Name: Vera Duong

ID: 109431166

Due February 12, 2021

CSCI 3104, Algorithms **Spring 2021, CU-Boulder**

Problem Set 4 (50 points) Collaborators: Nathan Straub, Matt Hartnett

---

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.
Advice 2: Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

Instructions for submitting your solution:

- The solutions **should be typed** and we cannot accept hand-written solutions. Here's a short intro to Latex.
- You should submit your work through **Gradescope** only.
- The easiest way to access Gradescope is through our Canvas page. There is a Gradescope button in the left menu.
- Gradescope will only accept **.pdf** files.
- It is vital that you match each problem part with your work. Skip to 1:40 to just see the matching info.

Name: Vera Duong
ID: 109431166

Due February 12, 2021

CSCI 3104, Algorithms **Spring 2021, CU-Boulder**

Problem Set 4 (50 points) Collaborators: Nathan Straub, Matt Hartnett

Recall that a function f expressed in terms that depend on f itself is a recurrence relation. "Solving" such a recurrence relation means expressing f without terms that depend on f.

1. Solve the following recurrence relations using the **unrolling method** (also called plug-in or substitution method), and find tight bounds on their asymptotic growth rates. Remember to show your work so that the graders can verify that you used the **unrolling method**. Assume that all function input sizes are non-negative integers. You may also assume that integer rounding of any fraction of a problem size won't affect asymptotic behavior.

   (a) $U_a(n) = \begin{cases} 2U_a(n-1) - 1 & \text{when } n \geq 1, \\ 2 & \text{when } n = 0. \end{cases}$

   (b) $U_b(n) = \begin{cases} 3U_b(n/4) + n/2 & \text{when } n > 3, \\ 0 & \text{when } n = 3. \end{cases}$

   Solution:

   (a)

$$\begin{aligned} U_a(n) &= 2U_a(n-1) - 1 \\ U_a(n-1) &= 2U_a(n-2) - 1 \\ U_a(n-2) &= 2U_a(n-3) - 1 \\ U_a(n-3) &= 2U_a(n-4) - 1 \end{aligned}$$

   We can plug these back into the original to find a pattern:

$$\begin{aligned} U_a(n) &= 2U_a(n-1) - 1 \\ U_a(n) &= 2(2U_a(n-2) - 1) - 1 = 4U_a(n-2) - 3 \\ U_a(n) &= 4(2U_a(n-3) - 1) - 3 = 8U_a(n-3) - 7 \\ U_a(n) &= 8(2U_a(n-4) - 1) - 7 = 16U_a(n-4) - 15 \end{aligned}$$

   After k times of recurrence: $U_a(n) = 2^k U_a(n-k) - (2^k - 1)$

   Solve for the base case when $n - k = 0 \Rightarrow n = k$

   Plug in the solution with our variable k:

   $U_a(n) = 2^n \cdot 2 - (2^n - 1)$ Here we replace the $U_a(n)$ with 2
   $U_a(n) = 2^n \cdot 2 - 2^n + 1$
   $U_a(n) = 2^n(2-1) + 1$
   $U_a(n) = 2^n + 1$

   Taking $f(n) = 2^n + 1$ and $g(n) = 2^n$, we can do a limit comparison test:

$$\lim_{n\to\infty} \frac{2^n + 1}{2^n} \Rightarrow \lim_{n\to\infty} \frac{2^n}{2^n} = 1, \lim_{n\to\infty} \frac{1}{2^n} = 0, 1 + 0 = 1$$

   We can see here that $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ is a constant value, so $U_a(n)$ is $\boldsymbol{\Theta(2^n)}$

Name: Vera Duong
ID: 109431166

(b)

$$U_b(n) = 3U_b\left(\frac{n}{4}\right) + \frac{n}{2}$$
$$U_b(\frac{n}{4}) = 3U_b\left(\frac{n}{16}\right) + \frac{n}{8}$$
$$U_b(\frac{n}{16}) = 3U_b\left(\frac{n}{64}\right) + \frac{n}{32}$$
$$U_b(\frac{n}{64}) = 3U_b\left(\frac{n}{256}\right) + \frac{n}{128}$$

