Lab 9 Solutions

Due Tuesday 9/17, 2 PM

1. Starting with an initially empty BST, imagine loading the BST using the following insertion sequence

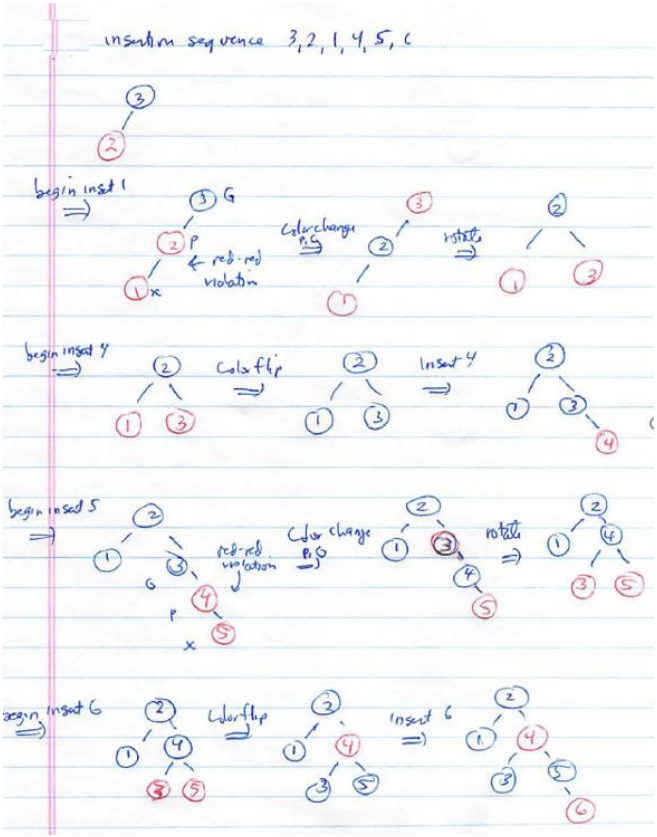
   $$a_1, a_2, \ldots, a_s, b_1, b_2, \ldots, b_t$$

   where $a_1 > a_2 > \ldots > a_s$ and $b_1 < b_2 < \ldots < b_t$, and $a_1 < b_1$ and $s + t = n$ and $s = t$. For this BST, what is the worst-case asymptotic running time for searches and insertions?

