PSO 1

I upload my slides on my website!
justin-zhang.com/teaching/CS251

Introductions

← YOUR PSO GTA (Justin)

YOUR PSO UTAs
Srushti
Siddarth

Some other things about this PSO (and me)

- Usually on slides
- I try to cover everything
- Raise your hand whenever
- 3-time 251 TA, 10-time overall TA

I upload my slides on my website!
justin-zhang.com/teaching/CS251

Question 1

Let c be the cost of calling the function WORK. That is, the cost of the function is constant, regardless of the input value. Determine the respective closed-form $T(n)$ for the cost of calling WORK.

```
function A1(n : Z+)
    val ← 0
    for i from 1 to n by multiplying by 3 do
        for j from i to i² do
            val ← val + WORK(n)
        end for
    end for
    return val
end function
```

What is this question asking?

Question 1

Let c be the cost of calling the function WORK. That is, the cost of the function is constant, regardless of the input value. Determine the respective closed-form $T(n)$ for the cost of calling WORK.

```
function A1(n : ℤ⁺)
    val ← 0
    for i from 1 to n by multiplying by 3 do
        for j from i to i² do
            val ← val + WORK(n)
        end for
    end for
    return val
end function
```

What is this question asking? Determine the respective closed-form $T(n)$ for the cost of calling WORK.

What is this question **NOT** asking?

Something with a $\sum$ or $\prod$ in it

An asymptotic answer

```
function A1(n : Z+)
    val ← 0
    for i from 1 to n by multiplying by 3 do
        for j from i to i^2 do
            val ← val + Work(n)
        end for
    end for
    return val
end function
```

c

(Let c be the cost of calling Work)

How to derive T(n) when there are weird loops that are nested

1. Write out the general form of T(n).

$$T(n) = \sum_{i=\_}^{\_} \sum_{j=\_}^{\_} c$$

<u>General idea</u>
loop == sum
nested loop == nested sum

```
function A1(n : Z+)
    val ← 0
    for i from 1 to n by multiplying by 3 do
        for j from i to i² do
            val ← val + WORK(n)
        end for
    end for
    return val
end function
```

(Let c be the cost of calling Work)

How to derive T(n) when there are weird loops that are nested

1. Write out the general form of T(n).

$$\sum_{i=\_}^{\_} \sum_{j=\_}^{\_} c$$

i	1	3	9	…	n
j (i to i^2)	1 to 1	3 to 9	9 to 81	---	n to n^2

2. Write out the values of i and j as the loop iterates

```
function A1(n : Z+)
    val ← 0
    for i from 1 to n by multiplying by 3 do
        for j from i to i^2 do
            val ← val + WORK(n)
        end for
    end for
    return val
end function
```

(Let c be the cost of calling Work)

How to derive T(n) when there are weird loops that are nested

1. Write out the general form of T(n).

$$\sum_{i=\_}^{\_} \sum_{J=i}^{i^2} c$$

i	1	3	9	…	n
J (i to i^2)	1 to 1	3 to 9	9 to 81		n to n^2

2. Write out the values of i and j as the loop iterates

3. Plug in the start and ending values for i and j

```
function A1(n : Z+)
    val ← 0
    for i from 1 to n by multiplying by 3 do
        for j from i to i^2 do
            val ← val + WORK(n)
        end for
    end for
    return val
end function
```

(Let c be the cost of calling Work)

How to derive T(n) when there are weird loops that are nested

1. Write out the general form of T(n).

$$\sum_{i=\_}^{\_} \sum_{j=i}^{i^2} c$$

i	1	3	9	…	n
J (i to i^2)	1 to 1	3 to 9	9 to 81		n to n^2

2. Write out the values of i and j as the loop iterates

3. Plug in the start and ending values for i and j

Problem: summations don’t “multiply by 3”

function A1($n : \mathbb{Z}^+$)
    $val \leftarrow 0$ k from 0 to Kend
    for ~~i from 1 to n by multiplying by 3~~ do
        for j from i to i^2 do
            $val \leftarrow val + \text{WORK}(n)$
        end for
    end for
    return val
end function

Problem: summations don't "multiply by 3"
Solution: introduce a new variable k that iterates 'normally' (replace i with k)

k	0	1	2	…	? e $3^{Kend} = n$
i $= 3^k$	1	3	9	…	n
j (i to i^2)	1 to 1	3 to 9	9 to 81		n to n^2

Next step: write i in terms of k

$3^{Kend} = n$

$Kend = \log_3 n$

```
function A1(n : Z+)
    val ← 0
    for i from 1 to n by multiplying by 3 do
        for j from i to i^2 do
            val ← val + WORK(n)
        end for
    end for
    return val
end function
```

