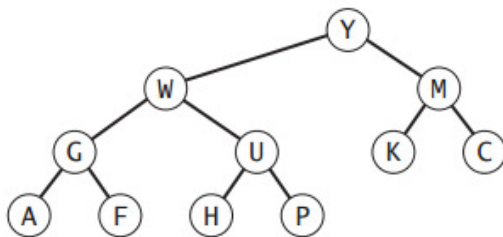


COS 226 – Data Structures and Algorithms
Fall 2014 – Flipped Lecture Section
Individual or small group worksheet week 4 – 10.02.14
25 minutes

Instructions: Answer these solo or in small groups (2-3).

1. Apply 2-way partitioning to the set: S S S O B E R R Show the array after each exchange in one partition run.
2. Apply 3-way partitioning to the set: S S S O B E R R Show the array after each exchange in one partition run.
3. Suppose that a mysterious data set (not seen by human eye) is randomized and yet the runtime of the quicksort ended up quadratic. What could be a possible reason for this?
4. The quick-select algorithm for finding median was applied to a non-randomized data set. What is the worst case runtime of the algorithm?
5. Assuming at each partition step, we were lucky to get median as the pivot. Write down a recurrence relation that represents the quicksort and argue that it is linearithmic.
6. Assuming at each partition step, we get 1/3 and 2/3 split of the array. Write down a recurrence relation that represents the runtime of the quicksort.
7. Consider the following binary max heap. Show the original array that represents this heap and the array after Max is deleted



8. Why is it impossible to build a PQ that supports insert() and delMax() in constant time?
9. Design an implementation of a PQ that supports delMax() in constant time.
10. Suppose we use a linked list instead of an array to implement a binary heap. Do we expect the running time to be better or worse? Memory usage?

11. We can use binary search to find out where a new item should be inserted in the heap (instead of sinking one-by-one) in $\log(\log(N))$. Why don't we do this?
12. Suppose we used a linked data structure, which allows constant time insertion. Why can't we use the binary-search trick from #4?
13. Suppose we use bottom-up SWIMMING (i.e. every node at the lowest level is swum, then every node at the next level up is swum, etc.). Is this procedure guaranteed to result in a heap?
14. For each of the following sorts, write down an invariant that is true at any intermediate step (i-th step) of the execution of the algorithm
- (a) selection sort → All elements in $A[0..i]$ is _____
 - (b) insertion sort → is _____
 - (c) top-down merge sort _____
 - (d) heap sort → _____
15. Binary heap is not cache friendly. Explain why?