CS5800: Algorithms — Spring '21 — Virgil Pavlu

Homework 2
Submit via Gradescope

Name: Ketaki Kolhatkar
Collaborators: Madhusan Malhar Deshpande, Ashwini Kumar

Instructions:

- Make sure to put your name on the first page. If you are using the LaTeX template we provided, then you can make sure it appears by filling in the `yourname` command.
- Please review the grading policy outlined in the course information page.
- You must also write down with whom you worked on the assignment. If this changes from problem to problem, then you should write down this information separately with each problem.
- Problem numbers (like Exercise 3.1-1) are corresponding to CLRS 3^{rd} edition. While the 2^{nd} edition has similar problems with similar numbers, the actual exercises and their solutions are different, so make sure you are using the 3^{rd} edition.

1. ***(Extra Credit)*** *Implement MergeSort without recursive calls.*

Solution:

2. ***(5 points)*** *Exercise 6.1-4, explain why.*

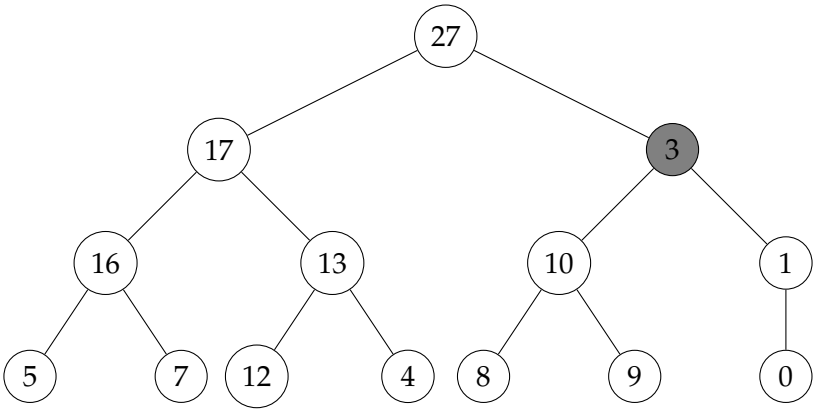
Solution:
The element in the max heap can reside anywhere in the leaf nodes. Max heap only requires the parent node of subtrees to be greater than that of the child nodes, it does not matter which side contains the larger value in the child node.

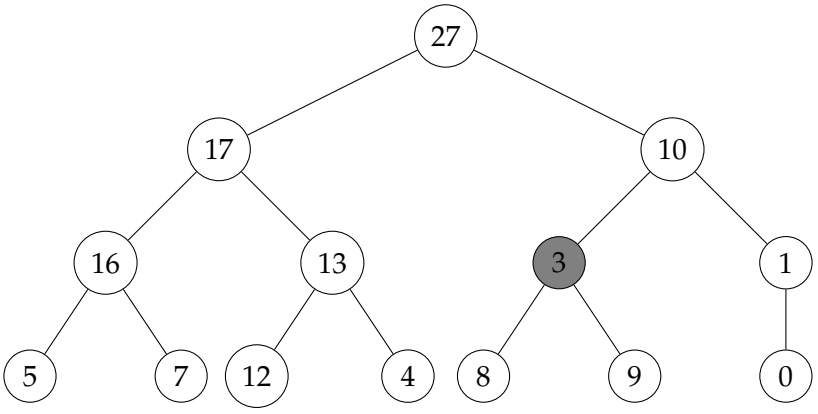
3. ***(5 points)*** *Exercise 6.1-6, explain why.*

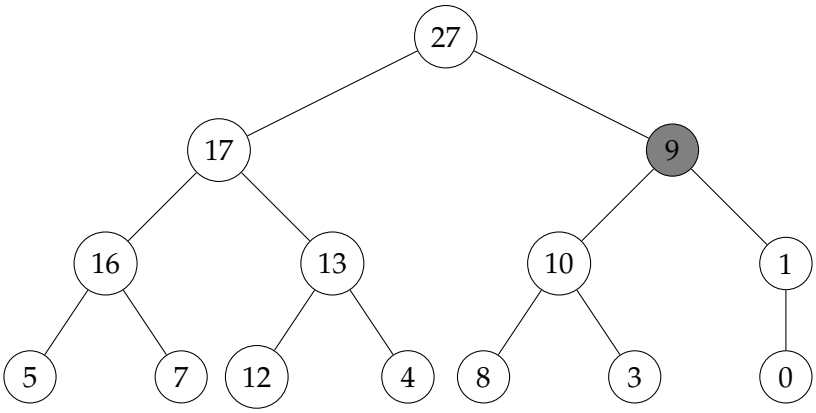
Solution:
The array is not a max heap. This is because the definition of a max heap suggests that the root node must contain a value greater than that of the subsequent child nodes. This is true recursively for all the child nodes in the heap. In the given example, one of the parent node is lesser in value as compared to the parent node, The node with the value 6 is the parent node of nodes with value 5 and 7. Hence the heap is not a max heap.

