CSCI 3104: Algorithms
Spring 2022
Recitation #12 - Midterm 2 Review I

SOLUTIONS

Problem 1

Analyze the runtime of the following code snippets and prove an asymptotic bound.

a.

```
procedure loopsA(integer n):
  for  i = 1; i <= n:
     i = i + 2;
     print i;
     for j = 5; j <= n:
        j = j * 2;
        print j;
end procedure
```

- *Derive bound.* The loops here are independent. We first analyze the inner loop. Since j is doubled at each iteration, the loop takes at most k iterations where $5 \cdot 2^k \geq n$, so we take $k = \lceil \log_2(n/5) \rceil$. The initialization of j takes one operation, and for each iteration we also: perform a check that $j \leq n$ (1 op), increment j (2 ops), and print (1 op). Thus, our loop's total runtime is

$$1 + \sum_{k=1}^{\lceil log_2(\frac{n}{5}) \rceil} 4 \approx 1 + 4\log_2(n/5).$$

We next analyze the runtime of the outer loop. At each iteration, i is incremented by 2, so the loop takes at most ℓ iterations, where $1+2\ell \geq n$, so we take $\ell = \lceil (n-1)/2 \rceil$. The initialization of i takes one operation and for each iteration we also: perform a check that $i \leq n$ (1 op), increment i (2 ops), print (1 op), and run a full inner loop ($1 + 4\log_2(n/5)$ ops). Thus, the outer loop's total runtime is

$$1 + \sum_{\ell=1}^{\lceil \frac{n-1}{2} \rceil} (4 + 1 + 4\log_2(n/5)) \approx 1 + \left(\frac{n-1}{2}\right)(5 + 4\log_2(n/5))$$

- *Prove asymptotic bound.* We claim this is $\Theta(n \log n)$, and use the limit comparison test to show this:

$$
\begin{aligned}
\lim_{n\to\infty} \frac{1+\left(\frac{n-1}{2}\right)(5+4\log_2(n/5))}{n\log n} &= \lim_{n\to\infty} \frac{1+5n/2-5/2+2n\log_2(n)-2n\log_2 5-2\log_2 n+2\log_2 5}{(\log_2 10)n\log_2 n} \\
&= \lim_{n\to\infty} \frac{1-5/2+2\log_2 5}{(\log_2 10)n\log_2 n} + \frac{n/2-2n\log_2 5}{(\log_2 10)n\log_2 n} - \frac{2\log_2 n}{(\log_2 10)n\log_2 n} + \frac{2}{\log_2 10} \\
&= \frac{2}{\log_2 10} + \lim_{n\to\infty} \frac{1-5/2+2\log_2 5}{(\log_2 10)n\log_2 n} + \frac{1/2-2\log_2 5}{\log_2 10\log_2 n} - \frac{2}{n\log_2 10} \\
&= \frac{2}{\log_2 10},
\end{aligned}
$$

since in the second-to-last line all denominators go to ∞ with n while the numerators are constant.

b.

```
procedure loopsB(integer n):
  for  i = 1; i <= n:
     i = i * 2;
     print i;
     for j = 1; j <= i:
        j = j + 2;
        print j;
end procedure
```

- *Derive bound.* The loops here are dependent. We first analyze the inner loop. Since j is incremented by 2 at each iteration, the loop takes at most k iterations where $1+2k \geq i$, so we take $k = \lceil (i-1)/2 \rceil$. The initialization of j takes one operation, and for each iteration we also: perform a check that $j \leq i$ (1 op), increment j (2 ops), and print (1 op). Thus, our loop's total runtime is

$$1 + \sum_{k=1}^{\lceil (i-1)2 \rceil} 4 \approx 2i - 1.$$

We next analyze the runtime of the outer loop. i is doubled at each iteration, so the loop takes at most ℓ iterations, where $1 \cdot 2^\ell \geq n$, so we take $\ell = \lceil \log_2 n \rceil$. The initialization of i takes one operation and for iteration i we also: perform a check that $i \leq n$ (1 op), increment i (2 ops), print (1 op), and run a full inner loop ($2i-1$ ops). Thus, we can compute the total runtime of the outer loop by summing over the iteration number ℓ, and remembering that $i = 1 \cdot 2^\ell$:

$$
\begin{aligned}
1 + \sum_{\ell=1}^{\lceil \log_2 n \rceil} (4 + 2i - 1) &= 1 + 3 \lceil \log_2 n \rceil + \sum_{\ell=1}^{\lceil \log_2 n \rceil} 2 \cdot 2^\ell \\
&\approx 1 + 3 \log_2 n + 2 \frac{1 - 2^{\log_2 n + 1}}{1 - 2} \\
&= 1 + 3 \log_2 n - 2(1 - 2n) \\
&= 2n + 3 \log_2 n - 1.
\end{aligned}
$$

