CSE 101 (WI21, Kane)

Homework 0

Jiahong Long
[redacted]

08 January 2021

1. *Program Runtimes*

   Consider:

```
Alg1(n):
    For i = 1 to n        // Outer
        j = 1
        while j < n       // Inner
            j = j + 1
```

```
Alg2(n):
    For i = 1 to n        // Outer
        j = 1
        while j < n       // Inner
            j = j + i
```

   Compute the asymptotic runtime in Big-Θ.

   To begin, we take the notation "x to y" to denote that an index variable will loop *from x to y*, inclusive. That is, if we state "for i from 1 to n", it is assumed that i will take on every integer value in the interval $[1, n]$. We acknowledge that in the context of asymptotic analysis on the given algorithms, whether the interval is $[1, n]$ or $(1, n]$ or $[1, n)$ or $(1, n)$ makes no practical difference as the off-by-one difference is subsumed by the overall algorithm runtime.

   Additionally, for computational purposes, we define the base of the logarithm $\log(n)$ to be 10, and we note that since changing the base of the logarithm operator only incurs a multiplicative factor, there is no impact on the asymptotic class.

   (a) *Algorithm 1*

   We proceed towards a direct proof.

   We observe that the outer loop runs from 1 to n times, and the inner loop runs from 1 to n times. It follows immediately that there are n^2 loop iterations, and we can state that the algorithm runs in $\Theta(n^2)$: to prove this, by definition of Big-Θ we let $c_1 = 2$ and $c_2 = 2$:

$$f(n) \in \Theta(n^2) \qquad \leftrightarrow \qquad \exists c_1 \; \exists c_2 \quad n^2 \leq c_1 f(n) \bigwedge f(n) \leq c_2 n^2$$

   which must be true for all $n \geq n_0$. Define $n_0 = 0$. Substituting witnesses c_1 and c_2:

$$n^2 \leq 2n^2 \bigwedge n^2 \leq 2n^2$$

   which is vacuously true. Hence `Alg1(n)` $\in \Theta(n^2)$. ■

   (b) *Algorithm 2*

   Collaboration with [redacted]

   We proceed towards a direct proof, using a termwise bounding methodology similar to the squeeze/two-policement-and-a-drunk theorem.

Similarly to the first algorithm, the outer loop runs from 1 to n times, but the inner loop runs slightly more opaquely: at the ith iteration of the outer loop, the inner loop will run $\lceil \frac{n-1}{i} \rceil$ times. Unrolling this we obtain the summation

$$f(n) = n - 1 + \left\lceil \frac{n-1}{2} \right\rceil + \left\lceil \frac{n-1}{3} \right\rceil + \left\lceil \frac{n-1}{4} \right\rceil + \ldots + \left\lceil \frac{n-1}{n-1} \right\rceil + \left\lceil \frac{n-1}{n} \right\rceil$$

In order to find the asymptotic bound, we first bound $f(n)$ termwise:

$$\begin{array}{ccccccccccccccc}
f(n) & \geq & a(n) & = & n-1 & + & \frac{n-1}{2} & + & \frac{n-1}{3} & + & \frac{n-1}{4} & + & \ldots & + & \frac{n-1}{n-1} & + & \frac{n-1}{n} \\
 & & f(n) & = & n-1 & + & \lceil \frac{n-1}{2} \rceil & + & \lceil \frac{n-1}{3} \rceil & + & \lceil \frac{n-1}{4} \rceil & + & \ldots & + & \left\lceil \frac{n-1}{n-1} \right\rceil & + & \lceil \frac{n-1}{n} \rceil \\
f(n) & \leq & b(n) & = & n & + & \frac{n}{2}+1 & + & \frac{n}{3}+1 & + & \frac{n}{4}+1 & + & \ldots & + & \frac{n}{n-1}+1 & + & \frac{n}{n}+1
\end{array}$$