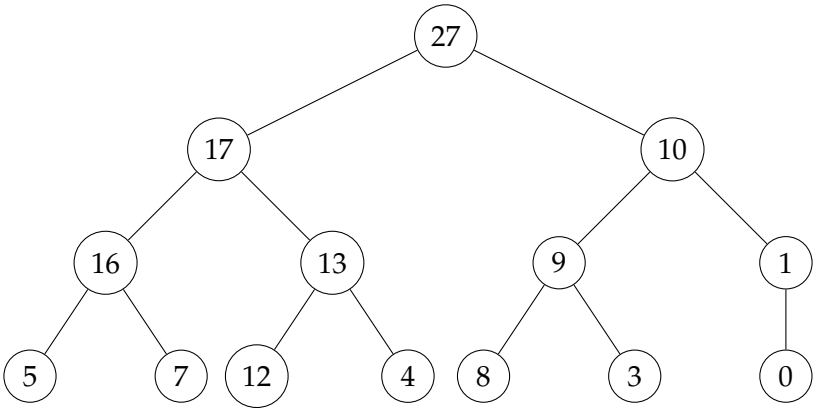
4. ***(10 points)*** *Exercise 6.2-1.*

Solution:



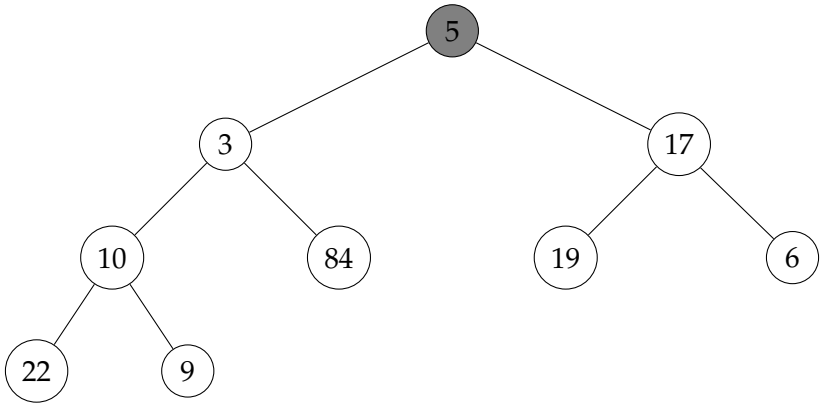


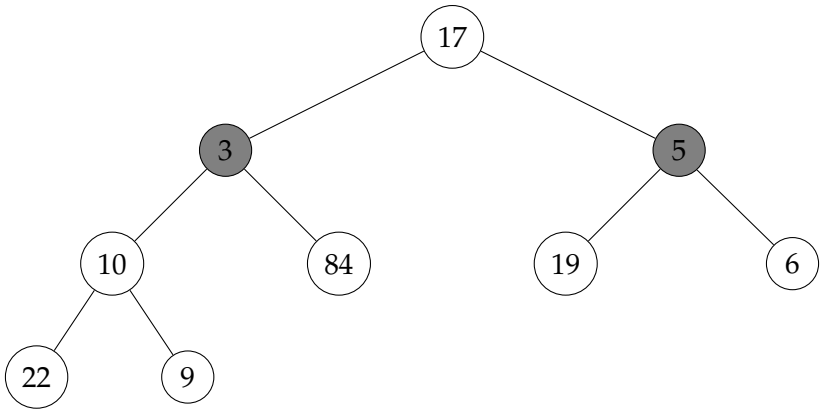


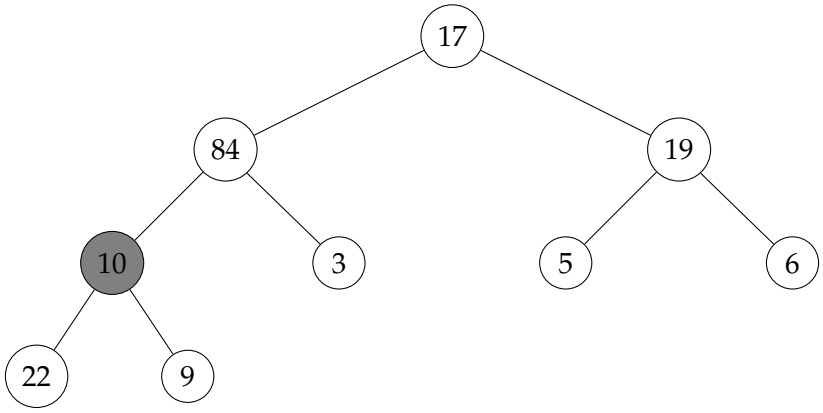


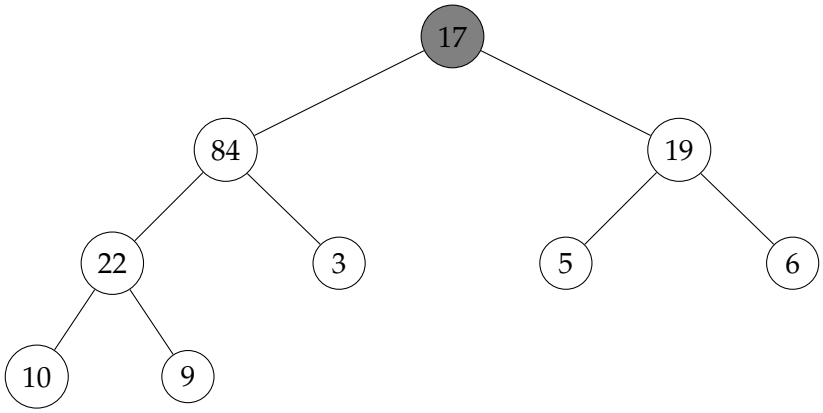
5. ***(10 points)*** *Exercise 6.3-1.*

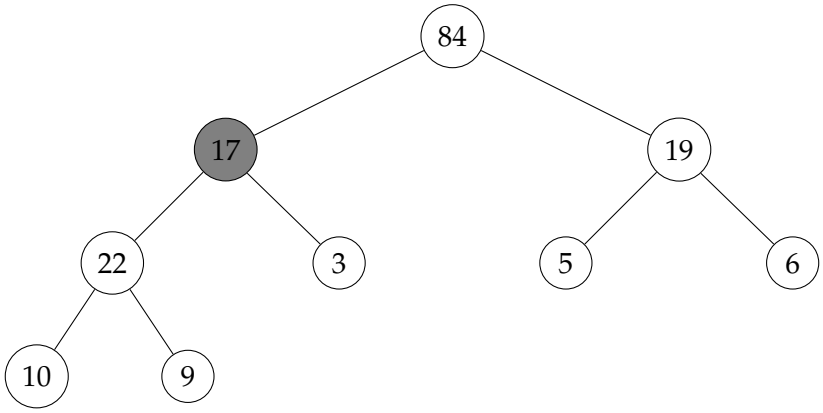
Solution:

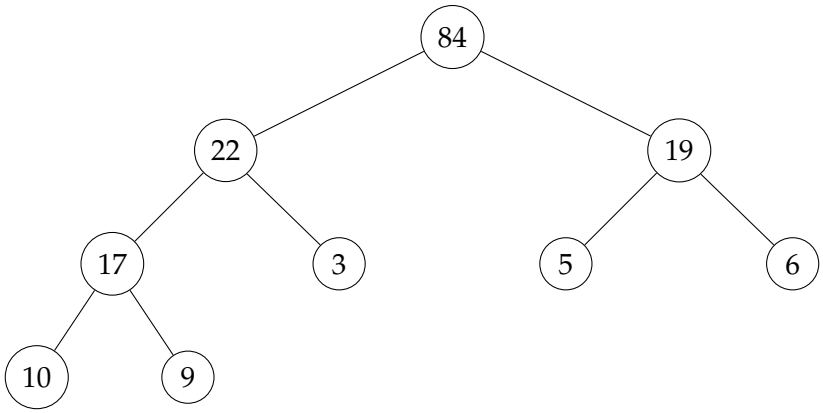



17
3
5
10
84
19
6
22
9


17
84
19
10
3
5
6
22
9


17
84
19
22
3
5
6
10
9


84
17
19
22
3
5
6
10
9



6. ***(15 points)*** *Problem 6-2.*

Solution:

(a) The elements of a d-array are represented as = i/d, where i is the total number of elements and d is the number of child nodes.

(b) The height of the d-array would be $log_d n$ since each node is branching out into d child nodes.

(c)

The d-array heap EXTRACT MAX function would be the similar to that of a binary tree, but the MAX HEAPIFY function would a little different.

```
HEAP EXTRACT MAX(A)
if heap.size < 1
        error ”heap underflow”
max = A[1]
A[1] = A[A.heap-size]
A.heap-size = A.heap-size - 1
MAX-HEAPIFY(A, 1)
return max
```

```
    MAX-HEAPIFY(A, n, d)
root = n
for k = 1 to d
   if CHILD(k,n) < A.heap-size and A[CHILD(k,n)] ≥ A[root]
            if root ! = n
                swap A[root] and A[n]
                MAX-HEAPIFY(A, root, n)
```

The runtime of this algorithm would be = $log_d n$ since the for loop would run through the entire depth of heap.