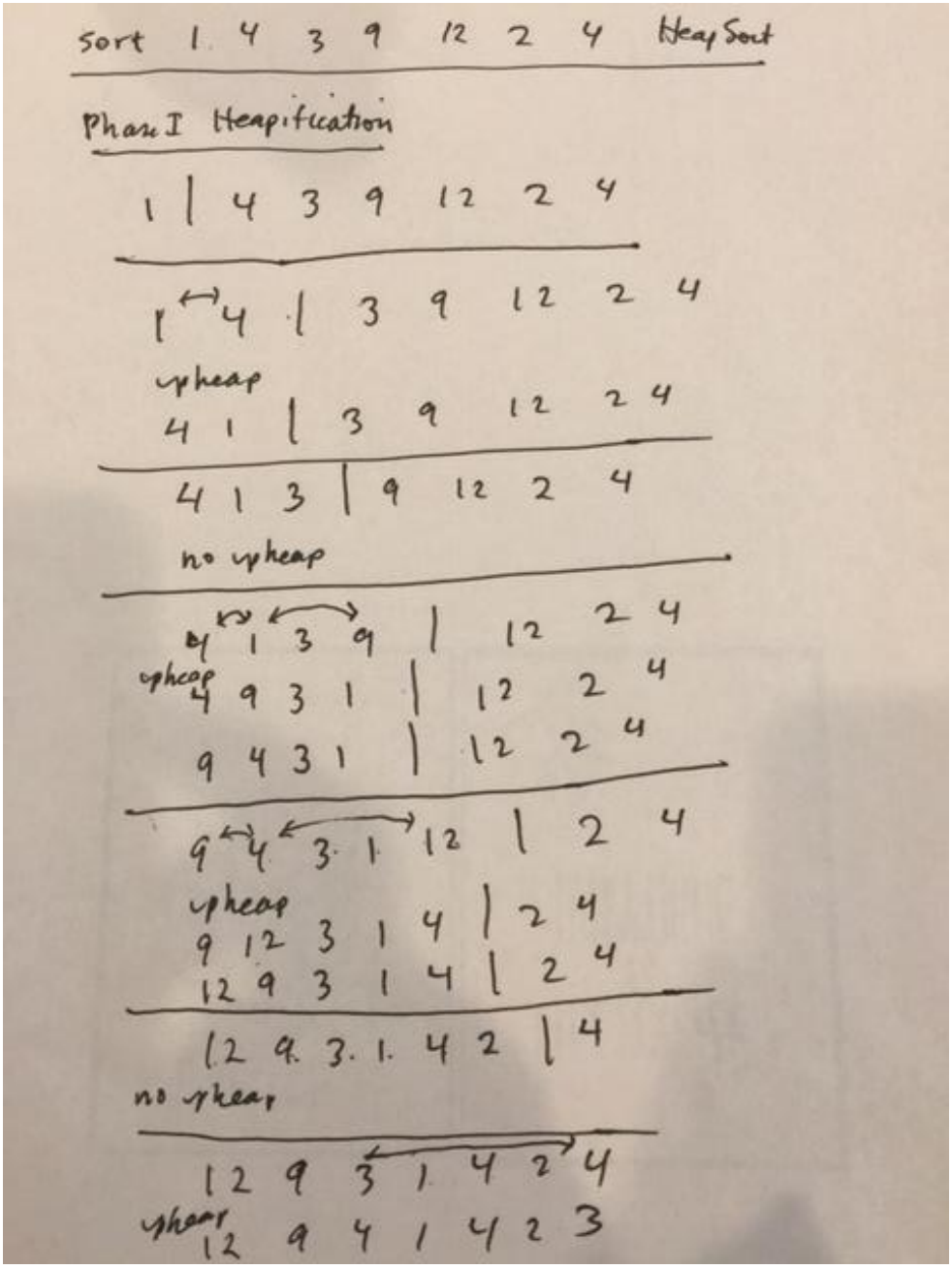
   Solution: In a worst case, the element being searched for is greater than b_t and will not be found. The BST search will be forced to visit each element among $b_1, b_2, \ldots, b_t$ – in other words, execution will require at least n/2 steps. Therefore the worst case asymptotic running time is $\Theta(n)$. Likewise, a worst case for insertion occurs when an element larger than b_t is inserted. In that case once again each element among $b_1, b_2, \ldots, b_t$ is visited, and so the worst case running time is $\Theta(n)$.

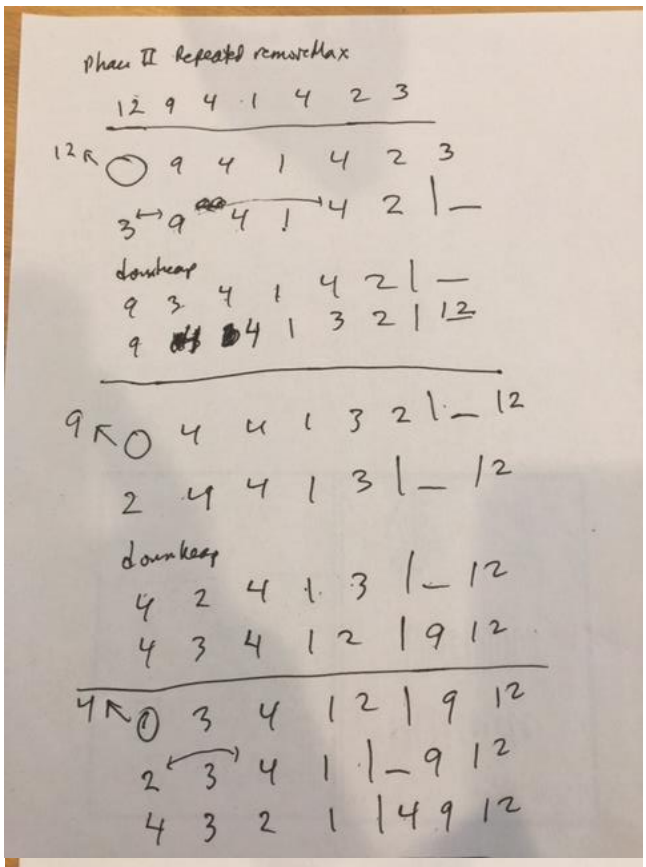
2. Use the insertion algorithm for red-black trees to successively insert the values 3, 2, 1, 4, 5, 6 starting with an empty tree.