- *Prove asymptotic bound.* We claim this is $\Theta(n)$, and use the limit comparison test to show this:

$$
\begin{aligned}
\lim_{n \to \infty} \frac{2n + 3 \log_2 n - 1}{n} &= 2 + \lim_{n \to \infty} \frac{3 \log_2 n - 1}{n} \\
&= 2 + \lim_{n \to \infty} \frac{3}{n \ln 2} = 2,
\end{aligned}
$$

where the last line follows using L'Hôpital's rule.

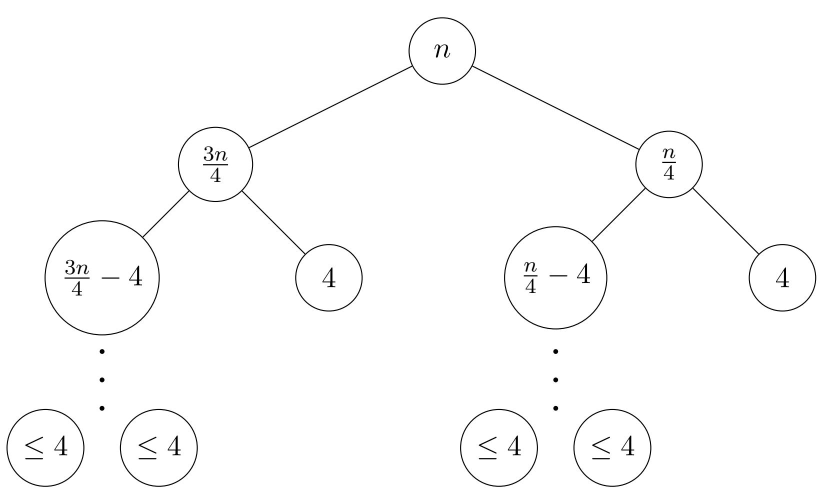
Problem 2

Consider a modified mergesort algorithm which, on alternating levels of the recursion, partitions the input into either an $(4, n-4)$ or $(n/4, 3n/4)$ split. Assume the first partition is $(n/4, 3n/4)$.

(*hint:* it may help to draw out the recurrence tree.)

a. Write down a recurrence for the modified mergesort algorithm.

The recurrence tree is:



We can write the recurrence as two separate recurrences, based on the depth of our recursive call:

$$T_{\text{even}}(n) = \begin{cases} T_{\text{odd}}(n/4) + T_{\text{odd}}(3n/4) + O(n) & \text{if } n > 4, \\ O(1) & \text{if } n \leq 4 \end{cases}$$

$$T_{\text{odd}}(n) = \begin{cases} T_{\text{even}}(n-4) + T_{\text{even}}(4) + O(n) & \text{if } n > 4, \\ O(1) & \text{if } n \leq 4 \end{cases}$$

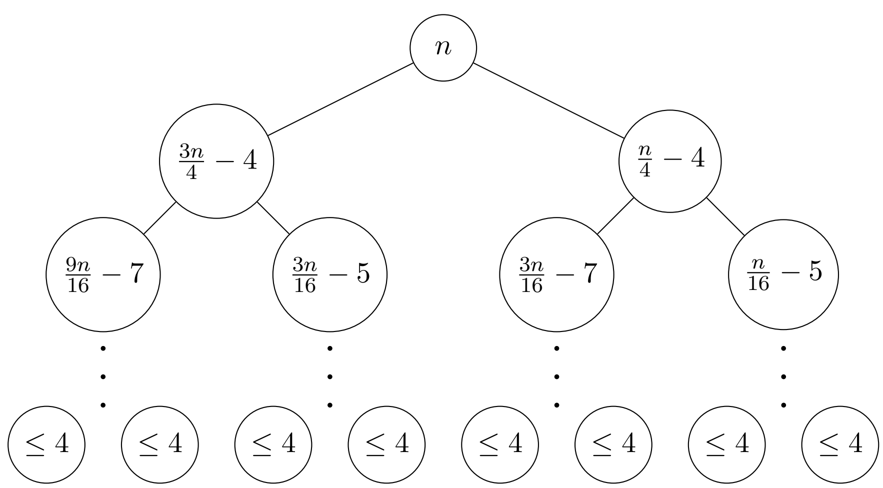
Note that since $T_{\text{even}}(4) = O(1)$, we can simplify the odd case where $n > 4$ to $T_{\text{odd}} = T_{\text{even}}(n-4) + O(n)$. Since solving two recurrences is difficult, we can instead unroll one level of the even recurrence to get a recurrence for the even-depth calls in terms of other even-depth calls only. This is equivalent to combining every two levels of our recursion tree:

$$T_{\text{even}}(n) = \begin{cases} T_{\text{even}}(n/4 - 4) + O(n/4) + T_{\text{even}}(3n/4 - 4) + O(3n/4) + O(n) & \text{if } n > 4, \\ O(1) & \text{if } n \leq 4 \end{cases}.$$

Again, we can simplify the larger case to $T_{\text{even}}(n) = T_{\text{even}}(n/4-4) + T_{\text{even}}(3n/4-4) + O(n)$.

b. Solve the recurrence relation using the tree method. How does this tree compare to the recursion tree from the previous problem?

1. *Draw the tree.*



with linear work at each vertex.

This tree essentially collapses every two levels from our original recurrence into one level, and removes the vertices with a constant input size as they don't add significantly to the merge work done at each level.

2. *Determine the depth.* The slowest-reducing branch of the input is the part that always takes the $3n/4 - 4$ branch in our tree. Thus, we have certainly bottomed

$\frac{3}{4}n - 4 \Rightarrow \frac{3}{4}\left(\frac{3}{4}n - 4\right) - 4$

$\frac{3}{4}^2 n - \frac{3}{4} \cdot 4 - 4$

$$\left(\frac{3}{4}\right)^k n - 4 \cdot \sum_{i=0}^{k} \left(\frac{3}{4}\right)^i \leq 4$$

$$\Rightarrow k = \Theta(\lg n).$$

out all branches at depth k when $(3/4)^k n - 4\sum_{i=0}^{k}(3/4)^i \leq 4$, since the subtraction of constants only reduces our input size faster. So our tree is of depth at most $\left\lceil \log_{4/3}((n+3/4)/5) \right\rceil$.

3. *Sum the work.* Since the input is divided exactly across all vertices in a level and each vertex runs in $O(n)$ time, each level does at most $O(n)$ work, and we can compute the runtime as

$$\sum_{i=1}^{\lceil \log_{4/3}((n+3/4)/5) \rceil} O(n) \in O(n \log n).$$

Asymptotics Cheat Sheet

(Limit Comparison Test) If the limit $L := \lim_{x\to\infty} \frac{f(x)}{g(x)}$ exists, then:

- if $0 < L < \infty$, then $f(x) \in \Theta(g(x))$ ("they grow at the same rate")
- if $L = 0$, then $f(x) \in O(g(x))$, but $f(x) \notin \Theta(g(x))$ ("g grows faster than f")
- if $L = \infty$, then $g(x) \in O(f(x))$, but $g(x) \notin \Theta(f(x))$ ("f grows faster than g")

(L'Hôpital's rule) Suppose g and f are both differentiable functions, with either

a. $\lim_{x\to\infty} f(x) = 0$ and $\lim_{x\to\infty} g(x) = 0$; or

b. $\lim_{x\to\infty} f(x) = \pm\infty$ and $\lim_{x\to\infty} g(x) = \pm\infty$

If $\lim_{x\to\infty} \frac{f(x)}{g(x)}$ exists, then $\lim_{x\to\infty} \frac{f(x)}{g(x)} = \lim_{x\to\infty} \frac{f'(x)}{g'(x)}$.

Some Helpful Identities

$$a\sum_{i=0}^{n} r^i = a\left(\frac{1-r^{n+1}}{1-r}\right) \qquad \text{(Finite Geometric Series)}$$

$$a\sum_{i=0}^{n} i = \frac{an(n+1)}{2} \qquad \text{(Sum of first } n \text{ integers/special case of Finite Arithmetic Series)}$$

$$\sum_{i=0}^{n} ix^i = \frac{x(nx^{n+1} - (n+1)x^n + 1)}{(x-1)^2}$$

$$\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$