(d) MAX-HEAP-INSERT(A, key)

```
A.heap-size = A.heap-size + 1
```

A.heap-size = -∞
HEAP-INCREASE-KEY(A, A.heap-size, key)

HEAP-INCREASE-KEY(A, i, key)
if key < A[i]
error "new key is smaller than current key"
A[i] = key
while i> 1 and A[PARENT] < A[i]
exchange A[i] with A[PARENT(i)]
i = PARENT(i)

The runtime of this algorithm would be = $log_d n$ since the while loop would run through the entire depth of heap.

(e) HEAP-INCREASE-KEY(A, i, key)
if key < A[i]
error "new key is smaller than current key"
A[i] = key
while i> 1 and A[PARENT] < A[i]
exchange A[i] with A[PARENT(i)]
i = PARENT(i)
The runtime of this algorithm would be = $log_d n$ since the while loop would run through the entire depth of heap.

7. ***(5 points)*** *Exercise 7.2-1.*

Solution:

$$T(n) = T(n-2) + \theta(n) \tag{1}$$

Using the recursive method ,
$= T(n-2) + (n-1) + n$
$= T(n-3) + (n-2) + (n-1) + n$
$... = T(1) + 2 + 3 + 4 + ... + n$
$= n(n+1)/2$
$= \theta(n^2)$

8. ***(5 points)*** *Exercise 7.2-2, explain why.*

Solution:
$A = 2, 2, 2, 2, 2, 2$
Quick sort has a time complexity of $O(n^2)$ in the worst case. The partition takes a time complexity of $O(n)$. In the worst case we consider that the pivot element would be located at the extreme left or right of the list. Hence the time complexity of the recurrence would be T(n-1). There is also another time complexity of combining the last element added T(1).

$$T(n) = T(n-1) + T(1) + \theta(n) \tag{2}$$

Solving this would give us a total T(n) = n^2.

9. ***(5 points)*** *Exercise 7.2-3.*

Solution:
$A = 6, 5, 4, 3, 2, 1$
When the array is in descending order with distinct elements, it means that the least element would be at the extreme right end. As we discussed in the earlier question, whenever we have the pivot at the extreme end that would be considered as the worst case condition. The time complexity of that comes out to be $T(n^2)$. When the pivot is in the middle, that is the best time complexity condition $T(n) = \theta(nlgn)$. But in this situation, the time complexity would be the same as the worst case condition.

10. ***(15 points)*** *Exercise 7.4-1.*

Solution:
T(n) = max (T(q) + T(n-q-1)) + $\theta(n)$ We take T(n) $\geq cn^2$,
Hence, solving the equation we get,
$= cq^2 + c(n-q-1)^2 + \theta(n)$
$= q^2 + ((n-q-1)^2 + kn$
$= q^2 + ((n-1)-q)^2 + kn$
$= q^2 + ((n-1)-q)^2 + 2q(n-1-q) - 2q(n-1-q) + kn$
$= (q+n-1-q)^2 - 2q(n-1-q) + kn$
$= (n-1)^2 + kn - [2q(n-1-q)]$
Hence, we would want the second term to be non negative which will if it is ≥ 0.
$2q(n-1-q) \geq 0$
Hence we get 2 values of q = 0, q = $n-1$. On these terms we will get the maximum value of T(n).

11. ***(15 points)*** *Exercise 7.4-2.*

Solution:
Quick sort has the best case scenario when the pivot lies in the middle having two partitions : $n/2$ and $n/2-1$. Therefore, we take $2.T(n/2)$ time for the 2 partitions and $\theta(n)$ for getting the partition.

$$T(n) = T(n/2) + T(n/2-1) + \theta(n) = 2T(n) + \theta(n) \tag{3}$$

$T(n) = 2T(n/2) + \theta(n) \geq c(nlgn)$
$= 2c(n/2logn/2) + kn \geq cnlgn$
$kn \leq cnlgn - cnlgn/2$
$kn \geq cnlg(n/(n/2))$
$c = k$

12. ***(10 points)*** *Exercise 7.4-3.*

Solution:
$q^2 + ((n-q-1)^2$

$= q^2 + ((n-1) - q)^2$
$= q^2 + ((n-1) - q)^2 + 2q(n-1-q) - 2q(n-1-q)$
$= (q + n - 1 - q)^2 - 2q(n-1-q)$
$= (n-1)^2 - [2q(n-1-q)]$

From this equation we infer that the term 2q(n-1-q) should be equal to 0. This gives us 2 values of q, q = 0 and q = n-1.

13. ***(Extra credit)*** *Problem 6-3.*

Solution: