Homework 1
CS 457

Damien Prieur

Question 1

Find two functions $f(n)$ and $g(n)$ that satisfy the following relationship. If no such f and g exist, then try to explain why this is the case.

a. $f(n) \in o(g(n))$ and $f(n) \notin \Theta(g(n))$
   Let $f(n) = n$ and $g(n) = n^2$

b. $f(n) \in \Theta(g(n))$ and $f(n) \in o(g(n))$
   No such functions exist as $g(n)$ cannot be strictly larger than $f(n)$ asymptotically and also bound $f(n)$ from below.

c. $f(n) \in \Theta(g(n))$ and $f(n) \notin O(g(n))$
   No such functions exist as $f(n) \in \Theta(g(n))$ is only true if $f(n) \in O(g(n))$

d. $f(n) \in \Omega(g(n))$ and $f(n) \notin O(g(n))$
   Let $f(n) = n(1 + \cos n)$ and $g(n) = 1$

Question 2

For each of the following questions, briefly explain your answer.

a. If I prove that an algorithm takes $O(n^2)$ worst-case time, is it possible that it takes $O(n)$ time on some inputs?
   Yes, your worst-case inputs does not give you an upper bound on your best case input. See insertion sort from the lecutures, it's $O(n)$ on sorted inputs and $O(n^2)$ in it's worst case.

b. If I prove that an algorithm takes $O(n^2)$ worst-case time, is it possible that it takes $O(n)$ time on all inputs?
   No because there is at least one input, your worst-case input, that will run in $O(n^2)$ time not $O(n)$.

c. If I prove that an algorithm takes $\Theta(n^2)$ worst-case time, is it possible that it takes $O(n)$ time on some inputs?
   Yes, if your best case input is $O(n)$, see insertion sort again.

d. If I prove that an algorithm takes $\Theta(n^2)$ worst-case time, is it possible that it takes $O(n)$ time on all inputs?
   No, because there is at least one input, your worst case input, that will run in at least $\Theta(n^2)$ time.

Question 3

Let $f(n)$ and $g(n)$ be any two functions with $f(n) > 1$ and $g(n) > 1$ for all n. Is it true that $15f(n)^2 + 23g(n)^2 \in \Theta((f(n)+g(n))^2)$? Give a formal justification for your answer. We want to show

$$0 \leq c_1((f(n)+g(n))^2) \leq 15f(n)^2 + 23g(n)^2 \leq c_2((f(n)+g(n))^2)$$

for some

$$c_1 > 0, c_2 > 0, \forall n > n_0$$

We can start by expanding which gives us

$$0 \leq c_1(f(n)^2 + 2g(n) \cdot f(n) + g(n)^2) \leq 15f(n)^2 + 23g(n)^2 \leq c_2(f(n)^2 + 2g(n) \cdot f(n) + g(n)^2)$$

I am going to ignore the $0 \leq$ case as it is given that both functions are greater. We must solve each inequality on it's own, we will start with the first. and divide by c_1

$$f(n)^2 + 2g(n) \cdot f(n) + g(n)^2 \leq \frac{15}{c_1} f(n)^2 + \frac{23}{c_1} g(n)^2$$

Now we can subtract the $f(n)^2$ and $g(n)^2)$ terms which gives

$$2g(n) \cdot f(n) \leq (\frac{15}{c_1} - 1)f(n)^2 + (\frac{23}{c_1} - 1)g(n)^2$$

Now we can divide by $2g(n) \cdot f(n)$ to get

$$1 \leq (\frac{15}{c_1} - 1)\frac{f(n)}{2g(n)} + (\frac{23}{c_1} - 1)\frac{g(n)}{2f(n)}$$

Now we need to know something about these function specifically what their ratio is. We can analyze this by using cases and looking at large enough n where the cases will be one of three things

1. $$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$
2. $$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$
3. $$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{a}{b}$$

   Where a and b are integers

Looking at each case we get

1. $$1 \leq (\frac{15}{c_1} - 1)\infty + (\frac{23}{c_1} - 1) \cdot 0$$
   $$1 \leq (\frac{15}{c_1} - 1)\infty$$

   Which holds as long as $(\frac{15}{c_1} - 1) > 0$ So solving for c_1

   $$c_1 < 15$$

2.

$$1 \leq (\frac{15}{c_1} - 1) \cdot 0 + (\frac{23}{c_1} - 1)\infty$$

$$1 \leq (\frac{23}{c_1} - 1)\infty$$

Which also holds as long as $(\frac{23}{c_1} - 1) > 0$ So solving for c_1

$$c_1 < 23$$

3.

