Information you might need.

1. Master Formula

Suppose T(n) satisfied

$$T(n)=\begin{cases} d & if\ n=1 \\ aT(\left\lceil \frac{n}{b} \right\rceil)+cn^k & otherwise \end{cases}$$

Where k is none negative integer and a, b, c, d are constants with a>0, b>1, c>0, d>=0 then

$$T(n)=\begin{cases} \Theta(n^k) & if\ a<b^k \\ \Theta(n^k \log n) & if\ a=b^k \\ \Theta(n^{\log_b^a}) & if\ a>b^k \end{cases}$$

2. $x=b^y \implies Log_b x = y$

3. $\sum_{i=0}^{n-1} i=\frac{n(n-1)}{2}$

Q1) (14 points) Complexity Analysis

1. [4 points] What is the worst case running time in Big-Oh notation for the following functions? Show your work for partial credit.

a.
```
void foo(int n )
   {
      int count = 0;
      for (int i = 0; i < n; i++)          n
        for (int j = i; j^2 > 0; j--)      n-1*n
            count = count + 1;             n-1 *n
     return count;
   }
```

n=1, count=1
n=2, count=3
n=3, count=9

n+(n-1)*n+(n-1)*n
2(n-1)*n + n
2n^2 - 2n + n
O(n^2)

b.
```
void foo(int n)
   {
      m = 0;
      while(n>=2)              n/3
       {
          n = n/3;             2*n/3
          m = m+1;             2*n/3
          System.out.println(m);     1 *n/3
       }
      return m
   }
```

1/3 =1/3^1
(1/3)/3 =1/3^2
((1/3)/3)/3 =1/3^3
...
1/3^k =1/3^k

n <= 2

n=1/3^k
(3^k)=1/n
k=log3 n
: O(log n)

2. [4 points] Discuss whether the next statements are true for the given function $f(n) = 2n^2 + n\log n$ O(n^2)

a) $f(n) = O((n^2 \log n)$ Yes: Lim (2n^2+nlog n)/(n^2*log n) --> 0

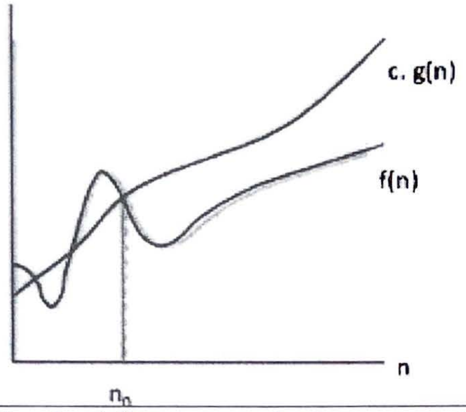
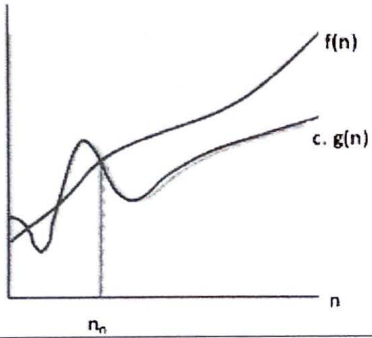
Lim (2n^2/n^2log n) + (nlog n/n^2log n)
Lim (1/log n) + (1/n) --> 0

b) $f(n) = \Theta(n^2 \log n)$ We have to check Omega:
if Lim (n^2*log n)/(2n^2+nlog n) is finite

Lim (n^2 log n)/(n*(2n+log n))
Lim (nlog n)/(2n+log n) is not finite.
:So, F(n) is not Theta of n^2 log n

3. [6 points] Multiple Choice Questions:

1.	What is the tightest asymptotic bound for the following f(n) $f(n) = 2n^n + 2^{100}$	✓a. $O(n^n)$ <-- This b. $O(2^{100})$ c. $O(2n^n)$
2.	What is the tightest asymptotic bound for the following f(n) $f(n) = n^2 \log(n^5) + 5n \log(n^5) + n^3$	a. $O(n^2 \log(n^5))$ b. $O(n^2 \log n)$ ✓c. $O(n^3)$ <--- This d. $O(5n \log(n^5))$