We can rewrite the lower bound as

$$a(n) = (n-1)\sum_{i=1}^{n} \frac{1}{i}$$

and the upper bound similarly as

$$b(n) = (n-1) + n\sum_{i=1}^{n} \frac{1}{i}$$

The summation terms in $a(n)$ and $b(n)$ are partial sums of the infinite harmonic series[1]. Using results from real analysis, this may be closely approximated by $\ln(n) + \gamma$ where γ is the Euler-Mascheroni constant:

$$\gamma \approx 0.577\ldots$$

It follows that we can now write

$$a(n) \approx n\ln(n) + n\gamma - \ln(n) - \gamma \qquad \text{and} \qquad b(n) \approx n\ln(n) + n\gamma + n - 1$$

We propose the Big-Θ bound of $\Theta(n\log(n))$, and from the definition of Θ-classes[2] we must find c_1 and c_2 where

$$n\log(n) \leq c_1 f(n) \bigwedge f(n) \leq c_2 n\log(n)$$

true for all $n > n_0$ where n_0 is a nonnegative integer constant, ie. f is both Big-Ω and Big-O of $n\log(n)$. Since we have found $a(n) \leq f(n) \leq b(n)$, we can find c_1 and c_2 that satisfy

$$n\log(n) \leq c_1 a(n) \leq c_1 f(n) \bigwedge f(n) \leq b(n) \leq c_2 n\log(n)$$

where we have defined a, f, and b to have an established ordering. Let $c_1 = \ln(10)$. We demonstrate that this value is a witness to the fact that $f \in \Omega(n\log(n))$:

> For ease of computation, we let $n \geq n_{0a} > 1$ such that neither n nor $\log(n)$ ever vanish over the domain, to avoid division by zero. Because we have defined that $a(n) \leq f(n)$ for all n in the domain, showing that $n\log(n) \leq c_1 a(n)$ will imply directly that $n\log(n) \leq$

[1] See `https://en.wikipedia.org/wiki/Harmonic_number`; this is also the p-series with power one.
[2] *Algorithms* p. 16, Dasgupta et al.

$c_1 f(n)$. Then, we show that our value for c_1 works:

$$\begin{aligned}
n\log(n) &\overset{?}{\leq} c_1 a(n) \\
&\overset{?}{\leq} \ln(10) n\ln(n) - \ln(n) + n\gamma - \gamma \\
\frac{1}{\ln(10)} &\overset{?}{\geq} \frac{n\log(n)}{n\ln(n) - \ln(n) + n\gamma - \gamma} \\
&\overset{?}{\geq} \frac{n\ln(n)}{n\ln(n) - \ln(n) + n\gamma - \gamma} \cdot \frac{1}{\ln(10)} \\
1 &\overset{?}{\geq} \frac{n\ln(n)}{n\ln(n) - \ln(n) + n\gamma - \gamma} \\
n\ln(n) - \ln(n) + n\gamma - \gamma &\overset{?}{\geq} n\ln(n) \\
n\gamma - \gamma &\overset{?}{\geq} \ln(n)
\end{aligned}$$

We proceed visually:

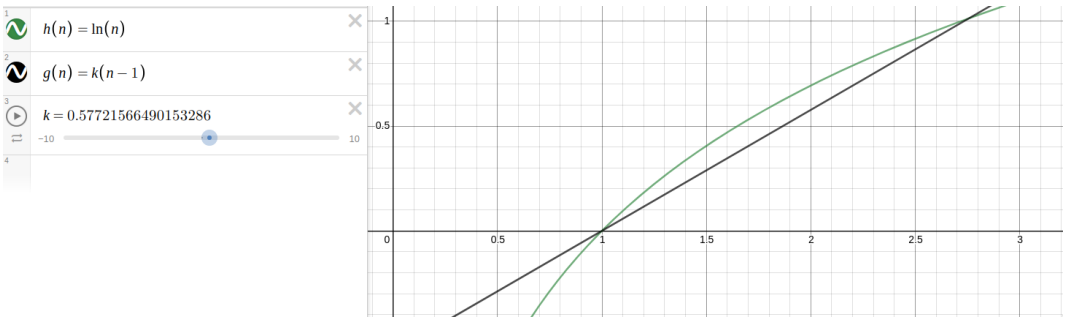

Figure 1:
Graphs of $\ln(n)$ and $\gamma(n-1)$. In the image, γ is denoted by k, a close rational approximation of the true value. The sequence is number `A001620` in the OEIS.