Problem: summations don't "multiply by 3"

Solution: introduce a new variable k that iterates 'normally'

$3^{\log_3 n} = n$

k	0	1	2	…	$\log_3 n$
$i = 3^k$	$\mathbf{3^0}$	$\mathbf{3^1}$	$\mathbf{3^2}$	**…**	$\mathbf{3^k} = n$
j (i to i^2)	1 to 1	3 to 9	9 to 81		n to n^2

$= (3^k \text{ to } 3^{2k})$

Now write j in terms of k

```
function A1(n : ℤ⁺)
    val ← 0
    for i from 1 to n by multiplying by 3 do
        for j from i to i² do
            val ← val + WORK(n)
        end for
    end for
    return val
end function
```

Problem: summations don't "multiply by 3"

Solution: introduce a new variable k that iterates 'normally'

k	0	1	2	…	$\log_3 n$
i	3^0	3^1	3^2	…	3^k
j $(i \text{ to } i^2)$	3^0 to 3^0	3^1 to 3^2	3^2 to 3^4		3^k to 3^{2k}

$$(i \text{ to } i^2) = (3^k \text{ to } 3^{2k})$$

The corresponding sum is then..

$$\sum_{k=0}^{\log_3 n} \sum_{j=3^k}^{3^{2k}} c$$

```
function A1(n : Z+)
    val ← 0
    for i from 1 to n by multiplying by 3 do
        for j from i to i^2 do
            val ← val + WORK(n)
        end for
    end for
    return val
end function
```

Problem: summations don't "multiply by 3"
Solution: introduce a new variable k that iterates 'normally'

k	0	1	2	…	$\log_3 n$
i	3^0	3^1	3^2	…	3^k
j (i to i^2)	3^0 to 3^0	3^1 to 3^2	3^2 to 3^4		3^k to 3^{2k}

The corresponding sum is then..

Last Steps: Solve the Sum (Carefully!!)

$$\sum_{k=0}^{\log_3 n} \sum_{j=3^k}^{3^{2k}} c$$

Pro technique:

Last Steps: Solve the Sum (Carefully!!)

$$\sum_{k=0}^{\log_3 n} \sum_{j=3^k}^{3^{2k}} c = \frac{9}{8}cn^2 - \frac{3}{2}cn + \left(\frac{11}{8} + \log_3 n\right)c$$

Cheat sheet

i) consecutive sum

$$\sum_{i=0}^{k} 1 = \frac{k(k+1)}{2}$$

ii) geometric sum

$$\sum_{i=1}^{k} r^i = \frac{r^{k+1} - 1}{r-1}$$

Pro technique: *Left as an exercise to the reader*

Question 2

Derive the closed-form $T(n)$ for the value returned by the following algorithm:

```
function A2(n : Z+)
    sum ← 0
    for i from 0 to n^4 − 1 do
        for j from i to n^3 − 1 do
            sum ← sum + 1
        end for
    end for
    return sum
end function
```

Let's follow the same steps

0. What's being asked? ✓
1. Write out the general form of T(n).
2. Write out the values of i and j as the loop iterates

$$T(n) = \sum_{i=0}^{n^4-1} \sum_{j=i}^{n^3-1} 1$$

Question 2

Derive the closed-form $T(n)$ for the value returned by the following algorithm:

```
function A2(n : Z+)
    sum ← 0
    for i from 0 to n^4 − 1 do
        for j from i to n^3 − 1 do
            sum ← sum + 1
        end for
    end for
    return sum
end function
```

$i = n^3$

n^3 to $n^3 - 1$

1. Write out the general form of T(n).
2. Write out the values of i and j as the loop iterates

$$\sum_{i=0}^{n^4-1} \sum_{j=i}^{n^3-1} 1$$

Simple! (Fishy..)

Question 2

Derive the closed-form $T(n)$ for the value returned by the following algorithm:

```
function A2(n : Z+)
    sum ← 0
    for i from 0 to n^4 − 1 do
        for j from i to n^3 − 1 do
            sum ← sum + 1
        end for
    end for
    return sum
end function
```

1. Write out the general form of T(n).
2. Write out the values of i and j as the loop iterates

$$\sum_{i=0}^{n^4-1} \sum_{j=i}^{n^3-1} 1$$

When i > n^3-1, inner loop does not run so this sum is wrong!

Simple! (Fishy..)