$$1 \leq (\frac{15}{c_1} - 1)\frac{a}{b} + (\frac{23}{c_1} - 1)\frac{b}{a}$$

Solving for c_1 we get

$$c_1 \leq \frac{15a^2 + 23b^2}{a^2 + ab + b^2} \qquad \forall a > 0, \forall b > 0$$

Now we must look at the second inequality and follow the same steps

$$\frac{15}{c_2}f(n)^2 + \frac{23}{c_2}g(n)^2 \leq f(n)^2 + 2g(n) \cdot f(n) + g(n)^2$$

Now we can subtract the $f(n)^2$ and $g(n)^2$) terms which gives

$$(\frac{15}{c_2} - 1)f(n)^2 + (\frac{23}{c_2} - 1)g(n)^2 \leq 2g(n) \cdot f(n)$$

Now we can divide by $2g(n) \cdot f(n)$ to get

$$(\frac{15}{c_2} - 1)\frac{f(n)}{2g(n)} + (\frac{23}{c_2} - 1)\frac{g(n)}{2f(n)} \leq 1$$

Using the same three cases as before we get

1.

$$(\frac{15}{c_2} - 1)\infty + (\frac{23}{c_2} - 1) \cdot 0 \leq 1$$

$$(\frac{15}{c_2} - 1)\infty \leq 1$$

Which holds as long as $(\frac{15}{c_2} - 1) \leq 0$ So solving for c_2

$$c_2 \geq 15$$

2.

$$(\frac{15}{c_2} - 1) \cdot 0 + (\frac{23}{c_2} - 1)\infty \leq 1$$

$$(\frac{23}{c_2} - 1)\infty \leq 1$$

Which also holds as long as $(\frac{23}{c_2} - 1) \leq 0$ So solving for c_2

$$c_2 \geq 23$$

3.

$$(\frac{15}{c_2} - 1)\frac{a}{b} + (\frac{23}{c_2} - 1)\frac{b}{a} \leq 1$$

Solving for c_2 we get

$$c_2 \geq \frac{15a^2 + 23b^2}{a^2 + ab + b^2} \qquad \forall a > 0, \forall b > 0$$

Since we can always find a c_1 and c_2 our original inequality holds. So

$$15f(n)^2 + 23g(n)^2 \in \Theta((f(n) + g(n))^2)$$

$$QED$$

Question 4

Is the following statement **true** or **false**, and why?
If $f(n) \in O(g(n))$ then $2^{f(n)} \in O(2^{g(n)})$.
In order to be big-Oh we must show

$$2^{f(n)} \leq c2^{g(n)} \qquad \forall n > n_0$$

$$\log_2(2^{f(n)}) \leq \log_2(c2^{g(n)})$$

$$f(n) \leq \log_2 c + g(n)$$

$$f(n) - g(n) \leq \log_2 c$$

Since $f(n) \in O(g(n))$ we know eventually $g(n) > f(n) \quad \forall n > n_0$ This tells us that

$$f(n) - g(n) < 0 \qquad \forall n > n_0$$

Which gives us

$$f(n) - g(n) < 0 < \log_2 c \qquad \forall n > n_0$$

$$c > 0$$

$$QED$$

Question 5

What value is returned by the following function? Express your answer as a function of n. From this, deduce the worst-case running time using Big Oh notation. For full credit, provide a lower bound as well, thus leading to a bound with $\Theta(\cdot)$ notation. Then, also deduce the worst-case running time if we change the while loop clause to $j \leq i$ instead of $j \leq n$.

```
function fool(n)
r := 0
for i := 1 to n do
    j:= 1
    while j ≤ n do
        r := r + 1
        j := 2·j
return r
```

This functions returns $n * (1 + \lfloor \log_2 n \rfloor)$ which can also be written as $n * \lceil \log_2 n \rceil$ so we have:

$$fool(n) = n\lceil \log_2 n \rceil$$

To find the runtime of this we can look at each line and determine it's runtime.

1. c_1
2. $c_2 n$
3. $c_3 n$
4. $c_4 n \lceil \log_2 n \rceil$
5. $c_5 n \lceil \log_2 n \rceil$
6. $c_6 n \lceil \log_2 n \rceil$
7. c_7

Adding all these terms up we get

$$T(n) = c_1 + c_7 + n(c_2 + c_3 + \lceil \log_2 n \rceil (c_4 + c_5 + c_6))$$

$T(n)$ can be bounded above by removing the ceiling with a plus one. This would make it strictly larger.