From this, we note that the condition is true for all $n > 3$. We can demonstrate this using a calculus argument:

Observe that

$$\frac{d}{dn}(n\gamma - \gamma) = \gamma \approx 0.577$$

and

$$\frac{d}{dn}\ln(n) = \frac{1}{n}$$

Now observe that for all $n > 3$

$$\frac{1}{n} < \frac{1}{3} \approx 0.333 < \gamma \approx 0.577$$

implying that $\ln(n)$ will grow at a slower rate than $n\gamma - \gamma$ throughout the domain, where both are monotone increasing. It follows that if at the initial boundary of the domain $\ln(n) < n\gamma - \gamma$, that same condition will hold throughout the domain. Hence we conclude that for all $n > 3$, we have $\ln(n) < n\gamma - \gamma$.

Therefore, we set $n_{0a} = 3$, and we have verified that $c_1 = \ln(10)$ and $n_{0a} = 3$ are witnesses to the condition

$$\exists c_1 \in \mathbb{R}\, \exists n_{0a} \in \mathbb{N}\, \forall n \in \mathbb{N}\, \Big(n \geq n_{01} \;\rightarrow\; n\log(n) \leq c_1 a(n) \leq c_1 f(n) \;\rightarrow\; \boxed{n\log(n) \leq c_1 f(n)}\Big)$$

as required, and consequently $f \in \Omega(n\log(n))$.

We now propose that $c_2 = \ln(10)(4+\gamma)$ and demonstrate that this value is a witness to the fact that $f \in O(n\log(n))$:

Again, for ease of computation, we let $n \geq n_{0b} > 1$ such that neither n nor $\log(n)$ ever vanish over the domain, to avoid division by zero. Because we have defined that $f(n) \leq b(n)$ for all n in the domain, showing that $b(n) \leq c_2 n\log(n)$ will imply directly that $f(n) \leq c_2 n\log(n)$. Then, we show that our value for c_2 works:

$$\begin{aligned} b(n) &\overset{?}{\leq} c_2 n\log(n) \\ n\ln(n) + n\gamma + n - 1 &\overset{?}{\leq} \ln(10)(4+\gamma)n\log(n) \\ &\overset{?}{\leq} \ln(10)(4+\gamma)n\ln(n)\cdot\frac{1}{\ln(10)} \\ \frac{n\ln(n)+n\gamma+n-1}{n\ln(n)} &\overset{?}{\leq} \frac{\ln(10)(4+\gamma)}{\ln(10)} \\ 1 + \frac{\gamma+1}{\ln(n)} - \frac{1}{n\ln(n)} &\overset{?}{\leq} 4+\gamma \end{aligned}$$

Since $\ln(n)$ is a nonzero monotone increasing function in the domain, its reciprocal must therefore be monotone decreasing. It follows that for all $n > e$ where e is the natural growth constant (ie. Euler's number, not to be confused with Euler's constant γ which we are also using here). Then, the same holds for all $n > 3 > e \approx 2.718$, so we let $n_{0b} = 3$ for simplicity. As an immediate consequence, we have the bounding conditions for all $n > n_{0b}$

$$\frac{\gamma+1}{\ln(n)} < \gamma + 1 \qquad \text{and} \qquad \frac{1}{n\ln(n)} < 1$$