Question 2

Derive the closed-form $T(n)$ for the value returned by the following algorithm:

```
function A2(n : ℤ⁺)
    sum ← 0
    for i from 0 to n⁴ − 1 do
        for j from i to n³ − 1 do
            sum ← sum + 1
        end for
    end for
    return sum
end function
```

1. Write out the general form of T(n).
2. Write out the values of i and j as the loop iterates

$$\sum_{i=0}^{\substack{n^3-1 \\ \cancel{n^4-1}}} \sum_{j=i}^{n^3-1} 1$$

The right sum

Last Steps: Solve the Sum (Carefully!!)

$$\sum_{j=i}^{k} 1 = k-i+1$$

$$\sum_{i=0}^{\cancel{n^4-1}\; n^3-1} \left(\sum_{j=i}^{n^3-1} 1 \right) = \sum_{i=0}^{n^3-1} \left(\underline{n^3-1-i+1}\right) = \sum_{i=0}^{n^3-1} \left(n^3-i\right)$$

$$= \underline{n^3 + (n^3-1) + \dots + (1)}$$

$$= 1 + \dots + n^3$$

$$= \sum_{i=1}^{n^3} i = \boxed{\frac{n^3(n^3+1)}{2}}$$

Question 3

(a) The following statements are true or false?

1. $n^2 \in \mathcal{O}(5^{\log n})$
2. $\frac{\log n}{\log \log n} \in \mathcal{O}(\sqrt{\log n})$
3. $n^{\log n} \in \Omega(n!)$

(b) Sort the following functions in increasing order of asymptotic (big-O) complexity:

$$f_1(n) = n^{\sqrt{n}}, \quad f_2(n) = 2^n, \quad f_3(n) = n^{10} \cdot 2^{n/2}, \quad f_4(n) = \binom{n}{2}$$

What does f(n) = O(g(n)) mean (in words)?

Question 3

(a) The following statements are true or false?

1. $n^2 \in \mathcal{O}(5^{\log n})$
2. $\frac{\log n}{\log \log n} \in \mathcal{O}(\sqrt{\log n})$
3. $n^{\log n} \in \Omega(n!)$

(b) Sort the following functions in increasing order of asymptotic (big-O) complexity:

$$f_1(n) = n^{\sqrt{n}}, \quad f_2(n) = 2^n, \quad f_3(n) = n^{10} \cdot 2^{n/2}, \quad f_4(n) = \binom{n}{2}$$

What does f(n) = O(g(n)) mean (in words)? *f(n) is upper-bounded by g(n), asymptotically*

What does f(n) = O(g(n)) *actually* mean?

$\exists\, c > 0,\ n_0 \in \mathbb{N}$ st

$cf(n) \leq g(n) \quad (n \geq n_0)$

Question 3

(a) The following statements are true or false?

1. $n^2 \in \mathcal{O}(5^{\log n})$
2. $\frac{\log n}{\log \log n} \in \mathcal{O}(\sqrt{\log n})$
3. $n^{\log n} \in \Omega(n!)$

(b) Sort the following functions in increasing order of asymptotic (big-O) complexity:

$$f_1(n) = n^{\sqrt{n}}, \quad f_2(n) = 2^n, \quad f_3(n) = n^{10} \cdot 2^{n/2}, \quad f_4(n) = \binom{n}{2}$$

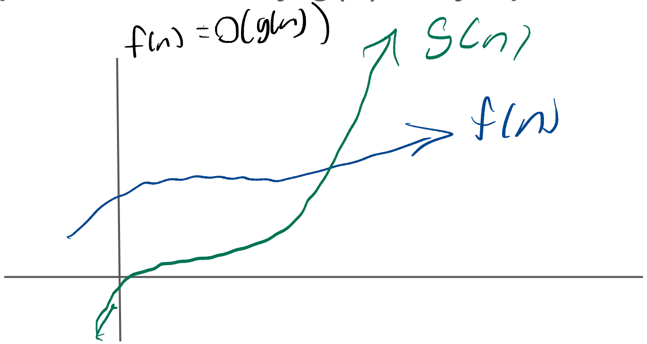
What does f(n) = O(g(n)) mean (in words)? *f(n) is upper-bounded by g(n), asymptotically*

What does f(n) = O(g(n)) *actually* mean?

There exists constants $c > 0$, $n_0 \in \mathbb{N}$ such that

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

for all $n \geq n_0$



(a) The following statements are true or false?

