WA2 - Assignment Group 16

Sravani Kirla - skirla
Matthew Heil - heilx069
Terrence Hanrahan - hanra028

1.

```
def ArraySum(A,x):
    merge_sort(A,0,len(A)-1)      # Array in ascending sorted order        Θ(n lgn)
    p = A[0]                      # Smallest value in A                      Θ(1)
    r = A[len(A) - 1]             # Largest value in A                       Θ(1)
    if (p + r) < x:                                                          Θ(1)
        for z in range(p,r):      # Search for remaining element sums        Θ(n)
            if (z + r) == x:      # This has it run in                       Θ(n)
                return True                                                  Θ(1)
        return False                                                         Θ(1)
    elif (p + r) > x:             # Search for remaining element sums        Θ(1)
        for z in range(p+1,r):                                               Θ(n)
            if (z + p) == x:                                                 Θ(n)
                return True                                                  Θ(1)
        return False                                                         Θ(1)
    elif (p + r) == x:                                                       Θ(1)
        return True                                                          Θ(1)
    else:                                                                    Θ(1)
        return False                                                         Θ(1)
```

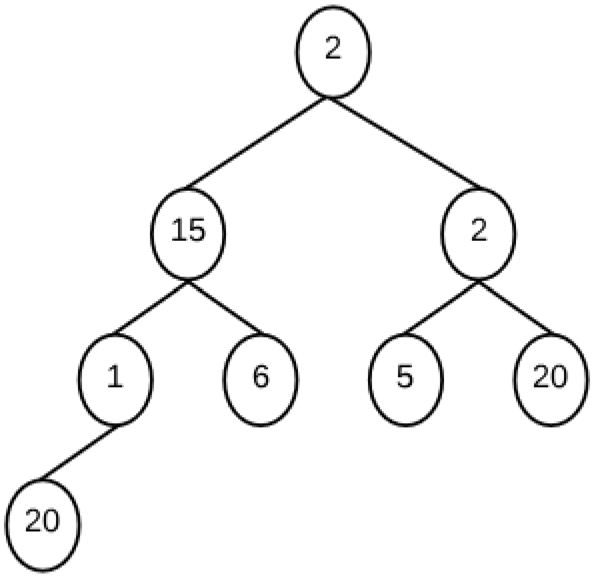
Even though there are for loops in the algorithm that run in Θ(n) time, the run-time is less than Θ(n lgn). This is because (n lgn) will always be larger than n for all n > 2. If there is a single element in the array, then False is returned automatically and the run-time is Θ(n lg n) [due to merge sort]. If there are only two elements in the array, it also runs in Θ(n lg n) time since the for loops are finished immediately. For arrays of 3 elements or larger, the algorithm runs in Θ(n lg n) time again since the loops runs in Θ(n) time. Additionally, only one of the for loops is ran before the algorithm determines if there are two elements in A that sum to X.

2. Initially, quicksort chooses the smallest element as the pivot, causing the partition split the array into two subarrays of size 0 and n-1. This means that the quicksort algorithm has to search through all elements (n-1) in the subarray, only to end up swapping the smallest element and largest element. The next recursive call will end up using the largest element in the array as the pivot, forming two subarrays with size 0 and n-2. Once again, quicksort searches through all elements (n-2) in the subarray, only to swap the largest element with the second largest element. This keeps repeating through each iteration of quicksort (swapping the n largest element with the n-1 largest element, after the first iteration where the pivot is the smallest element). This ends up causing quicksort to achieve its worst-case run-time of Θ(n^2) as is each recursive iteration it searches through the whole array.

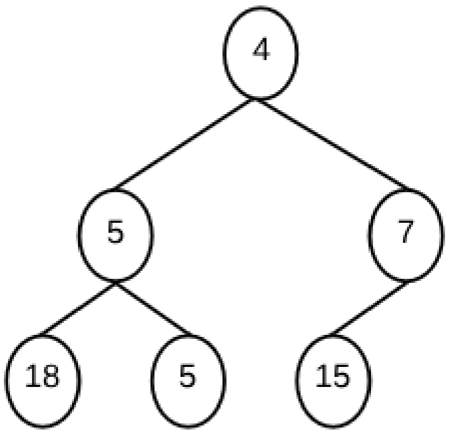
3.

WA2 - Assignment Group 16

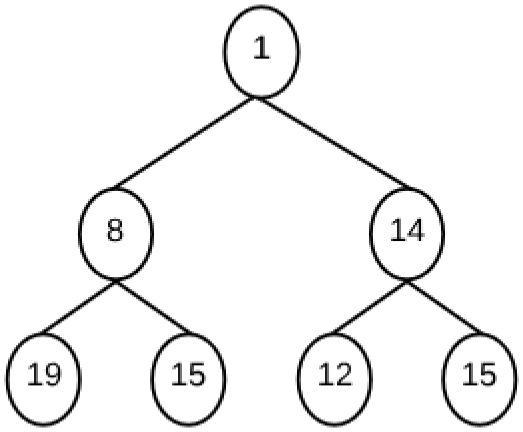
a. This is not a min-heap as 15 violates the min-heap property of the heap having large number than it's right and left child.



b. Yes, this is a min-heap, all nodes are lesser or equal to its children.

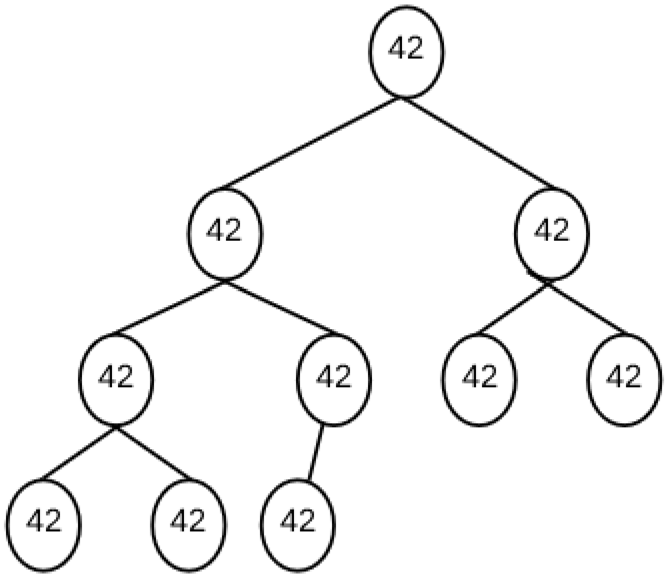


c. This is not a min-heap, as 14 violates the min-heap property having the larger number



than it's left child i.e., 12.

d. This is a min-heap as all elements are equal and satisfies min-heap property. Because by definition of Min-heap all nodes are lesser or equal in value than it's child nodes.


42
42
42
42
42
42
42
42
42
42