The latter is true because $\frac{1}{n\ln(n)}$ is monotone decreasing for the same reason as above, and

$$\frac{1}{3\ln(3)} \approx 0.3 < 0.4$$

implying

$$\frac{1}{n\ln(n)} < 0.4 < 1$$

for $n > 3$. Thus:

$$1 + \frac{\gamma+1}{\ln(n)} - \frac{1}{n\ln(n)} \leq (1+\gamma+1+1 = 3+\gamma) \leq 4+\gamma \qquad \rightarrow \qquad \gamma \leq 1+\gamma$$

as required. Therefore, $c_2 = \ln(10)(4+\gamma)$ and $n_{0b} = 3$ are witnesses to

$$\exists c_2 \in \mathbb{R}\ \exists n_{0b} \in \mathbb{N}\ \forall n \in \mathbb{N}\ \Big(n \geq n_{0b}\ \rightarrow\ f(n) \leq b(n) \leq c_2 n\log(n)\ \rightarrow\ \boxed{f(n) \leq c_2 n\log(n)}\Big)$$

and thus $f(n) \in O(n\log(n))$.

Thus we have shown f to be both $O(n\log(n))$ and $\Omega(n\log(n))$. We can now choose $n_0 = \max(n_{0a}, n_{0b}) = 3$, and we can say that for all $n > n_0$ we have

$$n\log(n) \leq c_1 f(n) \bigwedge f(n) \leq c_2 n\log(n)$$

It follows by definition that f is $\Theta(n\log(n))$. ■

2. *Big-O Computations*

Sort the functions in terms of their asymptotic growth rates. Which of them have polynomial growth rates?

Collaboration with [redacted]

We find the Big-O class of each of the algorithms, in terms of an exponential such that we can use direct comparison tests on the exponents to determine relative rate of growth:

- $a(n) = n + n^{1/2} + n^{1/3} + \ldots + n^{1/n}$

  Asymptotically [3], the only term that matters is the n term. Hence, $\boxed{a(n) \in O(n) = O(2^{\log_2 n})}$.

- $b(n) = 3^{\lceil \log_2 n \rceil}$

  We can approximate $b(n)$ with $b'(n) = 3^{\log_2 n}$, since exponentials are monotone increasing and $|\lceil t \rceil - t| \leq 1$ for all t, which is a negligible difference in the limit. This implies

  $$\frac{1}{3} \leq \left| \frac{3^{\lfloor \log_2 n \rfloor}}{3^{\log_2 n}} \right| \leq 3$$

  and $\frac{1}{3}, 3$ being finite, implies the two are Big-Θ of each other and therefore approximate the other well. Then:

  $$\begin{aligned} b'(n) &= 3^{\log_2 n} \\ &= 3^{\log_3 n \cdot \frac{1}{\log_3 2}} \\ &= n \cdot 3^{\frac{1}{\log_3 2}} \end{aligned}$$

  and trivially, $\boxed{b(n) \approx b'(n) \in O(n) \in O(2^{\log_2 kn})}$ where $k = 3^{\frac{1}{\log_3 2}} \approx 5.7$.

- $c(n) = n^2(2 + \cos(n))$

  Note:

  $$\begin{aligned} c(n) &= n^2(2 + \cos(n)) \\ n^2(2-1) \leq c(n) &= n^2(2 + \cos(n)) < n^2(2+1) \end{aligned}$$

  and clearly $\boxed{c(n) \in O(n^2) = O(2^{2\log_2 cn})}$ where $c \in [1, 3]$.

- $d(n) = n^{100} 2^{n/2}$

  The polynomial term is subsumed [4] by the exponential term, and we obtain $\boxed{d(n) \in O(2^{n/2 + 100 \log_2 n})}$.

- $e(n) = 2^n$

  Vacuously, $e(n) \in O(2^n)$.

Then we have the ordering, from slowest to fastest growth:

Big-O class	**More specific exponential Big-O Class**	**Function**
$O(n)$	$O(2^{\log_2 n})$	$a(n)$
$O(n)$	$O(2^{\log_2 nk}), k \approx 5.7$	$b(n)$
$O(n^2)$	$O(2^{2\log_2 cn}), c \in [1, 3]$	$c(n)$
$O(a^n)$	$O(2^{2/n + 100 \log_2 n})$	$d(n)$
$O(a^n)$	$O(2^n)$	$e(n)$

The functions with polynomial growth, therefore, are $a(n)$, $b(n)$, and $c(n)$.