1. $n^2 \in \mathcal{O}(5^{\log n})$

$\log_2 4 = 2$

(Suppose log is base 2)

Wts: $\exists c > 0, n_0 \in \mathbb{N}$ st $cn^2 \leq 5^{\log n}$ $(n \geq n_0)$

n^c c $5^{\log n} = n^{\log 5}$

$n^{\log_2 5} \approx 2.3$

True $\log(5^{\log n}) = \log(n^{\log 5})$

$cn^2 \leq n^{2.3}$

$\log n \times \log 5 = \log 5 \times \log n$

$c = 1$ $n_0 = 1$

(For general log base b, this is only true when..)

$$n^2 \leq n^{\log_b 5}$$

Take log: $2 \log n \leq \log_b 5 \log n$ so when $2 \leq \log_b 5$

(a) The following statements are true or false?

2. $\frac{\log n}{\log\log n} \in \mathcal{O}(\sqrt{\log n})$

$\sqrt{n} = n/\sqrt{n}$

$$\frac{\log n}{\sqrt{\log n}}$$

$\log\log n$ vs $\sqrt{\log n}$

$\ll$

$$\frac{\log n}{\text{small}} \text{ vs } \frac{\log n}{\text{big}}$$

large $\in \Theta(\text{small})$

F

(a) The following statements are true or false?

3. $n^{\log n} \in \Omega(n!)$

What does f(n) = O(g(n)) mean (in words)? *f(n) is upper-bounded by g(n), asymptotically*

What does f(n) = O(g(n)) *actually* mean?

There exists constants $c > 0$, $n_0 \in \mathbb{N}$ such that

$cf(n) \leq g(n)$

for all $n \geq n_0$

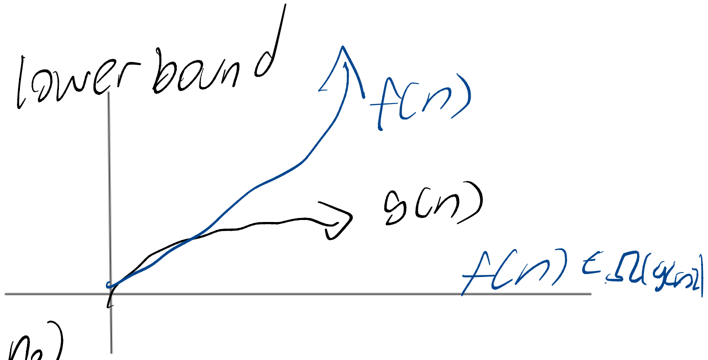
Now…

What does f(n) = Ω(g(n)) mean (in words)?

What does f(n) = Ω(g(n)) *actually* mean?

$\exists\, c > 0,\ n_0 \in \mathbb{N}$ st

$c f(n) \geq g(n)$ $(n \geq n_0)$



$n^{\log n} << n!$

(a) The following statements are true or false?

3. $n^{\log n} \in \Omega(n!)$

<u>idea</u>: Compare pairwise

I can write $n! = \overbrace{n \times (n-1) \times \ldots \times (n-\log n+1) \times \ldots \times 2 \times 1}^{n \text{ terms}}$

and $n^{\log n} = \underbrace{n \times n \times \ldots n}_{\log n \text{ terms}} \times \underbrace{1 \times 1 \times \ldots \times 1}_{n-\log n\ 1\text{'s}}$

If I divide $n!/n^{\log n}$, and look at individual terms

$$\frac{n!}{n^{\log n}} = \frac{n \times (n-1) \times \ldots (n-\log n+1) \times (n-\log n) \times \ldots \times 2 \times 1}{n \times n \times \ldots n \times 1 \times 1 \times \ldots 1 \times 1}$$

each term approaches (as $n \to \infty$)

$1 \times 1 \times$ (almost 1) $\times (n-\log n) \times 2 \times 1$

Multiplied all together, we get something <u>unbounded</u>.

i.e $\frac{n!}{n^{\log n}} \to$ infinity implying $n! >> n^{\log n}$

So it cannot be the case that $n^{\log n} \in \Omega(n!)$

False

(b) Sort the following functions in increasing order of asymptotic (big-O) complexity:

$$f_1(n) = n^{\sqrt{n}}, \quad f_2(n) = 2^n, \quad f_3(n) = n^{10} \cdot 2^{n/2}, \quad f_4(n) = \binom{n}{2}$$

$n^{\sqrt{n}}$ vs. 2^n — $2^{n/2} \cdot 2^{n/2}$; $2^{n/2} \cdot n^{10}$

$f_4 = \frac{n \times (n-1)}{2} = O(n^2)$

Intuition: Exponentials dominate

Any clear relationships?

