CS 371 HW 1

Josh Blaz

1/29/19

1. Chapter 2 Problem 4 (ascending order):
   (a) $g_1(n) = 2^{\sqrt{logn}}$
   (b) $g_3(n) = n(logn)^3$
   (c) $g_4(n) = n^{4/3}$
   (d) $g_5(n) = n^{logn}$
   (e) $g_2(n) = 2^n$
   (f) $g_7(n) = 2^{n^2}$
   (g) $g_6(n) = 2^{2^n}$

2. Chapter 2 Problem 6:
   (a) The first line of the algorithm (the outer loop) will always run n times. The second line of the algorithm will run at most n times. In the worst case scenario, where $i = 1$ and $j = n$, there will be n operations inside the nested for loop (where the algorithm calculates the sum of array integers $A[i]..A[j]$). This means that the worst case run time for this algorithm is $O(n^3)$, because the outer loop always runs n times, the inner loop runs at most n times, and there are at most n operations performed within the nested for loop.
   (b) The first line of the algorithm (the outer loop) will always run n times. On average, the inner loop will run $\frac{n}{2}$ times, meaning there will be (on average) another $\frac{n}{2}$ operations within the inner loop (where the algorithm calculates the sum of the array integers). Which would give the algorithm $\Omega(n^3)$ Given that the algorithm runs in both $\Theta(n^3)$ and $O(n^3)$ time, it must also be that the algorithm runs in $\Omega(n^3)$ time.
   (c)

```
For $i = 1, 2, ..., n$ $\leftarrow$ Initialize B matrix, nested for loop:$n^2$
    For $j = i + 1, i + 2, ..., n$
        $B[i][j] = 0$
For $i = 1, 2, ..., n$ $\leftarrow$ Set top row of B matrix, nested for loop:$n^2$
    For $j = i + 1, i + 2, ..., n$
        $B[1][i] += A[j]$
```

```
For  $i = 1, 2, ..., n$
    For  $j = i + 1, i + 2, ..., n$
        $B[i][j] = B[i-1][j] - A[i-1]$
```

Rather than adding up the entries "The Natural Way" as shown in the book, this algorithm bypasses the need to iterate through the entries by taking the difference between $B[i-1][j]$ and $A[i-1]$. This algorithm works because the difference between these two values is equal to the sum of entries $A[i]..A[j]$, by calculating the sum of $A[i]..A[j]$ this way, the algorithm goes from $O(n^3)$ to $O(n^2)$.

3. Chapter 2 Problem 8a :
Suppose that $n = 16$. A strategy for testing the jars would be to drop a jar at every $\sqrt{n}$ rungs, in this case, every 4th rung. If a dropped jar broke from one of these rungs, then I would proceed by dropping another jar at the median between the current rung (at which a dropped jar had broken) and the last safe rung. If the jar dropped at the median broke, then I would consider the last safe rung to be the highest safe rung. If the jar dropped at the median did not break, then I would drop the jar at the median between the last unsafe rung and the median, if the jar broke, then I would still consider the median to be the highest safe rung, otherwise I would consider the rung of that final drop (between the median and the last breaking rung) to be the highest safe rung.