3.	For the following graph, the asymptotic relationship of f and g functions is 	a. f(n) = Theta (g(n)) b. f(n) = O (g(n)) <-- This c. f(n) = Big-Omega (g(n)) Fasl
4.	For the following graph, the asymptotic relationship of f and g functions is 	a. f(n) = Theta (g(n)) b. f(n) = O (g(n)) c. f(n) = Big-Omega (g(n)) <-- This
5.	Which of the following is not $O(n^3)$	A. $N^{2.98}$ B. $15^{10}\, n + 1000$ C. $n^4 / n^{1/2}$ <-- This D. $2^{20}\, n^2$
6.	Which of the given options provide the increasing order of asymptotic complexity of functions f1, f2, f3 and f4? $f1(n) = 2n$ $f2(n) = n!$ $f3(n) = n \log n$ $f4(n) = 7^n$	a. f3, f2, f4, f1 b. f3, f2, f1, f4 c. f2, f3, f1, f4 d. f1, f3, f4, f2 <-- This

Q2) (11 points) Analysis

1. [4 points] Give tight asymptotic bound for the following recurrences:
   a. $T(n) = 4T(n/2) + 1$

      k=0
      c=1
      a=4
      b=2
      d=0
      a > b^k => Theta(n^log 4)

   b. $T(n) = T(n/2) + n^3$

      a=1
      b=2
      c=1
      d=0
      k=3

      As 1 < 2^3 => Theta(n^3)

2. [5 points] Given an array A with n integer elements. Write an algorithm to print the largest number and the smallest number in the array. Propose an algorithm to solve this problem in O(n) worst-case time or better.
   Example: A = {1, 3, 2, 1, 3, 5, 3} the result should be the largest number is 5 and the smallest number is 1.

```
public void search1(int[] input) {
    int maxValue = Integer.MIN_VALUE;
    int minValue = Integer.MAX_VALUE;
    for (int i=0; i < input.length; i++) {
        if(input[i] < minValue)
            minValue = input[i];

        if(input[i] > maxValue)
            maxValue = input[i];
    }
    System.out.println("maxValue: " + maxValue);
    System.out.println("minValue: " + minValue);
}
```

We need to transform this in pseudo-code sintax

A <-- BucketSort(A)
minValue <-- A[0]
maxValue <-- A[A.lenght-1]

Can use BucketSort that is O(n) because range is small: numbers < 10, if range is big can use RadixSort instead.

3. [2 points] Verify whether the following statement is true or false. The amortized cost of a sequence of n operations from S is O(n), and so the amortized cost of a single operation from S is O(n).
Given S consists of two operations:
   a. add(x) = inserts String x into next available slot
   b. clear() = replaces all Strings in array with nulls

False: the cost of a single operation from S would be O(n)/n=O(1)

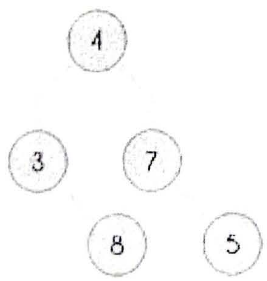
Conclusion. Therefore, the amortized cost of a sequence of n operatons from S is $O(2n) = O(n)$, and so the amortized cost of a single operation from S is $O(n)/n = O(1)$.

Q3) (14 points) Sorting algorithms:

1. [5 points] Answer the following questions by True or False
   × a. It is possible to develop a comparison based sorting algorithm that runs in Θ (n). False
   × b. The best time complexity of Bubble Sort is O(n log n). False
   × c. Merge Sort makes more swap operations than Selection-Sort. Check
   × d. Suppose we have a O(n) time algorithm that finds median of an unsorted array. Consider a QuickSort implementation where we first find median using the above algorithm, then use median as pivot. The worst case time complexity of this modified QuickSort is O(n) False: O(n*log n)
   ✓ e. Insertion-Sort is stable sorting algorithm. Yes

A	B	C	D	E

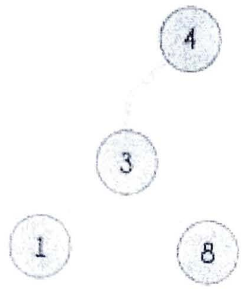
2. [3 points] Which of the following trees is a heap? Explain your answer for each tree.



