[Question PDF]

Do the following written exercises. If you hand-write your work, do so neatly. Work individually.

(1) Algorithmic complexity

For full credit, show your work for each problem—not just the answer. [Two points each.]
Suppose we have an algorithm with a runtime of f(n) = n(2^n) for a problem size of n.

(1a) What is f(2n)/f(n)? Simplify as much as possible.

f(2n) / f(n)

$$\frac{f(2n)}{f(n)} = \frac{(2n) * 2^{n}}{2^{(2n)}} = \frac{2 * 2^{(2n)}}{2^{n}} = \frac{2^{2n+1}}{2^{n}} = 2^{n+1} * 2^{-n} = 2^{n+1}$$

(1b) What is f(n + 1) − f(n)? Simplify as much as possible.

$f(n + 1) - f(n) = (n + 1)2^{n+1} - (n)2^{n} = (n)2^{n+1} + (1)2^{n+1} - (n)2^{n} = (n)2^{n+1} + 2^{n+1} - (n)2^{n}$

(1c) What is the largest integer n such that f(n) ≤ 1,000 ?

I know 2^n doubles each time and n increases by 1 each time

I know 2^{10} = 1024, so n has to be well under 10

$f(9) = 2048$
$f(8) = 4608$
$f(7) = 896$

896

(1d) What is the largest integer n such that f(n) ≤ 10,000 ?

From the previous problem we know $f(9) = 4608$

Because it at least doubles each time we know $f(11) > 4 * f(9) > 10,000$

now we will try $f(10) = 10240$

So it has to be $f(9)$ since $f(9) = 4608$

4608

(1e) What is the smallest integer n such that f(n + 1) − f(n) ≥ 5000 ?

From before we know $f(n + 1) - f(n)$ = $(n)2^{n+1} + 2^{n+1} - (n)2^{n}$

From before we also know $f(9) = 4608$, we will use this as a starting point.

let $g(n) = f(n + 1) - f(n)$

$g(9) = 5632$, This is too big.

$g(8) = 2560$, This is just right.

2560

(2) Algorithmic complexity

For full credit, show your work for each problem—not just the answer. [Two points each.]

Now suppose we have an algorithm with a runtime of g(n) = n $\log_2$ n.

$\log_2(x) = y \Leftarrow\Rightarrow 2^y = x$

log(cx) = log(c) + log(x)

(2a) What is g(2n)/g(n)? Simplify as much as possible.

$$\frac{g(2n)}{g(n)} = \frac{2n * log_2(2n)}{n * log_2(n)} = \frac{2 * log_2(2n)}{log_2(n)} = \frac{2 * (log_2(n) + (log_2(2))}{log_2(n)} =$$

$$\frac{2 * (log_2(n) + 1)}{log_2(n)} = \frac{2log_2(n)+2}{log_2(n)} = \frac{2log_2(n)}{log_2(n)} + \frac{2}{log_2(n)} = 2 + \frac{2}{log_2(n)}$$

(2b) What is g(2n) − g(n)? Simplify as much as possible.

$g(2n) - g(n) = 2n * log_2(2n) - n * log_2(n) = 2n * log_2(2n) - n * log_2(n)$
$= 2n(log_2(2) + log_2(n)) - log_2(2n) = 2n(1 + log_2(n)) - log_2(2n) =$
$2n + 2n * log_2(n) - log_2(2n) = n * log_2(n) + 2n$

(2c1) What is the largest integer n such that g(2n)/g(n) ≤ 2.5 ?

Because $log_2(n)$ is a logarithm, it can't be negative.

Because $log_2(n)$ is the denominator, it can't be zero

we first have to find where $log_2(n) > 0$

$log_2(0) = DNE$

$log_2(1) = 0$

$log_2(2) = 2$

So n has to be at least 2

From before we know we can rewrite $\frac{g(2n)}{g(n)}$ as: $2 + \frac{2}{log_2(n)}$

But we know that $lim\, n \to \infty\ of\ log_2(n) = 0$

so $lim\, n \to \infty$ of $2 + \frac{2}{log_2(n)} = 2$

Therefore there isn't a largest integer such that $\frac{g(2n)}{g(n)} \leq 2.5$

It might be worth noting though that the smallest integer such that $\frac{g(2n)}{g(n)} \leq 2.5$ is 16

$\frac{g(2(15))}{g(15)} = 2 + \frac{2}{15log_2(15)} \approx 2.512$

$$\frac{g(2(16))}{g(16)} = 2 + \frac{2}{16log_2(16)} = 2 + \frac{2}{4} \approx 2.5$$

(2c2) What is the largest integer n such that g(2n)/g(n) ≥ 2.5 ?

See 2c1

$$\frac{g(2(15))}{g(15)} = 2 + \frac{2}{15log_2(15)} \approx 2.512$$

$$\frac{g(2(16))}{g(16)} = 2 + \frac{2}{16log_2(16)} = 2 + \frac{2}{4} \approx 2.5$$

$$\frac{g(2(17))}{g(17)} = 2 + \frac{2}{17log_2(17)} \approx 2.489$$

(3) Run times

Suppose you have m circles and n squares, with m > 0 and n > 0. The time in seconds required for algorithm A to process the circles and squares is given by g(m, n) = (m^2)(2^n).

Answer the following questions. [Two points each.]

(3a) Assuming that both m and n must be greater than zero, what is the largest number of circles we can process in at most 1000 seconds?

$g(m, n) \leq 1000$

$m^2 2^n \leq 1000$

We want to minimize n, n ∈ $\mathbb{Z}$, and n > 0, so n = 1

$m^2 2^1 \leq 1000$

$2m^2 \leq 1000$

$m^2 \leq 500$

$m \le \sqrt{1000}$, note m > 0

$\sqrt{500} \approx$ 22.361

We could process at most 22 circles in 1000 seconds

(3b) Again, assuming that both m and n must be greater than zero, what is the largest number of squares we can process in at most 1000 seconds?