[3] *Algorithms* p. 17, Dasgupta et al.
[4] See footnote 3.

3. *Walks and Paths*

In a graph G, we say there is a walk from vertex u to another vertex w if there is a sequence of vertices $u = v_0, v_1, \ldots, v_n = w$ such that (v_i, v_{i+1}) is an edge of G for each $0 \leq i < n$. Prove that if there is a walk from u to w, there is a walk where all of these vertices v_i are distinct.

We proceed towards a direct proof by cases. We also note that all walks in consideration are arbitrarily long in length, but ultimately consist of finitely many edges and consequently finitely many vertices. This is a consequence of the fact that walks are defined as a sequence that is indexed by (can be put in a bijection with) a contiguous subset of $\mathbb{N}_0$ beginning at zero. In addition, we emphasise that the first and last vertices u and w in a walk are defined to be distinct, and as a consequence consider only walks of at least length two.

Suppose that there exists a walk consisting of L vertices from u to w. We denote this walk as the L-tuple $\mathfrak{W}$:

$$\mathfrak{W} = (u = v_0, v_1, v_2, \ldots, v_{L-2}, w = v_{L-1})$$

Then, either $\mathfrak{W}$ has repeated vertices or it does not.

(a) *Case 1. No repeated vertices.*
It immediately follows that $\mathfrak{W}$ is a walk from u to w consisting of all distinct vertices v_i. The statement is vacuously true. ■

(b) *Case 2. Repeated vertices.*
In this case, $\mathfrak{W}$ contains repeated vertices. We note that $\mathfrak{W}$ is a finite walk of L vertices. Therefore, $\mathfrak{W}$ contains at most finitely many repeated vertices, as it contains only finitely many vertices. Let Q denote the total number of vertices which have been repeated disregarding multiplicity, eg. if the vertex v_x is repeated twice, the vertex v_y is repeated thrice, and the vertex v_z is repeated four times, then $Q = 3$, for v_x, v_y, and v_z. We perform the following iterative process at most Q times, or at until there are no more repeated vertices, whichever is sooner:

Loop:

i. **End condition:** If $\mathfrak{W}$ does not contain any repeated vertices, then we have successfully found a walk containing no repeated vertices from u to w, given that that an arbitrary walk from u to w exists. ■

ii. Otherwise, denote q as the number of vertices which are currently repeated in $\mathfrak{W}$.

iii. Let v_r denote the first vertex that is repeated in $\mathfrak{W}$. Let a be the index of its first occurrence, and let b be the index of its last occurrence such that $v_r = v_a = v_b$.

iv. **Property:** Observe that the walk X from $u = v_0$ to $v_a = v_r$

$$X = (u = v_0, \ldots, v_a = v_r)$$

contains no repeated vertices by definition. The walk Y from $v_b = v_r$ to $w = v_{L-1}$

$$Y = (v_b = v_r, \ldots, w = v_{L-1})$$

may contain at most $q - 1$ repeated vertices, since we have ensured that v_a appears just once in Y.

v. Redefine $\mathfrak{W}$ as the concatenation of (X excluding its last element) and (Y in its entirety):

$$\mathfrak{W} = (u = v_0, \ldots, v_a = v_r = v_b, \ldots, w = v_{L-1})$$

vi. **Loop invariant:** Observe that now, $\mathfrak{W}$ contains at most $q - 1$ repeated vertices, ie. we have removed one repeated vertex, since we have eliminated duplication of the vertex v_r.

vii. **Loop invariant:** Observe that it is impossible to now have zero vertices remaining, since the first and last vertices u and w are defined to be distinct, and removing vertex repetitions will not remove u or w.

viii. Having removed one repeated vertex, go to step (1).