$f_4 \ll f_1 \lesssim f_3 \lesssim f_2$

$f_1 \lesssim f_2$

$\log(n^{\sqrt{n}}) = \sqrt{n} \times \log n < \sqrt{n} \times \sqrt{n} = n < \log(2^n)$

Question 4

(a) Show that $\max\{f(n), g(n)\} \in \Theta(f(n) + g(n))$ for any $f(n)$ and $g(n)$ that eventually become and stay positive.

(b) Give an example of f and g such that f is not $O(g)$ and g is also not $O(f)$.

$f(n) = \Theta(g(n))$ is defined as $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ (simultaneously) i.e.

There exists constants $c > 0$, $n_0 \in \mathbb{N}$ such that

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

for all $n \geq n_0$

AND

There exists constants $c' > 0$, $n_1 \in \mathbb{N}$ such that

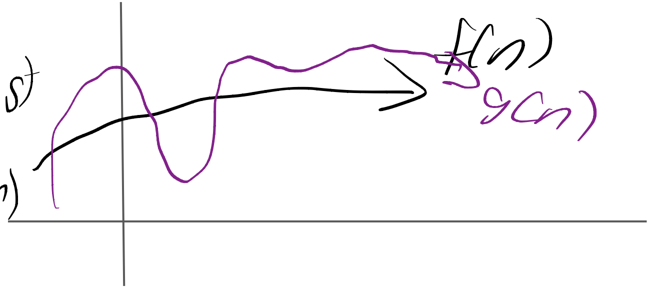
$$c' f(n) \geq g(n)$$

for all $n \geq n_1$

We can rewrite this as..

There exists constants $c, c' > 0$ and $n_0, n_1 \in \mathbb{N}$ s.t.

$$\underbrace{c f(n) \leq g(n) \text{ and } c' f(n) \geq g(n)}_{n \geq n_0, n_1}$$

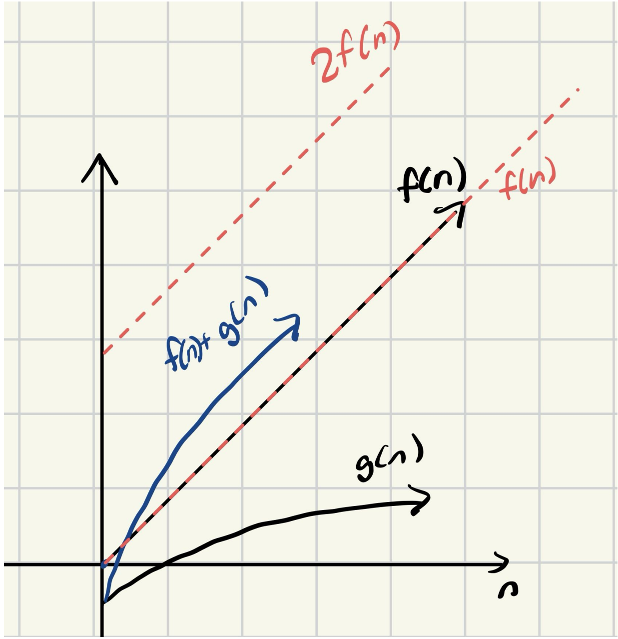


(a) Show that $\max\{f(n), g(n)\} \in \Theta(f(n) + g(n))$ for any $f(n)$ and $g(n)$ that eventually become and stay positive.

To convince ourselves, let's look at f(n) = n and g(n) = log n

(a) Show that $\max\{f(n), g(n)\} \in \Theta(f(n) + g(n))$ for any $f(n)$ and $g(n)$ that eventually become and stay positive.

Ex. f(n) = n and g(n) = log n



Prove formally using definition

There exist constants $c, c' > 0$, and $n_0 \in \mathbb{N}$ such that

$cf(n) \leq g(n)$ and $c'f(n) \geq g(n)$

for all $n \geq n_0$

1. max(f,g) = O(f + g)

wts: c, n_0 : $\max(f,g) \leq f+g$

$\max(f,g) \leq f$ or $g \leq f+g$

Take $c=1$, n_0 = the first pt where both are positive.

2. max(f,g) = **Ω**(f + g)

wts: c', n_1 : $\max(f,g) \times c' \geq f+g$

g
f avg(f,g)

$c' = 2$: $\max(f,g) \geq \frac{f+g}{2}$

(b) Give an example of f and g such that f is not $O(g)$ and g is also not $O(f)$.

Idea: crazy oscillating behavior

$f(n) = n$ $\qquad$ $g(n) = \begin{cases} 1, & n \text{ odd} \\ n^3, & n \text{ even} \end{cases}$

$n^n = n \times \dots \times n$

$n! = n \times \dots \times 1$