$g(m,\ n) \le 1000$

$m^2 2^n \le 1000$

We want to minimize m, m ∈ ℤ, and m > 0, so m = 1

$1^2 2^n \le 1000$

$2^n \le 1000$

$log_2(1000)\ \approx\ 9.966$

so we could process at most 9 squares in 1000 seconds

(3c) If I double both the number of squares and the number of circles, how does the runtime change? In other words, what is f(2m, 2n)/f(m, n)?

$(2m)^2 2^{(2n)}$ / $m^2 2^n$

$$\frac{(2m)^2 * 2^{(2n)}}{m^2 * 2^n}$$

$$\frac{4 * m^2 * 2^{2n}}{m^2 * 2^n}$$

$$\frac{4 * 2^{2n}}{2^n}$$

$4\ *\ 2^{2n} * 2^{-n}$

$4 \ * \ 2^n$

The runtime increases by a factor of $4 \ * \ 2^n$

(3d) Suppose that the sum of the number of squares and circles that I will process is 10. What is the longest it could take to process 10 shapes, and how many of each kind of shape would this be?

we want to maximize $g(m, n)$ while $m + n = 10$

we can get m in terms of n, $m = 10 - n$

now $g(m, n) = g(n - 10, m)$

$g(n - 10, m) = (n - 10)^2 * 2^n$

$(n^2 - 20n + 100) * 2^n$

let j(n) = $(n^2 - 20n + 100) * 2^n$, we want to find the highest value for j(n) such that $0 < n < 10$.

We will first check the end points $j(1) = g(1, 9) = 512$ and $j(9) = g(9, 1) = 162$.

Now suppose there was an even higher point, let's call it j(h).

As n goes from 1 from h, $f(n)$ approaches $f(h)$. As n goes from h from 9, $f(n)$ approaches $f(9)$. Since $j(h) > g(1)$ and $j(h) > g(9)$, $f(n)$ would increase until it reaches h then decrease after. Therefore the point h would be where $f(n)$ went from increasing to decreasing.

We can take the derivative of $f(n)$, $f'(n)$, and solve for $f'(n) = 0$

$$f'(n) = \frac{d}{dn}((n^2 - 20n + 100) * 2^n)$$

$$(\frac{d}{dn}(n^2 - 20n + 100) * 2^n) + ((n^2 - 20n + 100) * \frac{d}{dn}2^n)$$

$((2n - 20) * 2^n) + ((n^2 - 20n + 100) * ln(2) * 2^n)$

so $f'(n) = ((2n - 20) * 2^n) + ((n^2 - 20n + 100) * ln(2) * 2^n)$

we want to find $f'(n)$ = 0

$((2n - 20) * 2^n) + ((n^2 - 20n + 100) * ln(2) * 2^n) = 0$

$(((2n - 20)) + ((n^2 - 20n + 100) * ln(2))) * 2^n = 0$

Since $2^n \neq 0$ while $1 \leq n \leq 9$, the other factor has to be 0

$2n - 20 + (n^2 - 20n + 100) * ln(2)) = 0$

$2n - 20 + ln(2)n^2 - 20ln(2)n + 100ln(2) = 0$

$ln(2)x^2 + (-20ln(2)x + 2x) + (100ln(2) - 20) = 0$

Let $a = ln(2)$, $b = 2 - 20ln(2)$, and $c = -20 + 100ln(2)$

We can now use the quadratic formula $\frac{-b \pm \sqrt{(b^2 - 4ac)}}{2a}$ to get $\frac{-2+10ln(2)}{ln(2)}$

$\frac{-2+10ln(2)}{ln(2)} \approx 7$

So j(h) = j(7)

j(7) = g(3, 7)

g(3, 7) = 1152

The longest processing time would be 1152 seconds with m = 3 circles and n = 7 squares.

(4) Asymptotic bounds

Knowing that f(n) and g(n) are each O(h(n)) for some function h, show rigorously (using the definition of O(·)) that f(n) + g(n) is also O(h(n)).

Remember: f(n) is O(h(n)) if there exists c > 0 and $n_0 \geq 0$ such that for all $n \geq n_0$, f(n) ≤ ch(n).

Since $f(n)$ is $O(h(n))$ there are constants $c_1 > 0$ and $n_1 \geq 0$ such that for all $n \geq n_1$,

$f(n) \leq c_1 h(n)$

Since $g(n)$ is $O(h(n))$ there are constants $c_2 > 0$ and $n_2 \geq 0$ such that for all $n \geq n_2$,

$g(n) \leq c_2 h(n)$

To prove that $f(n) + g(n)$ is also $O(h(n))$, we need to prove there are constants $c_3 > 0$ and $n_3 \geq 0$ such that for all $n \geq n_3$, $f(n) + g(n) \leq c_3 h(n)$

We can write $f(n) + g(n) \leq c_1 h(n) + c_2 h(n)$

$c_1 h(n) + c_2 h(n) = (c_1 + c_2)h(n)$

let $c_3 = (c_1 + c_2)$

Let n_3 be a number such that $n_3 \geq n_1$ and $n_3 \geq n_2$.

for all $n \geq n_3$, $f(n) + g(n) \leq (c_1 + c_2)h(n) = c_3 h(n)$

Therefore, by the definition of O(·). $f(n) + g(n)$ is $O(h(n))$ with constants $c_3 = c_1 + c_2$ and n_3 such that $n_3 \geq n_1$ and $n_3 \geq n_2$.