Hence we have shown by cases that if there exists a walk from u to w, there must exist a walk where all vertices are distinct. ■

4. *Recurrence Relations*

Consider

$$T(1) = 1, T(n) = 2T(\lfloor n/2 \rfloor) + n$$

(a) What is the exact value of $T(2^n)$?

Unrolling the recurrence relation:

$$\begin{aligned}
T(2^n) &= 2T(\lfloor 2^n/2 \rfloor) + 2^n \\
&= 2T(\lfloor 2^{n-1} \rfloor) + 2^n && \forall n \in \mathbb{N}\ 2^n \in \mathbb{Z} \\
&= 2T(2^{n-1}) + 2^n \\
&= 2^2 T(2^{n-2}) + 2(2^n) \\
&= 2^3 T(2^{n-3}) + 3(2^n) \\
\text{At the } i\text{th step where } n - i > 0: \quad & 2^i T(2^{n-i}) + i2^n \\
\text{At the } n\text{th step:} \quad & 2^n T\left(\cancel{2^{n-n}}\ 1\right) + n2^n \\
&= 2^n \cancel{T(1)}\ 1 + n2^n \\
&= 2^n + n2^n \\
&= \boxed{2^n(n+1)}
\end{aligned}$$

(b) Give a Θ expression for $T(n)$.

To begin, we unroll $T(x)$ for arbitrary x an obtain a closed form for the recurrence:

$$\begin{aligned}
T(x) &= 2T\left(\left\lfloor \frac{x}{2} \right\rfloor\right) + x \\
&= 2^2 T\left(\left\lfloor \frac{x}{2^2} \right\rfloor\right) + 2x \\
&= 2^3 T\left(\left\lfloor \frac{x}{2^3} \right\rfloor\right) + 3x \\
\text{At the } i\text{th step where } 2^{i+1} < x: \quad & 2^i T\left(\left\lfloor \frac{x}{2^i} \right\rfloor\right) + ix \\
\text{At the final step where } l = \lfloor \log_2 x \rfloor \to 2 > \frac{x}{2^l} > 1: \quad & 2^l + lx
\end{aligned}$$

Therefore we have found the closed form

$$T(x) = 2^{\lfloor \log_2 x \rfloor} + x\lfloor \log_2 x \rfloor$$

We observe that $T(n)$ is a monotone increasing function of n, by way of termwise comparison. Let $a < b$, then:

$$\begin{array}{rcccl}
T(a) & = & 2^{\lfloor \log_2 a \rfloor} & + & a\lfloor \log_2 a \rfloor \\
T(b) & = & 2^{\lfloor \log_2 b \rfloor} & + & b\lfloor \log_2 b \rfloor
\end{array}$$

Because logarithms are monotone increasing, it is vacuously true that

$$\log_2 a < \log_2 b$$

and consequently since $a < b$

$$a\lfloor \log_2 a \rfloor \leq b\lfloor \log_2 b \rfloor$$

and immediately, because exponentials are monotone increasing

$$2^{\lfloor \log_2 a \rfloor} \leq 2^{\lfloor \log_2 b \rfloor} \qquad \text{and} \qquad a\lfloor \log_2 a \rfloor \leq b\lfloor \log_2 b \rfloor$$

so

$$T(a) \leq T(b)$$

thus we have shown $T(n)$ is monotone increasing. We can now use bounding conditions on $T(n)$ to show that $T(n) \in \Theta(n \log(n))$. By definition, we now must find c_1, c_2 where

$$n \log(n) \leq c_1 T(n) \qquad \text{and} \qquad T(n) \leq c_2 n \log(n)$$

for all $n \geq n_0$ where n_0 is a nonnegative integer constant. Let $c_1 = 2$. We will show the condition holds:

For simplicity, let $n_0 = 2$. We note that by termwise comparison over the domain

$$\underbrace{2^{\log_2 n} + n \log_2 n}_{\text{LHS}} \qquad < \qquad 2T(n) = \underbrace{2^{\lfloor \log_2 n \rfloor + 1} + 2n\lfloor \log_2 n \rfloor}_{\text{RHS}}$$

