COSC 3020 Name: Jacob Tuttle
Algorithms

Homework 1

October 10, 2019

1. Asymptotic Complexity: Let S be some arbitrary set such that $S \in O(log_2(n))$.

$S(n) \in O(log_2(n)) \implies S(n) \leq c \cdot log_2(n)$ for some arbitrary c.

Now, let c be some value such that both $c \cdot log_2(n)$ and $c \cdot log_{10}(n)$ are greater than or equal to $S(n)$.

$S(n) \leq c \cdot log_{10}(n)$ as a result of our definition of c, and $S(n) \leq c \cdot log_{10}(n) \implies S(n) \in O(log_{10}(n))$.

Now consider $T(n)$ such that $T(n) \in O(log_{10}(n))$.

$T(n) \in O(log_{10}(n)) \implies T(n) \leq c \cdot log10(n)$

Using the same arbitrarily large c as above, we can show that:

$$\begin{aligned} T(n) &\leq c \cdot log_2(n) \\ &\implies T(n) \in O(log_2(n)) \end{aligned}$$

Since $T(n) = c \cdot log_{10}(n) = S(n)$

$$T(n) = S(n) \implies O(log_{10}(n)) = O(log_2(n))$$

2. Runtime Analysis: Before being able to perform a runtime analysis evaluation, first the following code must be analyzed for complexity.

```
if (n <= 1) return;                                   (1)
mystery(n/3)                                          (2)
for (var i = 0; i < n*n; i++) { count = count + 1; }  (3)
```

These equations have runtimes as presented in the table below.

Equation	Time Complexity
1	1
2	$T(\frac{n}{3})$
3	n^2

Using these analyzed times, we can then constuct a piecewise function to define the time complexity of the function `mystery` in terms of the input size n.

$$T(n) = \begin{cases} 1 & n \leq 1 \\ 2T(\frac{n}{3}) + n^2 & n > 1 \end{cases}$$

From there, we are able to trace the recurrence relation step by step for some arbitrarily large n.

$$\begin{aligned} T(n) &= T(1) + 2T(\frac{n}{3} + n^2) \\ &= 1 + 2(T(1) + 2T(\frac{n}{9}) + \frac{n}{3}^2) + n^2 \\ &= 3 + 4T(\frac{n}{9}) + \frac{2n^2}{9} + n^2 \\ &= 3 + 4T(\frac{n}{9}) + \frac{11n^2}{9} \\ &= 3 + 4(T(1) + 2T(\frac{n}{27}) + \frac{n}{9}^2) + \frac{11n^2}{9} \\ &= 7 + 8T(\frac{n}{27}) + \frac{103n^2}{81} \\ &= \vdots \\ &= (2^i - 1) + 2i \cdot T(\frac{n}{3^i}) + \lambda_{i-1} n^2 \end{aligned}$$

This approximation for the runtime of the ith recurence of the loop makes use of an equation[1] (4) for the fraction multiplied with n^2, indicated by λ_{i-1} above. The m present in the equation is the iteration number i minus 1 since the equation uses an input of 0 to produce its first value.

$$\lambda_m = \frac{\frac{-2^{m+1}}{7} + \frac{9^{m+1}}{7}}{9^m} = \frac{-2^{m+1} + 9^{m+1}}{7 \cdot 9^m} \tag{4}$$

Since this function, for some value of n will split it up into thirds, compounding the factor each time, (i.e. the first split is $\frac{1}{3}$, second $\frac{1}{9}$, etc.) the total number of executions will be $i = log_3(n)$. Substituting this value for i provides a recurence relation of:

[1]This formula for part of the fractional was retrieved from the Online Encyclopedia of Integer Sequences, http://oeis.org/A016133. Paolo P. Lava, June 16, 2008

$$\begin{aligned}T(n) &= (2^{log_3(n)} - 1) + 2 \cdot log_3(n) \cdot T(\frac{n}{3^{log_3(n)}}) + \lambda_{log_3(n)-1} n^2 \\ &= (2^{log_3(n)} - 1) + 2 \cdot log_3(n) \cdot T(\frac{n}{n}) + \lambda_{log_3(n)-1} n^2 \\ &= (2^{log_3(n)} - 1) + 2 \cdot log_3(n) \cdot 1 + \lambda_{log_3(n)-1} n^2 \\ &= (2^{log_3(n)} - 1) + 2 \cdot log_3(n) + \lambda_{log_3(n)-1} n^2\end{aligned}$$

Of the three families of functions represented ($2^{log_3(n)}$, $log_3(n)$, n^2), the function with the most rapid growth is n^2. Disregarding the constants represented by λ, we arrive at the conclusion that $T(n) \in O(n^2)$

3. Sorting — Insertion Sort:

Array at the beginning of Insertion Sort:

1	7	1	5	3	-1	9
1	7	1	5	3	-1	9
1	1	7	5	3	-1	9
1	1	5	7	3	-1	9
1	1	3	5	7	-1	9
-1	1	1	3	5	7	9
-1	1	1	3	5	7	9

End of sorting

4. Sorting — Merge Sort:

Code solutions to the recursive, in-place merge sort problem provided in `mergeSort.js`.

Test code provided in `mergeSortTest.js`

Here is my bullshit analysis

5. Sorting — Quicksort:

test