We can plug these back into the original to find a pattern:

$$U_b(n) = 3U_b\left(\frac{n}{4}\right) + \frac{n}{2}$$
$$U_b(n) = 3\left(3U_b\left(\frac{n}{16}\right) + \frac{n}{8}\right) + \frac{n}{2} = 9U_b\left(\frac{n}{16}\right) + \frac{7n}{8}$$
$$U_b(n) = 9\left(3U_b\left(\frac{n}{64}\right) + \frac{n}{32}\right) + \frac{7n}{8} = 27U_b\left(\frac{n}{64}\right) + \frac{37n}{32}$$
$$U_b(n) = 27\left(3U_b\left(\frac{n}{256}\right) + \frac{n}{128}\right) + \frac{37n}{32} = 81U_b\left(\frac{n}{256}\right) + \frac{229n}{128}$$

After k times of recurrence: $U_b(n) = 3^k U_b\left(\frac{n}{4^k}\right) + n\left(\sum_{i=1}^{k} 3^{i-1}(\frac{2}{4^i})\right)$ which can be simplified more below:

$$\begin{aligned} U_b(n) &= 3^k U_b\left(\frac{n}{4^k}\right) + n\left(\sum_{i=1}^{k} 3^{i-1}\left(\frac{2}{4^i}\right)\right) \\ &= 3^k U_b\left(\frac{n}{4^k}\right) + 2n\left(\sum_{i=1}^{k}\left(\frac{3^{i-1}}{4^i}\right)\right) \\ &= 3^k U_b\left(\frac{n}{4^k}\right) + \frac{2n}{3}\left(\sum_{i=1}^{k}\frac{3^i}{4^i}\right) \\ &= 3^k U_b\left(\frac{n}{4^k}\right) + \frac{2n}{3}\left(\sum_{i=1}^{k}\left(\frac{3}{4}\right)^i\right) \end{aligned}$$

This is in the format of a geometric series, which is defined as $\sum_{k=1}^{n} r^k = \frac{r(1-r^n)}{1-r}$. My sum will then equate to $\frac{3}{4}\frac{1-(3/4)^k}{1-3/4} \Rightarrow 3(1-(3/4)^k)$. We can plug this back into our equation to get $U_b(n) = 3^k U_b\left(\frac{n}{4^k}\right) + 2n(1-(3/4)^k)$.

We terminate recursion when $\frac{n}{4^k} = 3$ for the base case. Solving for k, we get that $k = log_4(n/3)$. Plug this back in to get:

$$\begin{aligned} U_b(n) &= 3^{log_4(n/3)} \cdot 0 + 2n(1-(3/4)^{log_4(n/3)} \text{ here we multiply by 0 to account for the base case.} \\ &= 2n(1-(n/3)^{log_4(3/4)}) \\ &= 2n - 2\cdot 3^{log_4(4/3)} n^{log_4(3/4)+1} \\ &= 2n - 2\cdot 3^{log_4(4/3)} n^{log_4(3)} \\ &= 2n - 2.51n^{0.792} \end{aligned}$$

(continued on next page)

Name: Vera Duong
ID: 109431166

Due February 12, 2021

CSCI 3104, Algorithms **Spring 2021, CU-Boulder**

Problem Set 4 (50 points) Collaborators: Nathan Straub, Matt Hartnett

Taking $f(n) = 2n - 2.51n^{0.792}$ and $g(n) = n$, we can do a limit comparison test:

$$\begin{aligned}
\lim_{n\to\infty} \frac{2n - 2.51n^{0.792}}{n} &\Rightarrow \lim_{n\to\infty} \frac{2n}{n}, \lim_{n\to\infty} \frac{-2.51n^{0.792}}{n} \\
&\Rightarrow \lim_{n\to\infty} \frac{2n}{n} = 2 \\
&\Rightarrow \lim_{n\to\infty} -2.51n^{0.792} \cdot n^{-1} \\
&\Rightarrow \lim_{n\to\infty} -2.51n^{0.792-1} \\
&\Rightarrow \lim_{n\to\infty} -2.51n^{-0.208} \\
&\Rightarrow \lim_{n\to\infty} \frac{-2.51}{n^{0.208}} = 0 \\
&= 2 + 0 = 2
\end{aligned}$$

We can see here that $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ is a constant value, so $U_b(n)$ is $\mathbf{\Theta(n)}$

Name: Vera Duong
ID: 109431166

Due February 12, 2021

CSCI 3104, Algorithms **Spring 2021, CU-Boulder**

Problem Set 4 (50 points) Collaborators: Nathan Straub, Matt Hartnett

2. Consider this recurrence:

$$T(n) = \begin{cases} 4T(n/3) + 2n & \text{when } n > 1, \\ 1 & \text{when } n = 1. \end{cases}$$

(a) How many levels will the recurrence tree have?