False, a heap must fill its last level from left to rigth

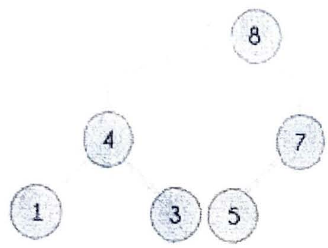
a.

b.



False, a heap must be a complete tree, the second level is not full (is missing a node)

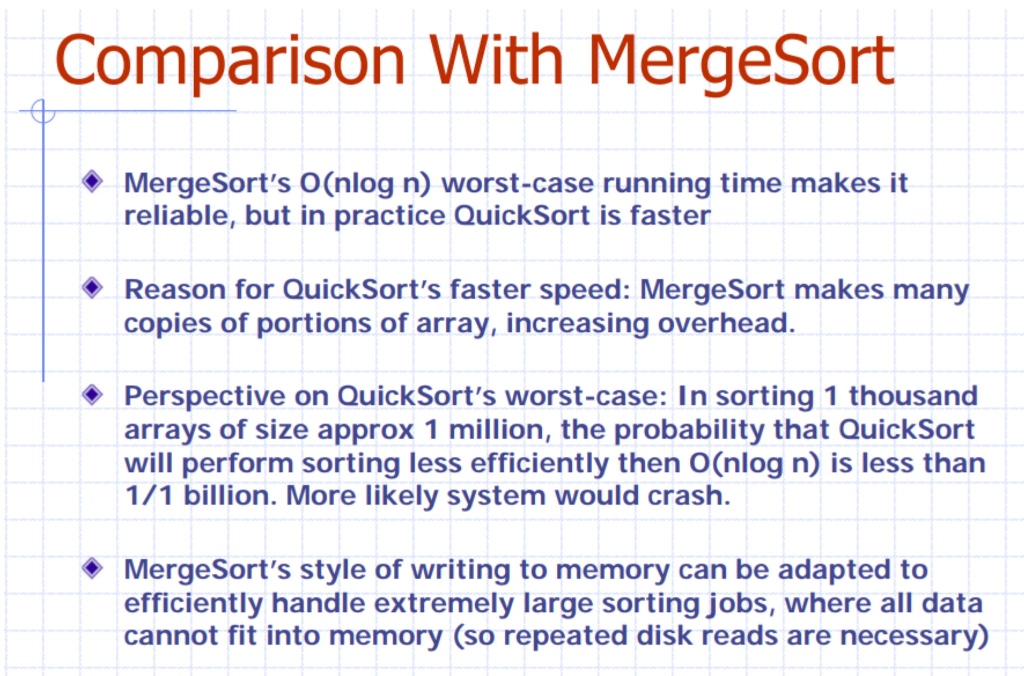
c.



True

According to my understanding this is a Heap Tree, because this is a Max-Heap. which means the root is the max number of all nodes.

3. [2 points] Given that the running time worst case for merge-sort is better than quicksort, why quick-sort is commonly used?



4. [4 points] Use Merge Sort Algorithm to sort the following array of integers. Show the merge-sort tree.

30 25 10 80 20 15 99 88

30 25 10 80 | 20 15 99 88

10 20 25 30 80 88 99

30 25 10 80   20 15 99 88

10 25 30 80   15-20-88-99

30 25   10 80   20 15   99 88

25-30   15-20   88-99

$\boxed{30}$ $\boxed{25}$ $\boxed{10}$ $\boxed{80}$ $\boxed{20}$ $\boxed{15}$ $\boxed{99}$ $\boxed{88}$