Homework 4

1. Master Theorem
    a. T(n) = 5T(n/3) + n

a = 5, b = 3, d = 1,   b^ d = 3 < 5 = a

since b^d is less than a, this is recursion dominant

therefore Θ (n^(log3 (n)))

    b. T(n) = 2.7T(n/5) + n^2

a = 2.7, b = 5, d = 2,   b^d = 25 > 2.7 = a

since b^d is greater than a, this is overhead dominate

therefore Θ (n^2)

    c. T(n) = 2T(n-1) + n

There is no b in this case and therefore we cannot use the master theorem

    d. T(n) = 1.1 T(0.2n) + 1 = 1.1 T(n/5) +1

a = 1.1, b = 5, d = 0   b^d = 1 < 1.1 = a

since b^d is less than a, this is recursion dominant

therefore Θ (n^(log5 (n)))

    e. T(n) = 2T(n/2) + nlogn

a = 2, b = 2, c = log 2 (2) = 1

n^1 log n == f(n) meaning the MT is balanced therefore Θ (n log(n))

f. T(n) = 2T(n/2) + n^1/2

a = 2, b = 2, d = ½ b^d ~= 1.414 < 2 = a

since b^d is less than a, this is recursion dominant

therefore Θ (n^(log2 (n)))

g. T(n) = 4T(n/2) + (n^4 – n + 10)^1/2 ~= Θ (n^2)

a = 4, b = 2, d = 2 b^d = 4 == 4 = a

since b^d is equal to a, this is balanced

therefore Θ (n^(log2 (n)) log 2 (n))

h. T(n) = 7T(n/3) + $\sum_{i=1}^{n} i$ = 7 T(n/3) + (n^2 +n)/2

a = 7, b = 3, d = 2 b^d = 9 > 7 = a

since b^d is greater than a, this is overhead dominate

therefore Θ (n^2)

i. T(n) = 4T(n/2) + n^n

Not in right format to get d, though n^n is one of the worst complexities so will assume that is the dominant one.

j. T(n) = 8T(n/3) + n^3

a = 8, b = 3, d = 3 b^d = 27 > 8 = a

since b^d is greater than a, this is overhead dominate

therefore Θ (n^3)

2. D&C Master Theorem

a. Algo 1 T(n) = 1 T(n/3) + 1

a = 1, b = 3, d = 0 b^d = 1 == 1 = a

Worst case would be if there were no negative numbers in the list as it looks as though the algorithm looks for the first negative number?

Θ (n^(log3 (n)) log 3 (n))

Best case the first element is negative as it makes the recursive call at the front of the list first

b. Algo 2 T(n) = 2 T(n/2) + 1

a = 2, b = 2, d = 0 b^d = 1 < 2 = a so this is recursive dominant

therefore Θ (n^(log2 (n)))

Wost case would be same as the best case as this simply looks like it sums all the elements in a list

3. Merge Sort

a. Design k way merge sort

```
MergeSortK(A[0, n-1]):
    If n > 1:  # still the same since n is multiple of K
        i = 0
        while i < n+1: # will be n in the last one
                currentStart = i
                i += k
                Merge(MergeSortK(A[0, i-1])
                # should end at i = n
    Else:
        Return A
```

b. Worst Case

$T(n) = k\ T(n/k) + \Theta\ (n)$

$a = k, b = k, d = 1$ $b^{\wedge} d = k == k = a$

therefore the equation is balanced at Θ (n)

c. Better or Worse

Better, in class we discussed that the recursion by 2 is Θ (n log n) and Θ (n) is better than that.

We would pick a value of k depending on the number of pieces we want, this skips the number of recursions for later?

This seems wrong but I am not sure what about it is off.

d. Does it run in Θ (n)

Yes it is reasonable since the master theorem showed that T(n) is balanced.

4. Consider Algorithm

The algorithm as written will always return 0 as it adds up a number of zeros equal to the number of nodes (including root)

To correct this we could check if left + right = 0 then we would know this is a leaf and could return 1

If T = null return 0

Else If (LeafCounter(Tleft) + LeafCounter(Tright) == 0

Return 1

Else return LeafCounter(Tleft) + LeafCounter(Tright)

Now we have check if this node is null, if this node has children (if not it is a leaf, and if not a leaf, call further down to see where the leaves are.

5. Travers Tree

a. Pre

A, B, D, E, C, F

b. In

D, B, E, A, C, F

c. Post

D, E, B, F, C, A

6. Karatsuba Algorithm (2101 * 1130)

10^4 axc + 10^2 (ad + bc) + bd   a = 21, b = 01, c = 11, d = 30

21x11 10^4 + (21x 30 + 1x11) 10^2 + 1x30

231 10^4 + (630 + 11) 10^2 + 30

2,310,000 + 64,100 + 30

2,310,000 + 64,130

2,374,130

7. Straasen's Algorithm (left is top of matrix, right is bottom)

R = 10 41 * 01 21 + 21 10 * 20 13   s = 10 41 * 01 04 + 21 10 * 11 50

T = 01 50 * 01 21 + 30 21 * 20 13   u = 01 50 * 01 04 + 30 21 * 11 50

R = 21 25 + 53 20 = 74 45   Final Matrix: 7 4 7 3

S = 01 08 + 72 11 = 73 19   4 5 1 9

T = 21 05 + 60 53 = 81 58   8 1 3 7

U = 04 05 + 33 72 = 37 77   5 8 7 7

8. D&C 2-D Closest Pair

Mergesort = Θ (n log (n))

T(n) = 2 T(n/2) + Θ (n log (n))

a = 2, b = 2, c = log 2 (2) = 1

n log (n) lower bounded by n is true so case 3 of master theorem

Θ (n log (n))