(b) What is the cost at the level below the root?

(c) What is the cost at the ℓ'th level below the root?

(d) Is the cost constant for each level?

(e) Find the total cost for all levels. *Hint: You may need to use a summation. The Geometric Sum formula may be helpful.*

(f) If $T(n)$ is $\Theta(g(n))$, find $g(n)$.

Solution:

(a) At the i^{th} level the subproblem size will be $\frac{n}{b^i}$. In our case, $b = 3$ so we can substitute this into our equation to be $\frac{n}{b^k} = 1$ where k is our lowest level and where 1 is the subproblem size. We can solve for k and get $n = 3^k \Rightarrow k = log_3 n$. However, our base case is when n=1 and $log_3(1) = 0$, so we need to add an additional 1 to account for the root, hence there are $log_3(n) + 1$ levels in the recurrence tree.

(b)
To find the cost at the level below the root, or the first level, we will first "build" the tree. The root will contain the node with a cost of 2n. T(n) has 4 subproblems, thus there will be 4 branches in the first level from the root. Each of these branches will contain a cost of $2(n/3)$ since the size of the subproblem is $\frac{n}{3}$. We will then add these 4 up, $\frac{2n}{3} + \frac{2n}{3} + \frac{2n}{3} + \frac{2n}{3} = \frac{8n}{3}$. Thus, the cost at the first level below the root is $8(n/3)$.

(c)
Starting at the first node, we have cost of 2n. The first level will have 4 branches since the number of subproblems is 4, and each of those 4 branches will have a cost of $2(n/3)$. The second level will have 16 branches from 4^2, and each of those branches will have a cost of $32(n/9)$ where we found 9 by doing $n/3^2$. The third level will have 64 branches each with a cost of $128(n/27)$. We can see that the denominator increments by $n/3^\ell$ and the numerator increments by 4^ℓ and the whole thing is multiplied by 2n. Thus, the pattern continues and we will have a cost of $2n(4/3)^\ell$ at the ℓ^{th} level below the root.

(d) The cost is not constant for each level. This is due to the subproblems growing a lot slower than the cost of each subproblem as the levels go down and because it's increasing in each level.

(e)
$k = log_3(n) + 1$, levels of recurrence of the tree.
This is in the format of a geometric series, which is defined as $\sum_{k=1}^{n} r^k = \frac{r(1-r^n)}{1-r}$.

$$\begin{aligned} \text{Total Cost} &= 2n \sum_{i=1}^{k} (4/3)^i \\ &= 2n(4/3) \cdot \frac{1 - (4/3)^k}{1 - (4/3)} \\ &= 2n(4/3) \cdot \frac{1 - (4/3)^k}{-1/3} \\ &= -8n(1 - (4/3)^{log_3(n)+1}) \\ &= -8n(1 - (4/3)(n)^{log_3(4/3)} \\ &= \frac{32}{3} n^{log_3(4/3)+1} - 8n \end{aligned}$$

Name: Vera Duong
ID: 109431166

(f) We can see that $n^{log_3(4/3)+1}$ is our largest term. We will do a limit comparison test where $f(n) = \frac{32}{3}n^{log_3(4/3)+1} - 8n$ and $g(n) = n^{log_3(4/3)+1}$ to calculate $lim_{n\to\infty}\frac{f(n)}{g(n)}$.

$$\begin{aligned}
\lim_{n\to\infty}\frac{\frac{32}{3}n^{log_3(4/3)+1} - 8n}{n^{log_3(4/3)+1}} &\Rightarrow \lim_{n\to\infty}\frac{(32/3)n^{log_3(4/3)+1}}{n^{log_3(4/3)+1}} \text{ and } \lim_{n\to\infty}\frac{-8n}{n^{log_3(4/3)+1}} \\
&\Rightarrow \lim_{n\to\infty}\frac{(32/3)n^{log_3(4/3)+1}}{n^{log_3(4/3)+1}} = 32/3 \\
&\Rightarrow \lim_{n\to\infty}\frac{-8n}{n^{log_3(4/3)+1}} \\
&\Rightarrow \lim_{n\to\infty}\frac{-8}{n^{log_3(4/3)+1-1}} = 0 \\
&= (32/3) + 0 \\
&= 32/3
\end{aligned}$$

