Homework 1

Joseph Sepich

Contents

1 Problem 1 **1**

1.1 a. 1

1.2 b. 4

2 Problem 2 **4**

1 Problem 1

1.1 a.

Functions as given:

1. $log^2(n)$
2. $\binom{n}{2}$
3. $log(n^2)$
4. $log(n!)$
5. $2^{log(n)}$
6. $n * log(n)$
7. $4^{log(n)}$
8. $\sqrt{n}$
9. $2^{log^2(n)}$
10. n
11. $log(log(n))$

Recall the definition of big O notation:

c > 0, $n_0 > 0$ and $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$

We will proceed by comparing each function to the previous and adjusting the order accordingly.

Recall that $\binom{n}{2} = \frac{n!}{2(n-2)!}$

With $n_0 = 1$ and c = 1

$$log^2(n) \leq \binom{n}{2}$$

So we start with:

1. $log^2(n)$
2. $\binom{n}{2}$

Now compare function 2. and 3.

Here we can choose $n_0 = 2$ and $c = 2$

$$log(n^2) \leq \frac{2n!}{2(n-2)!} = n(n-1)$$

So compare 3. and 1.

Here we can choose $n_0 = 2$ and $c = 2$

$$log(n^2) \leq 2log^2(n)$$

So we have:

1. $log(n^2)$
2. $log^2(n)$
3. $\binom{n}{2}$

Now compare 3. and 4.

Here we can choose $n_0 = 2$ and $c = 1$

$$log(n!) \leq \binom{n}{2}$$

And compare to $log^2(n)$

We can choose $n_0 = 1$ and $c = 1/2$

$$log^2(n) \leq 2log(n!)$$

So we have:

1. $log(n^2)$
2. $log^2(n)$
3. $log(n!)$
4. $\binom{n}{2}$

Next looking at $2^{log(n)}$ we know equal x by logarithm properties, so we can put it between the two functions that increase at an increasing rate and increase at a decreasing rate and have:

1. $log(n^2)$
2. $log^2(n)$
3. $2^{log(n)}$
4. $log(n!)$
5. $\binom{n}{2}$

Next we look at $n * log(n)$

This function increases at an increasing rate, so we know it is above $2^{log(n)}$ then let's compare to the next highest $log(n!)$

Choosing $n_0 = 1$ and $c = 1$ we get

$$n * log(n) \leq log(n!)$$

So we have:

1. $log(n^2)$
2. $log^2(n)$
3. $2^{log(n)}$
4. $n * log(n)$
5. $log(n!)$
6. $\binom{n}{2}$

Next we look at $4^{log(n)}$. This is similar to the exponential function 4^n, so compare it to our fastest growing function:

Choosing $n_0 = 1$ and $c = 1$ we get

$$\binom{n}{2} \leq 4^{log(n)}$$

So we have:

1. $log(n^2)$
2. $log^2(n)$
3. $2^{log(n)}$
4. $n * log(n)$
5. $log(n!)$
6. $\binom{n}{2}$
7. $4^{log(n)}$

Next we look at $\sqrt{n}$, which obviously grows less than the linear function $2^{log(n)}$. Let's compare to our bottom two functions.

Choose $n_0 = 2$ and $c = 2$

$$\sqrt{n} \leq 2log^2(n)$$

and choose $n_0 = 1$ and $c = 3$

$$log(n^2) \leq 3\sqrt{n}$$

So we have:

1. $log(n^2)$
2. $\sqrt{n}$
3. $log^2(n)$
4. $2^{log(n)}$
5. $n * log(n)$
6. $log(n!)$
7. $\binom{n}{2}$
8. $4^{log(n)}$

log(log(n)) is the slowest growing with the composed log functions and n is the same function as $2^{log(n)}$.

Keeping with this process I get:

1. $log(log(n)$
2. $log(n^2)$
3. $log^2(n)$
4. $\sqrt{n}$
5. $2^{log(n)}$
6. n
7. $n * log(n)$
8. $log(n!)$
9. $\binom{n}{2}$
10. $4^{log(n)}$
11. $2^{log^2(n)}$

1.2 b.

Now we want to sort the functions into classes that are upper and lower bounded by each other. Basically can you find an n_0 and c for each other that it is upper bounded and a different one that they are lower bounded.

There are only two groups of classes that I can see functions belonging to. The others are not grouped with anyone else.

1. n and $2^{log(n)}$ grow at the exact same rate Θn.
2. $\binom{n}{2}$ and $4^{log(n)}$ are both bounded by $\Theta(n^2)$

The second group makes sense, because n choose 2 is the binomial coefficient, or the coefficient for the term x^2, so it grows at a rate bounded $\Theta(x^2)$. $4^{log(n)}$ is equivalent to x^2, so it must be bounded.

2 Problem 2

We are running the insertion sort algorithm on the following input sequence:

2,1,4,3,6,5,...,n,n-1

We can see that each pair of numbers is switch around in order i.e. 1 is where 2 should be and 2 is where 1 should be, 3 is where 4 should be and 4 is where 3 should be and so on up to n is where n-1 should be and n-1 is where n should be.

This means that for every 2 numbers we need to perform 4 actions (with exception of the first pair, since we start at the second index). This means we will perform $4 * \frac{n}{2}$ actions or 2n actions. 2n is bother lower and upper bounded by n, if you use $n_0 = 1$, c = 1, and $n_0 = 1$, c = 3 respectively. Therefore the running time of the input here is $\Theta(n)$.