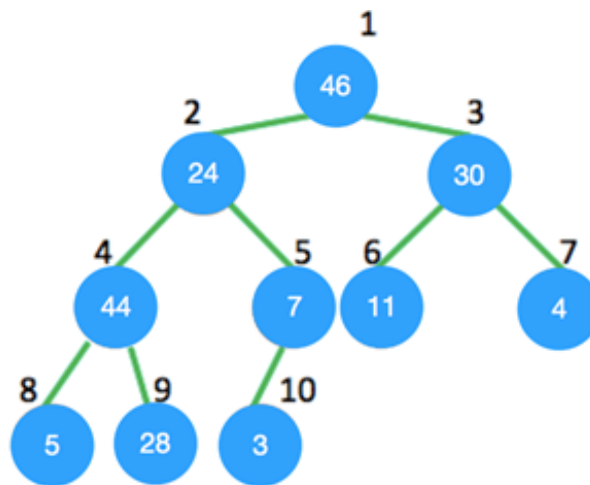


After Lecture 07 & 08 – Answer any questions on HW2 (due today)
Practice Problems (all taken from previous exams)

1. Which one of the following is false?
 - a) Heap sort is an in-place algorithm.
 - b) Heap sort has $O(n \log n)$ average case time complexity.
 - c) **Heap sort is a stable sort.** Heap sort is an in-place algorithm as it needs $O(1)$ auxiliary space.
 - d) Heap sort is a comparison-based sorting algorithm.
2. Consider the max heap shown below, the node with value 24 violates the max-heap property. Once heapify procedure is applied to it, which position will it be in?



- a) 5
 - b) 8
 - c) **9**
 - d) You cannot call heapify at the node with value 24
3. Counting sort can be used on any numeric data.
 - a) **TRUE**, it can be used, but it is a very bad idea. Counting sort requires a very large range of numbers, counting sort requires a very large array. This reduces its memory efficiency and increases space consumption. So while it possible to be used, it isn't a good idea to do so on any numeric data.
 - b) FALSE
4. Which of the following is not true about all comparison based sorting algorithms?
 - a) The minimum possible runtime growth on a random input is $O(n \log n)$.
 - b) Can be made stable by also using position when two elements are compared.
 - c) Counting Sort is not a comparison-based sorting algorithm.

- d) Merge Sort is a comparison-based sorting algorithm.
5. The BUILD-MAX-HEAP discussed in class and shown to be $O(n)$ uses this process. Call Heapify from heap index position $\lfloor \frac{heapsize}{2} \rfloor$ down to heap index position 1. Building a heap can also be implemented by starting with an empty heap and repeatedly using MAX-HEAP-INSERT to insert the elements into the heap. Consider the following implementation:

```

1: function BUILD-MAX-HEAP1( $A$ )
2:    $H \leftarrow$  empty heap (of max size  $A.length$ )
3:   for  $i = 1$  to  $A.length$  do
4:      $H.MAX-HEAP-INSERT(A[i])$ 
5:   end for
6: end function

```

- a) Do the procedures BUILD-MAX-HEAP and BUILD-MAX-HEAP1 always create the same heap when run on the same input array? Prove that they do, or provide a counterexample. **The procedures do not always create the same heap when run on the same input array.**
- b) Show that in the worst case, BUILD-MAX-HEAP1 requires $\Theta(n \lg n)$ time to build an n -element heap. **Since all but the last level is always filled, the height h of an n element heap is boundend because $\sum_{i=0}^h 2^i = 2^{h+1} - 1 = \dots$**
6. The operation HEAP-DELETE(A, i) deletes the item in node i from heap A . Give an implementation of HEAP-DELETE that runs in $O(\lg n)$ time for an n -element max-heap.

Algorithm 1 HEAP-DELETE

```

1: function HEAPDELETE( $A, i$ )
2:   swap  $A[i]$  with  $A[heapsize]$   $\triangleright$  move the item to be deleted to the last position in the heap
3:   decrement heapsize
4:    $key \leftarrow A[i]$ 
5:   if  $key \leq A[PARENT(i)]$  then
6:     HEAPIFY( $A, i$ )
7:   end if
8:   ...
9: end function

```

7. Professor Fermat has the policy of giving A's to the top $n^{\frac{1}{2}} = \sqrt{n}$ students of his class, where n is the number of students. The algorithm that he uses to determine the top $n^{\frac{1}{2}}$ students first sorts the list of students by their numerical, real-valued grade, and then picks the top $n^{\frac{1}{2}}$ students from the sorted list. This algorithm has time-complexity $O(n \log n)$ because of the sort. Can you suggest a more efficient algorithm that has time-complexity $O(n)$? Describe your algorithm informally in English and justify its time-complexity. **You can use a MaxHeap to effectively sort the students based off of the students grades which takes $O(n)$. Now perform RemoveMax \sqrt{n} times. Each RemoveMax operation takes time $O(\log N)$. Total time complexity of this algorithm is $O(n) + \log(n)\sqrt{n} = O(n)$ because $\log(n)\sqrt{n} = O(n)$, because $\log(n) = O(\sqrt{n})$**