CMPS 101: Winter 2016: HW 3 (Update 4)

Due: 12th February 2016

• The assignment is to be attempted in groups of two. If you choose to not work with a partner, one point will be automatically deducted from your score.

• Each group needs to submit only one set of solutions. Solutions need to be handed in the class on the due date.

• \LaTeX is preferred, but neatly handwritten solutions will also be accepted. All solutions need to be handed over in the class before the beginning of the lectures.

• The names of the group members, and their UCSC ID (@ucsc.edu email address) should prominently be written on the upper left corner of the first page.

• Multiple sheets should be stapled together in the upper left corner.

• Solutions to the problems should be clearly labeled with the problem number.

• Although no points are given for neatness, illegible and/or poorly organized solutions can be penalized at the graders option.

• Clearly acknowledge sources, and mention if you discussed the problems with other students or groups. In all cases, the course policy on collaboration applies, and you should refrain from getting direct answers from anybody or any source. If in doubt, please ask the instructors or TAs.

**Question 1 (1 + 1 points):** Recall that when we analyzed the procedure for constructing heaps, we used the following fact: A complete binary tree can have at most \( \lceil \frac{n}{2^h+1} \rceil \) nodes at any height \( h \). Here we will prove a weaker version of this fact in two steps:

- First, prove that every complete binary tree of height \( h \) has exactly \( 2^{h+1} - 1 \) nodes. Use the following conventions: a complete binary tree of height 0 is a single root, and a complete binary tree of height \( h \) contains a root node connected to two disjoint complete binary trees of height \( h - 1 \). **Hint:** Use induction on the height.
Next, use the previous result to prove that in an \( n \) element heap, there are at most \( \left\lceil \frac{n}{2^h+1} \right\rceil \) nodes of height \( h \).

Note that we are proving a weaker result than the one we used in the class. For instance, the above result says that there are at most \( \left\lceil \frac{n}{2^1+1} \right\rceil = n \) nodes at the leaf level (height \( h = 0 \)), while the result in the book claims a tighter \( \left\lceil \frac{n}{2} \right\rceil \) bound. Similarly, at height \( h = 1 \), the above result gives a \( \left\lceil \frac{n}{4} \right\rceil \) bound while the claim in the book gives a \( \left\lceil \frac{n}{4} \right\rceil \) bound.

**Question 2 (1 + 1 points):** Consider sorting an array with the following values: \([6, 2, 9, 5, 7, 10, 4]\). Show the sequence of steps used by heapsort, and quicksort to sort this array. You may assume that quicksort using the partitioning scheme described in section 7.1 of the book. For heapsort, I am looking for an illustration similar to Figure 6.4 in the book. For quicksort, it is sufficient to show the end result of each partitioning step (see e.g., Figure 7.1 (i)).

**Question 3 (1 point)** Recall that a priority queue has to implement the following four functions:

- \( \text{INSERT}(S, x) \)
- \( \text{MAXIMUM}(S) \)
- \( \text{EXTRACT-MAX}(S) \)
- \( \text{INCREASE-KEY}(S, x, k) \)

Write pseudo-code to show how you will implement these four operations using a linked list. Analyse the time complexity of each of the operations. Compare it with the time complexity of the heap implementation which we discussed in the class.

**Question 4 (1 point)** Consider a max binary heap of \( n \) elements, stored in an array.

- Write pseudo-code for an algorithm that deletes an element at an index \( 1 \leq k \leq n \). The heap property must be preserved after the deletion of this element.
- What is the time complexity of your algorithm? Justify.

**Changelog**

- Update 1: Fixed a missing – sign in bullet 2 of question 1.
- Update 2: Clarified the connection between the bounds in problem 1 and the claim in the book.
- Update 3: Fixed a typo in problem 3. Insert only requires two parameters.
- Update 4: Fixed a typo in problem 3. Rename \( k \) to \( x \) to follow the convention in the book.