Ethan Tuning

10/28/1026

CSCD300

Homework 4

To Turn in: please submit the questions and your answers below them in a pdf file on canvas.

Perform a time-complexity (Big-O) analysis for each of the next three problems (problems 1, 2, and 3). For full credit you should be able to produce a logical justification for your answer (a growth rate function can help demonstrate this – but is NOT required – so at least show in general why the Big-O is what it is). Equations you may need: (1) $1 + 2 + 3 + 4 + \ldots + n = (1 + n) * n / 2$; (2) $1 + a + a^2 + a^3 + \ldots + a^n = (a^{n+1} – 1) / (a-1)$.

1. (40 Points)

```
    public static void two(int n)
    {
       if(n > 0)
       {
          System.out.println("n: " +n);
          two(n - 1);
          two(n - 1);
       }
       else if (n < 0)
        {
           two(n + 1);
           two(n + 1);
           System.out.println("n: " + n);
        }
    }
```

Within the first if() statement, the recursive calls will happen 2^n. The else-if() will do the exact same thing as the if() as long as n is greater than 0. So the total time complexity will be $O(2^n)$

2. (30 Points)

```
public void three(int n)
{
  int i, j, k;
  for (i = n/2; i > 0; i = i/2)
      for (j = 0; j < n; j++)
            for (k = 0; k < n; k++)
                  System.out.println("i: " + i + " j: " + j+" k: " + k);

}  // end three
```

first for() log(n) times, second for() runs n times, and the third for() runs n times as well. So the total time complexity will be $O(n^2log(n))$

3. (30 points)

```
public static void four(int n)
{
   if (n > 1)
   {
      System.out.println(n);
      four(n-1);
   }
   for (int i = 0; i < n; i++)
     System.out.println(i);
}
```

The if() will perform n times, as well as the recursive call, then the for() will perform once the recursion hits the base case. The for() will not perform at all because after the if() is done n will always be 0. So we are looking at a total time complexity of $0(2^n)$.