We can see here that $\lim_{n\to\infty}\frac{f(n)}{g(n)}$ is a constant value, so $f(n)$ is $\mathbf{\Theta(g(n))}$ where $g(n) = n^{log_3(4/3)+1}$.

Name: Vera Duong
ID: 109431166

Due February 12, 2021

CSCI 3104, Algorithms **Spring 2021, CU-Boulder**

Problem Set 4 (50 points) Collaborators: Nathan Straub, Matt Hartnett

3. Showing your work for relevant comparisons, for the following recurrence relations apply the **master method** to identify whether original problems or subproblems dominate, or whether they are comparable. Then write down a Θ bound.

(a) $M_a(n) = \begin{cases} 2M_a(n/3.14) + n\log(n) & \text{when } n > 0.001, \\ 1337 & \text{otherwise.} \end{cases}$

(b) $M_b(n) = \begin{cases} 6M_b(n/2) + n^{7/3}\log(n) & \text{when } n > 2^{273}, \\ 6734 & \text{otherwise.} \end{cases}$

(c) $M_c(n) = \begin{cases} 9M_c(n/3) + n^3\log(n) & \text{when } n > 8/3, \\ 86 & \text{otherwise.} \end{cases}$

Solution:

(a) First we will need to find our values of a and b. $a = 2$ and $b = 3.14$. Next, our $\log_b(a) = \log_{3.14} 2 = 0.606$. We will next need to compare this value to $f(n) = n\log(n)$ where $n\log(n)$ is $\Omega(n^{0.606+\epsilon})$. ϵ is some constant $\epsilon = 0.394$ to get $n^{0.606+\epsilon} = n$. We will then need to use a limit comparison test where $f(n) = n\log n$ and $g(n) = n^{0.606+0.394}$. This is shown below:

$$\begin{aligned} \lim_{n\to\infty} \frac{f(n)}{g(n)} &= \frac{n\log n}{n^{0.606+0.394}} \\ &= \lim_{n\to\infty} \frac{n\log n}{n} \\ &= \lim_{n\to\infty} \frac{n\frac{\ln(n)}{\ln(10)}}{n} \\ &= \frac{1}{\ln(10)} \lim_{n\to\infty} \frac{n\ln(n)}{n} \\ &= \frac{1}{\ln(10)} \lim_{n\to\infty} \ln(n) \\ &= \infty \end{aligned}$$

Since the limit test shows that it is evaluated to ∞, we know that $n\log n$ is $\Omega(n^{0.606+\epsilon})$. Next, since this is case 3 of the Master Method, we need to show that if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $M_a(n) = \Theta(f(n))$.
In our case, we know that $a = 2$ and $b = 3.14$. So,

$$\begin{aligned} af(n/b) &= 2(n/3.14 \cdot \log\left(\frac{n}{3.14}\right)) \\ &= (2/3.14)n \cdot \log\left(\frac{n}{3.14}\right)) \end{aligned}$$

Here we have $(\frac{2}{3.14})$, so we can choose a value c that's larger but still less than 1. Therefore, $(2/3.14)n \cdot \log\left(\frac{n}{3.14}\right)) \leq cn\log n$ and all sufficiently large n. Thus, by using the Master Method, it's proven that $M_a(n)$ is $\Theta(n\log n)$. Cost is dominated by the root.