Therefore, if we can show

$$n \log(n) \leq (\text{LHS})$$

then it follows immediately by definition that

$$n \log(n) \leq (\text{LHS}) \leq c_1 T(n)$$

With this in mind, we continue:

$$\begin{aligned}
n \log(n) &\leq (LHS) < c_1 T(n) \\
&\leq \left(2^{\log_2 n} + n \log_2 n\right) \\
&\leq n + n \log_2 n \\
\log(n) &\leq 1 + \log_2 n \\
&\leq 1 + \frac{\log(n)}{\log(2)} \\
1 &\geq \frac{\log(n)}{1 + \frac{\log(n)}{\log(2)}} \\
&\geq \frac{\log(n)\log(2)}{\log(2) + \log(n)}
\end{aligned}$$

Observe that the derivative of the left-hand side is

$$\frac{d}{dn}\left[\frac{\log(n)\log(2)}{\log(2) + \log(n)}\right] = \frac{\ln 2}{\ln 10} \cdot \frac{\ln n}{\ln n + \ln 2}$$

which is positive for all $n > 1$. Hence the function is strictly increasing over the domain. Therefore, if the function converges to defined value as $n \to \infty$, we can state that this value L is the maximum value of the right-hand side over the domain:

$$\begin{aligned}
L &= \lim_{n \to \infty} \frac{\log(n)\log(2)}{\log(2) + \log(n)} && \text{L'Hôpital:} \\
&= \lim_{n \to \infty} \frac{\ln 2}{\ln 10} \cdot \frac{\ln n}{\ln n + \ln 2} && \text{L'Hôpital:} \\
&= \lim_{n \to \infty} \frac{\ln 2}{\ln 10} \cdot \frac{\cancel{\ln n}}{\cancel{\ln n}} \\
&= \log 2
\end{aligned}$$

Then the function does not attain any value greater than $\log 2$ over the domain, and hence we have the bounding condition

$$1 > \log 2 > \frac{\log(n)\log(2)}{\log(2)+\log(n)}$$

which satisfies the requirement. Thus $c_1 = 2$ and $n_0 = 2$ are witnesses to the fact that $T(n) \in \Omega(n\log(n))$.

Now, let $c_2 = \left\lceil \frac{2}{\log 2} \right\rceil$. We prove that $T(n) \in O(n\log(n))$:

Here, the proof proceeds in an almost identical manner as before. For simplicity, let $n_0 = 2$. To begin, note

$$\lfloor t \rfloor \leq t$$

Then we can note that by termwise comparison over the domain that

$$T(n) = 2^{\lfloor \log_2 n \rfloor + 1} + n\lfloor \log_2 n \rfloor \qquad \leq \qquad 2^{\log_2 n} + n\log_2 n$$

Therefore, if we can show

$$2^{\log_2 n} + n\log_2 n \leq c_2 n\log(n)$$

then it follows immediately by definition that

$$T(n) \leq c_2 n\log(n)$$

which is as required. With this in mind, we continue:

$$\begin{aligned}
c_2 n\log(n) \geq \frac{2}{\log 2} n\log(n) &\geq 2^{\log_2 n} + n\log_2 n > T(n) \\
&\geq n + n\log_2 n \\
\frac{2}{\log 2}\log(n) &\geq 1 + \log_2 n \\
&\geq 1 + \frac{\log(n)}{\log(2)} \\
\frac{1}{c_2} \leq \frac{\log 2}{2} &\leq \frac{\log(n)}{1 + \frac{\log(n)}{\log(2)}} \\
&\leq \frac{\log(n)\log(2)}{\log(2)+\log(n)}
\end{aligned}$$

