assume we call $\textsc{Partition}(A, 0, 5)$. Show the contents of A after it returns. What is the value of q returned?

(b) What is the run time of Quicksort when called on an array of n elements, all with the same key? Write a recurrence and give a $\Theta()$ bound.

6. 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 $\lfloor (p + r)/2 \rfloor$, that is, an element in the middle of the chunk to be sorted.

(a) What is the running time of this version of Quicksort on a sequence that is already sorted?

(b) 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.

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

8. Which of the following sorting algorithms are stable? Bubblesort, Insertion Sort, Selection Sort, Mergesort, Heapsort.

3 Recommended, but optional

1. (Leet code #74) Searching a 2D matrix: You are given an $m \times n$ integer matrix A with the following properties: each row is sorted in non-decreasing order, and the first integer of each row is greater than the last integer of the previour row.

   Given an integer `target`, write an algorithm to return `true` if `target` is in A or `false` otherwise.

   We expect: Write your algorithm in pseudo-code with sufficient detail so that you can implement it in no time. Even better, you could try coding it in either Python or Java.

2. (Leet code #504): Given an integer num, write a function to return a string of its base-7 representation.

   Example 1: input: $num = 100$, output: "202"

   Example 2: input: $num = -7$, output: "-10"