Name: Vera Duong
ID: 109431166

(b) First we will need to find our values of a and b. $a = 6$ and $b = 2$. Next, our $\log_b a = \log_2 6 = 2.58$. We will next need to compare this value to $f(n) = n^{7/3} \log n$ where $n^{7/3} \log n$ is $\mathcal{O}(n^{2.58-\epsilon})$. ϵ is some constant $\epsilon = 0.05$ to get $n^{2.58-\epsilon} = n^{2.53}$. We will then need to use a limit comparison test where $f(n) = n^{7/3} \log n$ and $g(n) = n^{2.53}$. This is shown below:

$$\begin{aligned}
\lim_{n\to\infty} \frac{f(n)}{g(n)} &= \lim_{n\to\infty} \frac{n^{7/3} \log n}{n^{2.53}} \\
&= \lim_{n\to\infty} \frac{n^{2.33} \log n}{n^{2.53}} \\
&= \lim_{n\to\infty} \frac{\log n}{n^{0.2}} \\
&= \lim_{n\to\infty} \frac{\frac{\ln(n)}{\ln(10)}}{n^{0.2}} \\
&= \frac{1}{\ln(10)} \lim_{n\to\infty} \frac{\ln(n)}{n^{0.2}} \\
&= \textit{using L'H } \frac{1}{\ln(10)} \lim_{n\to\infty} \frac{1/n}{0.2n^{0.2-1}} \\
&= \frac{1}{\ln(10)} \lim_{n\to\infty} \frac{1/n}{0.2/n^{0.8}} \\
&= \frac{1}{\ln(10)} \lim_{n\to\infty} \frac{5}{n^{0.2}} \\
&= 0
\end{aligned}$$

We can see here that the limit comparison test goes to zero. This means that f(n) is polynomially smaller than the number of leaves, so asymptotically, work done at the leaves dominates. Thus, $M_b(n)$ is $\Theta(n^{\log_2 6})$ or $\Theta(n^{2.58})$.

Name: Vera Duong
ID: 109431166

(c) First we will need to find our values of a and b. $a = 9$ and $b = 3$. Next, our $\log_b a = \log_3 9 = 2$. We will next need to compare this value to $f(n) = n^3 \log n$ where $n^3 \log n$ is $\Omega(n^{2+\epsilon})$. ϵ is some constant $\epsilon = 1$ to get $n^{2+\epsilon} = n^3$. We will then need to use a limit comparison test where $f(n) = n^3 \log n$ and $g(n) = n^3$. This is shown below:

$$\begin{aligned}\lim_{n\to\infty} \frac{f(n)}{g(n)} &= \lim_{n\to\infty} \frac{n^3 \log n}{n^3} \\ &= \lim_{n\to\infty} \log n \\ &= \infty\end{aligned}$$

Since the limit test shows that it is evaluated to ∞, we know that $n^3 \log n$ is $\Omega(n^3)$. Next, since this is case 3 of the Master Method, we need to show that if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $M_a(n) = \Theta(f(n))$.
In our case, we know that $a = 9$ and $b = 3$. So,

$$af(n/b) = 9((n/3)^3 \log (n/3))$$

We can choose a value c that's less than 1 for $9((n/3)^3 \log (n/3)) \leq cn^3 \log n$ and all sufficiently large n. Thus, by using the Master Method, it's proven that $M_c(n)$ is $\Theta(n^3 \log n)$. Cost is dominated by the root.

Name: Vera Duong
ID: 109431166

Due February 12, 2021

CSCI 3104, Algorithms **Spring 2021, CU-Boulder**

Problem Set 4 (50 points) Collaborators: Nathan Straub, Matt Hartnett

4. This is a coding problem. You will implement a version of Quicksort.

   - **You must submit a Python 3 source code file with a `quicksort` and a `partitionInPlace` function as specified below.** You will not receive credit if we cannot call your functions.
   - The `quicksort` function should take as input an array (numpy array), and for large enough arrays pick a pivot value, call your partition function based on that pivot value, and then recursively call quicksort on resulting partitions that are strictly smaller in size than the input array in order to sort the input.
     - Additionally, your quicksort should transition from recursive calls to "manual" sorting (via `if` statements or equivalent) when the arrays become small enough.
   - The `partitionInPlace` function should take as input an array (numpy array) and pivot value, partition the array (*in at most linear amount of work and constant amount of space*), and return an index such that (after returning) no further swaps need to occur between elements below and elements above the index in order for the array to be sorted.
   - You are provided with a scaffold python file that you may use, which contains some suggested function behavior and loop invariants, as well as a simple testing driver. You may alter anything within or ignore it altogether **so long as you maintain the function prototypes specified above**.
     - In particular, the suggestions are meant to allow the pivot value to not be in the array, which is NOT a requirement for Quicksort.

Solution: