=============================================================================

CSC 263 Tutorial 2 Winter 2019

=============================================================================

(1)

Alice claims that the \*minimum\* element of a binary max-heap must be one of its leaf nodes. Do you agree? Prove or disprove it.

Actually, the answer is "it depends". If we assume all elements are distinct  
then it is true, because if it were not a leaf, then its child is going to be  
smaller which contradicts the claim that it is minimum.  If we assume there can be duplicate elements, then the claim is false. The min's parent is not a leaf and can be equal to the min.

(2)

Bob claims that the \*median\* element of a binary max-heap must be one of its

leaf nodes. Do you agree? Prove or disprove it.

False. The median can actually be any node except the root node.   
Come up with examples of median at different positions, e.g.,   
  
    5  
  3   4  
1  2

(3)

A ternary max-heap is like a binary max-heap except that non-leaf nodes have

three children instead of two children. (As with binary heaps, there can of

course be one non-leaf node that has fewer than three children.) We refer to

the children as the left child, middle child, and right child.

The values stored in the nodes are ordered according to the same principle as

for binary max-heaps: the value at each node is greater than or equal to the

values in the node’s children. Answer the following questions regarding ternary max-heaps.

(a) Similar to binary heaps, a ternary heap is represented by an array. Given

the index i of an element in the array, what are the indices of the left child, the middle child, the right child, and the parent of the element? Assume that the array index starts at 0. Be thorough and discuss all possible corner cases.

answer:

For the node at index i, its left child is at index 3i+1 unless 3i+1 >= size of the heap (it has no left child). Similarly, the middle child is at index 3i + 2 unless 3i + 2 >= size of the heap and the right child is at index 3i + 3 unless 3i + 3 >= size of the heap.

(b) How do EXTRACT-MAX and INSERT work for a ternary max-heap? Explain their

differences from the corresponding operations for binary max-heaps. No

pseudo-code required.

answer:

EXTRACT-MAX for ternary heaps is the same as it is for binary heaps except that in the BUBBLE-DOWN operation, a smaller-than-child parent would swap with the largest of the three children, instead of the larger of two children in binary heaps.  
  
INSERT for ternary heaps also works exactly the same way as it does for binary heaps. The only difference is in how to calculate the parent’s index.

(c) Consider a function IS-TERNARY-MAX-HEAP(A) that takes an array A as the

input and returns TRUE if and only if A represents a valid ternary max-heap.

Write the pseudo-code of a recursive implementation (i.e., no loops) of

IS-TERNARY-MAX-HEAP. Briefly explain (not a proof) the correctness of your

pseudo-code and give an asymptotic upper-bound (big-Oh) on the worst-case

runtime of your pseudo-code.

The recursive version of IS-TERNARY-HEAP:  
  
def is\_ternary\_heap(A):  
    return is\_ternary\_heap\_helper(A, 0)  
  
def is\_ternary\_heap\_helper(A, i):   
    if A[i] has no child:  
        return True  
    elif A[i] has only a left child:  
        return A[i] >= A[3i+1]  
    elif A[i] has only a left child and a middle child:  
        return A[i] >= A[3i+1] and A[i] >= A[3i+2]   
    else:  
        return A[i] >= A[3i+1] and A[i] >= A[3i+2] and A[i] >= A[3i+3] \  
               and is\_ternary\_heap\_helper(A, 3i+1) \  
               and is\_ternary\_heap\_helper(A, 3i+2) \  
               and is\_ternary\_heap\_helper(A, 3i+3)  
  
Runtime analysis: Consider values of n that make the ternary tree complete  
(i.e., all levels of the tree are fully filled), the sizes of the three  
subtrees of the root are (n - 1)/3. Therefore, the recurrence of an upper-bound on the worst-case runtime is T (n) = 3T (n/3) + c. By the master theorem, it is in O(n).

(d) Write the pseudo-code of an iterative implementation (i.e., no recursion) of IS-TERNARY-MAX-HEAP. Briefly explain (not a proof) the correctness of your

pseudo-code and give an asymptotic upper-bound (big-Oh) on the worst-case

runtime of your pseudo-code.

answer:

The iterative version of IS-TERNARY-HEAP:  
  
def is\_ternary\_heap(A):  
    for i from 0 to (len(A) - 2) // 3:  
        if A[i] < A[3i+1]:  
            return False  
        if 3i+2 < len(A) and A[i] < A[3i+2]:  
            return False  
        if 3i+3 < len(A) and A[i] < A[3i+3]:  
            return False  
    return True  
  
Runtime analysis: The loop goes through 1/3 of the list and does constant work for each element, so the worst-case runtime is in O(n).