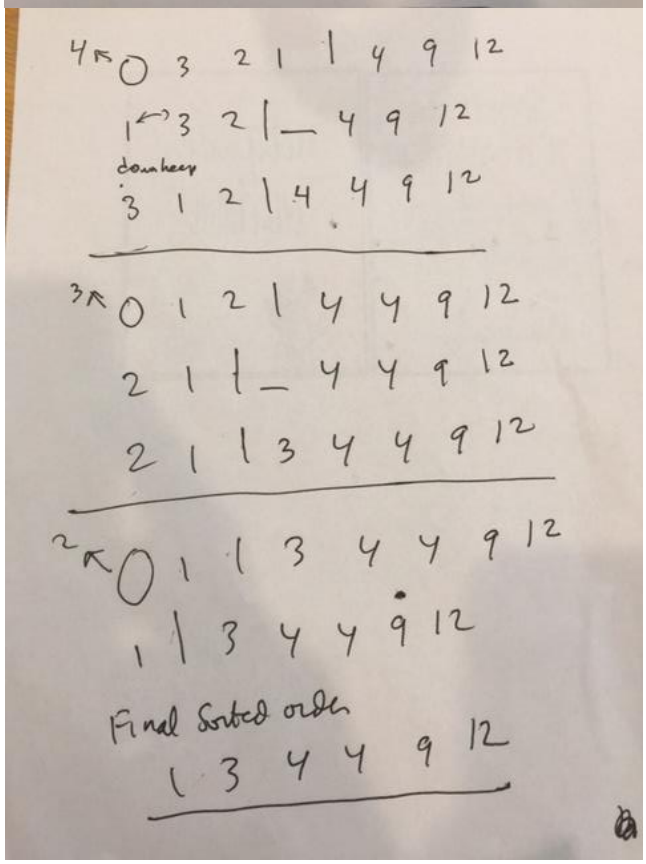
   Solution:



3. Carry out the array-based version of HeapSort on the input array [1, 4, 3, 9, 12, 2, 4]. Indicate clearly your steps and separate Phase I (heapification) from Phase II (in-place sorting).




Phase II Repeated removeMax
12 9 4 1 4 2 3
12↖ 9 4 1 4 2 3
3↔9 4 1 4 2 |_
downheap
9 3 4 1 4 2 |_
9 4 4 1 3 2 | 12
9↖ 4 4 1 3 2 |_ 12
2 4 4 1 3 |_ 12
downheap
4 2 4 1 3 |_ 12
4 3 4 1 2 | 9 12
4↖ 3 4 1 2 | 9 12
2↔3 4 1 |_ 9 12
4 3 2 1 | 4 9 12


4↖ 3 2 1 | 4 9 12
1↔3 2 |_ 4 9 12
downheap
3 1 2 | 4 4 9 12
3↖ 1 2 | 4 4 9 12
2 1 |_ 4 4 9 12
2 1 | 3 4 4 9 12
2↖ 1 1 3 4 4 9 12
1 | 3 4 4 9 12
Final Sorted order
1 3 4 4 9 12

4. *Interview Question.* Devise an algorithm to solve the following problem. Input is a set S = {s_0, s_1, . . ., s_{n-1}} of positive integers and a non-negative integer k. Output is *true* if there is a subset T of S whose sum is k; *false*, otherwise. What is the running time of your algorithm? Write your algorithm in pseudo-code.

   Solution. Different solutions are possible – we will discuss more in a later lesson. The brute force approach is the following:

```
P ← obtain all subsets of S
for each X in P
    if sum(X) = k, return true
return false
```