CST 370
Ivan Alejandre, ID: 8889
2/3/2020

Week 5

1.a) a = 2, b = 2, d = 4. Use case 1 as 2 < 16. **Thus, $T(n) = \Theta(n^4)$**

1.b) a = 16, b = 4, d = 2. Use case 2 as 16 = 16. Thus, **$T(n) = \Theta(n^{2\,*} \log n)$**

2) Mark array:

1	2	3	4	5	6	7	8
A	B	C	D	G	H	E	F

Depth-First Tree with only edges:

$$\begin{tikzcd}
A \arrow[d] & \\
B \arrow[d] \arrow[dr] & \\
C \arrow[d] & E \arrow[d] \\
D \arrow[d] & F \\
G \arrow[d] & \\
H &
\end{tikzcd}$$

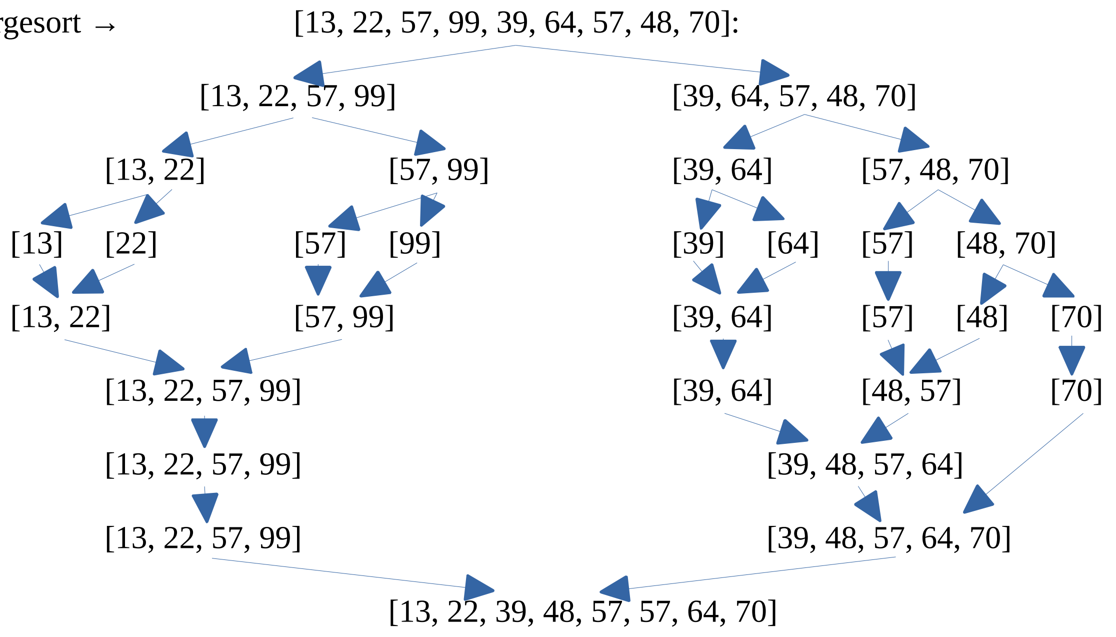
3.a) We'll take the general idea of divide and conquer algorithm presented to us in the google doc and make some slight modifications. Instead of returning the sum of the two numbers, we'll instead return the index where the largest number resides. So in each subcall, we'll be returning an index of the number that is larger. Within the function we'll compare the two divisions' returned index *values* and return the *index* of the larger value.

3.b) Algorithm ReturnIndexOfLargestNum(A[0..n-1], start, end)

```
    // returns index of largest number in array
    // Input: A[] is array of integers, start is starting index, end is n-1
    // Output: Index of largest number
    start ← 0
    end ← n - 1
    if n = 1
        return n-1
    else
        num1 = ReturnIndexOfLargestNum(A[], start, (start + end)/2)
        num2 =  ReturnIndexOfLargestNum(A[], (start + end)/2 + 1, end)
        if A[num1] ≥ A[num2]
            return num1
        else
            return num2
```

3.c) T(n) = 2 * T($^{n}/_{2}$) + n where a = 2, b = 2, and d = 1. d is equal to 1 because of the comparison operation which occurs n/2 + 1 times. Thus, applying the Master Theorem, we use case two and our time complexity is **T(n) = $\Theta(n^{*}\log n)$**

4) Mergesort →



5)

```
 p i             j
[6 10 13 5 8 3 2 11]
 p i           j
[6 10 13 5 8 3 2 11]
 p i          j
[6 2 13 5 8 3 10 11]
 p   i      j
[6 2 13 5 8 3 10 11]
 p   i     j
[6 2 3 5 8 13 10 11]
 p     j i
[6 2 3 5 8 13 10 11]
 j     p
[5 2 3 6 8 13 10 11]
```

First partition: [5 2 3]