$$T(n) = c_1 + c_7 + n(c_2 + c_3 + \log_2(2n)(c_4 + c_5 + c_6))$$

Combine constants to get

$$T(n) = C_1 + C_2 n + C_3 n \log_2(2n)$$

Which we can see is $O(n \log n)$ as well as $\Omega(n \log n)$ in the worst case

$$T(n) \in \Theta(n \log n)$$

If the loop's condition was changed to $j \leq i$ lines 4-6 would need to change and would run according to this formula

$$\sum_{i=1}^{n} \lceil \log_2 i \rceil$$

Worst case again would be taking the log and adding 1

$$\sum_{i=1}^{n} (1 + \log_2(i)) = n + \log_2(n!)$$

Which gives us these new runtimes for each line

1. c_1
2. $c_2 n$
3. $c_3 n$
4. $c_4(n + \log_2(n!))$
5. $c_5(n + \log_2(n!))$
6. $c_6(n + \log_2(n!))$
7. c_7

The worst case running time would be

$$T(n) = c_1 + c_7 + n(c_2 + c_3 + \log_2(n!)(c_4 + c_5 + c_6))$$
$$T(n) = C_1 + C_2 n + C_3 n \log_2(n!)$$

Question 6

What value is returned by the following function? Express your answer as a function of n. From this, deduce the worst-case running time using Big Oh notation. For full credit, provide a lower bound as well, thus leading to a bound with $\Theta(\cdot)$ notation.

```
function foo2(n)
r := 0
j := n
while j ≥ 1 do
    for i := 1 to j do
        r := r + 1
    j := j/2
return r
```

This function returns $\sum_{i=1}^{\lfloor \log_2 n \rfloor} (\lfloor \frac{n}{2^{n-1}} \rfloor - 1)$ which can be simplified to

$$\sum_{i=1}^{\lfloor \log_2 n \rfloor} (\lfloor \frac{n}{2^{n-1}} \rfloor) - \lfloor \log_2 n \rfloor$$

To determine the runtime we can look at each line again

1. c_1
2. c_2
3. $c_3 \lfloor \log_2 n \rfloor$
4. $c_4 \lfloor \log_2 n \rfloor \sum_{i=1}^{j} 1$
5. $c_5 \lfloor \log_2 n \rfloor \sum_{i=1}^{j} 1$
6. $c_6 \lfloor \log_2 n \rfloor$
7. c_7

Since we only care about worst case runtime we can remove the floors and add 1 to get

$$T(n) = c_1 + c_2 + c_7 + \log_2 n(c_3 + c_6 + (c_4 + c_5)(\sum_{j=1}^{\log_2(2n)} \sum_{i=1}^{j} 1))$$

Combining constants we get

$$T(n) = C_1 + \log_2 n(C_2 + C_3(\sum_{j=1}^{\log_2(2n)} \sum_{i=1}^{2^j} 1))$$

Simplifying the sum

$$\sum_{j=1}^{\log_2(2n)} \sum_{i=1}^{2^j} 1 = \sum_{j=1}^{\log_2(2n)} 2^j = 4n - 2$$

Which means our worst-case run time is

$$T(n) = C_1 + \log_2 n(C_2 + C_3(4n - 2))$$

$$T(n) = C_1 + C_2 \log_2 n + C_3(4n - 2) \log_2 n$$

Which we can see is $O(nlogn)$ as well as $\Omega(nlogn)$ so we have

$$T(n) \in \Theta(nlogn)$$

Question 7

Show that, if c is a positive real number, then $g(n) = \sum_{i=0}^{n} c^i$

a. $\Theta(1)$ if $c < 1$

b. $\Theta(n)$ if $c = 1$

c. $\Theta(c^n)$ if $c > 1$

First we have a closed form for our sum

$$g(n) = \sum_{i=0}^{n} c^i = \frac{c^{n+1} - 1}{c - 1} \qquad c \neq 1$$

$$g(n) = \sum_{i=0}^{n} c^i = n + 1 \qquad c = 1$$

a.

$$k_1 \leq \frac{c^{n+1} - 1}{c - 1} \leq k_2 \qquad c < 1$$

Since $c^{n+1} - 1 < c - 1 \qquad c < 1$ then $0 < \frac{c^{n+1}-1}{c-1} < 1$ So we know what k_1 and k_2 have to be

$$k_1 = 0 + \epsilon; k_2 = 1$$

Where ϵ is arbitrarily small

b.

$$k_1 n \leq n + 1 \leq k_2 n \qquad c = 1$$

This holds true when

$$k_1 <= 1 \qquad k_2 >= 2$$

c.

$$k_1 c^n \leq \frac{c^{n+1} - 1}{c - 1} \leq k_2 c^n \qquad c > 1$$

$$k_1(c^{n+1} - 1) \leq c^{n+1} - 1 \leq k_2(c^{n+1} - 1) \qquad c > 1$$

Holds true when

$$k_1 <= 1 \qquad k_2 >= 2$$