Observe that the derivative of the right-hand side is

$$\frac{d}{dn}\left[\frac{\log(n)\log(2)}{\log(2)+\log(n)}\right] = \frac{\ln 2}{\ln 10} \cdot \frac{\ln n}{\ln n + \ln 2}$$

which is positive for all $n > 1$. Hence the function is strictly increasing over the domain. It follows that if we sample the function at the leftmost boundary of the domain (ie. the domain is $n \geq n_0$, so we evaluate the function at n_0) we can obtain a value which can bound the function from the bottom for all values it attains over its domain. Thus:

$$\begin{aligned}
\left.\frac{\log(n)\log(2)}{\log(2)+\log(n)}\right|_{n=n_0=2} &= \frac{\log(2)\log(2)}{\log(2)+\log(2)} \\
&= \frac{\log(2)\log(2)}{2\log(2)} \\
&= \frac{\log 2}{2}
\end{aligned}$$

Then the function does not attain any value less than $\frac{\log 2}{2}$ over the domain, and hence we have the bounding condition

$$\frac{1}{c_2} = \frac{1}{\lceil \frac{2}{\log 2} \rceil} \leq \frac{\log 2}{2} \leq \frac{\log(n)\log(2)}{\log(2) + \log(n)}$$

which satisfies the requirement. Thus $c_2 = \left\lceil \frac{2}{\log 2} \right\rceil$ and $n_0 = 2$ are witnesses to the fact that $T(n) \in O(n \log(n))$.

Having demonstrated that $T(n)$ is both $O(n \log n)$ and $\Omega(n \log n)$, we have in turn shown that $T(n) \in \Theta(n \log n)$ using the witnesses $c_1 = 2$, $c_2 = \frac{2}{\log 2}$, and $n_0 = 2$. ■

(c) Consider the following purported proof that $T(n) \in O(n)$ by induction:

If $n = 1$, then $T(1) = 1 = O(1)$.
If $T(m) = O(m)$ for $m < n$, then

$$T(n) = 2T(\lfloor n/2 \rfloor) + n = O(n) + O(n) = O(n)$$

Thus T(n) = O(n).

What is wrong with this proof?

Collaboration with [redacted]

The author mistakenly dropped the $+n$ term, which is iterated upon in subsequent iterations of recursion. The actual definition of $T(n)$ is:

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T\left(\lfloor \frac{n}{2} \rfloor\right) + n & \text{otherwise} \end{cases}$$

The author made the assumption that the $+n$ term in the recursive definition of $T(n)$ exists in a vacuum, ie. purely in this stack frame of recursion. When the loop is unrolled (see part (b)) we find that the closed form of $T(n)$ is

$$T(n) = 2^{\lfloor \log_2 n \rfloor} + n\lfloor \log_2 n \rfloor$$

The second term is incurred due to the repeated stacking of the $+n$ term in successive iterations of recursion, but the incorrect proof mistakenly drops this term. The first $2^{\lfloor \log_2 n \rfloor} \approx n$ term is preserved in the flawed proof as $O(n)$, but the second $n\lfloor \log_2 n \rfloor \approx n \log_2 n \in O(n \log(n))$ term is missing.

We can observe the impact this makes by directly applying the definition of Big-O: Suppose, as the flawed proof does, that $T(n) \in O(n)$. Then, by definition, there exists a constant c_1 such that

$$T(n) \leq c_1 n$$

for all $n > n_0$ where n_0 is a nonnegative integer constant. As the flawed (strong) induction hypothesis, assume that $T(m) = O(m)$ for all $m < n$. From this follows a contradiction:

$$\begin{aligned} T(m) \leq c_1 O(n): \\ T(n) &= 2T(\lfloor n/2 \rfloor) + n && \leq 2(c_1 \frac{n}{2}) + n \text{ since } \frac{n}{2} < n \\ &\leq c_1 n + n \\ &\leq (c_1 + 1)n \end{aligned}$$

which is a direct contradiction: the coefficient *will actually increase* as $n \to \infty$, making it impossible to find a *single* constant value c_1 that makes the $O(n)$ bound valid.

5. *Time spent*

9hr2m as of last update. Tracking done using Wakatime (`wakatime.com`).