Just going to write answers on this page & work is later

CMPS 2200 Assignment 2

Name: Mackenzie Bookamer

In this assignment we'll work on applying the methods we've learned to analyze recurrences, and also see their behavior in practice. As with previous assignments, some of of your answers will go in `main.py`.. You should feel free to edit this file with your answers; for handwritten work please scan your work and submit a PDF titled `assignment-02.pdf` and push to your github repository.

1. Derive asymptotic upper bounds of work for each recurrence below.

- $W(n) = 2W(n/3) + 1$.

  $O(n^{\log_3 2})$

- $W(n) = 5W(n/4) + n$.

  $O(n^{\log_4 5})$

- $W(n) = 7W(n/7) + n$.

  $O(n \log n)$

- $W(n) = 9W(n/3) + n^2$.

  $O(n^2 \log n)$

- $W(n) = 8W(n/2) + n^3$.

  $O(n^3 \log n)$

- $W(n) = 49W(n/25) + n^{3/2} \log n$.

  $O(n^{3/2} \log n)$

see later pages for work!!

- $W(n) = W(n-1) + 2$.

  .
  . $O(n)$
  .
  .

- $W(n) = W(n-1) + n^c$, with $c \geq 1$.

  .
  . $O(n^{c+1})$
  .
  .

- $W(n) = W(\sqrt{n}) + 1$ $O(\log(\log n))$

2. Suppose that for a given task you are choosing between the following three algorithms:

- Algorithm $\mathcal{A}$ solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.

- Algorithm $\mathcal{B}$ solves problems of size n by recursively solving two subproblems of size $n-1$ and then combining the solutions in constant time.

- Algorithm $\mathcal{C}$ solves problems of size n by dividing them into nine subproblems of size $n/3$, recursively solving each subproblem, and then combining the solutions in $O(n^2)$ time.

  What are the asymptotic running times of each of these algorithms? Which algorithm would you choose? Algorithm C

work is on later page

3. Now that you have some practice solving recurrences, let's work on implementing some algorithms. In lecture we discussed a divide and conquer algorithm for integer multiplication. This algorithm takes as input two n-bit strings $x = \langle x_L, x_R \rangle$ and $y = \langle y_L, y_R \rangle$ and computes the product xy by using the fact that $xy = 2^{n/2} x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$. Use the stub functions in `main.py` to implement Karatsaba-Ofman algorithm algorithm for integer multiplication: a divide and conquer algorithm that runs in subquadratic time. Then test the empirical running times across a variety of inputs to test whether your code scales in the manner described by the asymptotic runtime. Please refer to Recitation 3 for some basic implementations, and Eqs (7) and (8) in the slides https://github.com/allan-tulane/cmps2200-slides/blob/main/module-02-recurrences/recurrences-integer-multiplication.ipynb

Asymptotic Upper Bounds

- $W(n) = 2W(n/3) + 1$
  - Root: $W(0) = 1$
  - Level 1: $W(1) = 1 + 1 = 2$
  - Level 2: $W(2) = 1 + 1 + 2 = 4$

So this is leaf dominated

- 2^i nodes @ every level i
- height: $\log_3 2$

When $i = \log_3 2$, we get the work done at the leaf level $\Rightarrow$ $O(n^{\log_3 2})$

- $W(n) = 5W(n/4) + n$
  - Root: $W(0) = n$
  - Level 1: $W(1) = \frac{5n}{4}$

Geometric, leaf dominated like previous $\Rightarrow$ $O(n^{\log_4 5})$

- $W(n) = 7\,W(n/7) + n$

These #s are the same, so the tree is balanced

- Root: $W(0) = n$
- Level 1: $W(1) = n$

- # of levels: $\log n$
- work done @ worst level: n

$\Rightarrow$ $O(n \log n)$

- $W(n) = 9\,W(n/3) + n^2$

- Root: $W(0) = n^2$
- Level 1: $W(1) = 9 \cdot \frac{n^2}{9} = n^2$

so, this is balanced!

- # of levels: $\log n$
- work @ worst level: n^2

$\Rightarrow$ $O(n^2 \log n)$

- $W(n) = 8W(n/2) + n^3$
  - root: $W(0) = n^3$
  - Level 1: $W(1) = 8 \cdot \frac{n^3}{8} = n^3$

so, this is balanced

- # of levels: $\log n$
- work @ worst level: n^3

$\Rightarrow$ $O(n^3 \log n)$

- $W(n) = 49W(n/25) + n^{3/2}\log n$
  - root: $W(0) = n^{3/2}\log n$
  - Level 1:

$$W(1) = 49\left(\frac{n^{3/2}}{125}\right)\log\frac{n}{25} =$$

$$\frac{49}{125}n^{3/2}\log n - \frac{49}{125}n^{3/2}\log 25 \leq \frac{49}{125}n^{3/2}\log n$$

we know $\frac{49}{125} < 1$, so this recurrence is not dominated

$\Rightarrow \boxed{O(n^{3/2} \log n)}$

- $W(n) = W(n-1) + 2$
  - Root: $W(0) = 2$
  - Level 1: $W(1) = 2$

so, this is balanced

- # levels: n
- work @ worst level: 2, but this is a constant so it's not needed in our final answer

$\Rightarrow \boxed{O(n)}$

- $W(n) = W(n-1) + n^c$
  - ROOT: $W(0) = n^c$
  - Level 1: $W(1) = (n-1)^c \in O(n^c)$

- $c=1$: $W(n) = W(n-1) + n \rightarrow$ balanced
- $c=2$: $W(n) = W(n-1) + n^2 \rightarrow$ balanced

- $c = 3 : W(n) = W(n-1) + n^3 \rightarrow$ balanced

so, our original $W(n)$ is balanced

- # levels : n
- work @ worst level : n^c

$\Rightarrow O(n \cdot n^c) = \boxed{O(n^{c+1})}$

- $W(n) = W(\sqrt{n}) + 1$

we know this is balanced because each node has cost 1, so we have our work at worst level but now we need our # of levels

- Decay of n: n, $n^{1/2}$, $n^{1/4}$, $n^{1/8}$, $\vdots$, $n^{1/2^i}$, i = level

we only want 1 node per level,
so when $n=2$, we are done.

$$n^{\frac{1}{2^i}} = 2 \quad | \log$$

$$\log n^{\frac{1}{2^y}} = \log 2$$

$$\frac{1}{2^y} \log n = 1$$

$$\log n = 2^y \quad | \log$$

$$\log(\log n) = y = \#\ \text{levels}$$

$$\Rightarrow O(1 \cdot \log(\log n)) = \boxed{O(\log(\log n))}$$

Question 3

Let's write recurrences for each algorithm & find asy. upper bounds

- $A(n) = 5A(n/2) + n$
  - root : n
  - Level 1 : $\frac{5}{2}n$

so, this is leaf dominated

$\Rightarrow$ $\boxed{O(n^{\log_2 5})}$

- $B(n) = 2B(n-1) + 1$
  - root : 1
  - Level 1 : $1+1=2$
  - Level 2 : $1+1+2=4$

} 2^i series

so, this is leaf dominated

$\Rightarrow$ $\boxed{O(2^n)}$

- $C(n) = 9C(n/3) + n^2$
  - Root: n^2
  - Level 1: $9 \cdot \frac{n^2}{9} = n^2$

So, this is balanced!
- # levels: $\log_3 n$
- work @ worst level: n^2

$\Rightarrow$ $O(n^2 \log_3 n)$

Let's observe that Algorithm C is essentially running in polynomial time because $\log_3 n < n$. Algorithms A and B are exponential and will always be greater than C, so we should choose

Algorithm C.