CMPE 300 Homework 2

Ibrahim Ozgurcan OZTAS

20/11/2019

1 Question 1:

In Question 1, the answers are below.
a) The best case input for this algorithm that executes the least amount of basic operations is a sorted list which is in descending order. In the best case, the algorithm iterates over the first for loop block, n-1 times. In this block, there are two comparisons, which are the FindStack function that compares the top element of a stack with the search value and the max(i,j) function that compares the two inputs, i and j, and returns the greater one. Inside the body of the first for loop, there are 2 basic operations total, hence by iterating (n-1) times, total number of basic operations is $2 \cdot (n-1)$.

In addition to the first part, the second for loop block checks the stack list for each index to be ensured that whether the stacks are empty or not. To do so, there is one if statement which consists of Empty(Stack s) function that checks the input stack s is empty or not. This is the first comparison and it does n comparison in n iteration. Also, the complete list is in the first stack, thus the remaining stacks are empty and the while block checks k¡s comparison n-1 times.

Overall, the total comparison is $2 \cdot (n-1) + n + (n-1) = 4 \cdot n + 3$

Hence the $B(n) \in \theta(n)$

b) The worst case input for this algorithm that executes the most amount of basic operations is a list that is the result of the merge of two sublists that are L1[$n/2$:$n-1$] and L2[0:$n/2-1$], respectively.

FindStack executes $\log_2(n)$ operation per iteration, thus $(n-1) \cdot \log(n)$ basic operation executes throughout the first for loop.

In second for loop, the for loop iterates n items and there is $n/2$ stacks and for each while iteration, there are 2 comparisons, thus there are $2 \cdot (n/2) \cdot (n/2-1)$ basic operations executed.

In total, there are $(n-1) \cdot \log_2 n + (n^2/2) - n$ basic operations.

Hence, the $W(n) \in \theta(n^2)$

2 Question 2:

In Question 2, the recurrence relation for the worst case input n is;
$W(n) = W(\sqrt{n}) + 2\sqrt{n}$

The RootStack function has one recursive call and a while loop that iterates from $n - \sqrt{n}$ to 0. The beginning of the function body includes one comparison as $n = 2$ in if part. The amount of iteration of the while loop is $\sqrt{n} - 1$ and it has 2 comparison operations per iteration. Hence, $2 \cdot (\sqrt{n} - 1)$ comparisons are done in $\sqrt{n} - 1$ iterations, therefore the current value of b is 0. In the next iteration, there is one more comparison which is the evaluation of b and 0, results in which the comparison returns false, and breaks the while loop done by the processor. Finally, the total comparison done in the function is $2 \cdot (\sqrt{n} - 1) + 1 + 1 = 2 \cdot \sqrt{n}$.

The recursive call inside the function will be called with the list that has input size $\sqrt{n}$. Thus, the recurrence relation can be written as;

$W(n) = W(\sqrt{n}) + 2\sqrt{n}$

3 Question 3:

In Question 3, the answers are below.
a) For this question, the characteristic equation of the recurrence relation can be written as $\alpha^2 = a\alpha + b$. Therefore, the current characteristic equation is $\alpha^2 = 4\alpha - 3$. The roots are $\alpha_1 = 3$ and $\alpha_2 = 1$. Hence, the function is $T(n) = c_1 3^n + c_2$, since $1^n = 1$ for all integer $n \in Z^+$. Then, checking the initial conditions sets the constants c_1 and c_2 to 4 and 5 respectively.

- $e_1 \leftarrow c_1 + c_2 = 9$
  $e_2 \leftarrow 3c_1 + c_2 = 17$
  Substituting e_2 from $3e_1$ results in
  $3e_1 - e_2 = 3c_1 + 3c_2 - 3c_1 - c_2 = 27 - 17 \Longrightarrow 2c_2 = 10$,
  which implies $\Longrightarrow c_2 = 5$.

In conclusion, the function can be written in closed form as

$T(n) = 4 \cdot 3^n + 5$

b) For this question, the closed form of the recurrence relation can be written as $T(n) = [\sum_{i=0}^{n-1} (\frac{1}{2})^i] + 10 \cdot \frac{1}{2^n}$

– By backward substitution method,

$T(n) = T(n-1) \cdot \frac{1}{2} + 1$
$T(n-1) = T(n-2) \cdot \frac{1}{2} + 1$
$T(n-2) = T(n-3) \cdot \frac{1}{2} + 1$
It continues until $T(0)$ is reached. (After n-1 iterations)
$T(1) = T(0) \cdot \frac{1}{2} + 1$

Swapping every equation from $T(1)$ to $T(n-1)$ with the corresponding equation will result in;
$T(n) = T(0) \cdot (\frac{1}{2})^n + 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ... + \frac{1}{2^{(n-1)}}$

Plug $T(0)$ into the equation;
$T(n) = 10 \cdot (\frac{1}{2})^n + 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ... + \frac{1}{2^{(n-1)}}$

In the right side of the equation, there is the series of $x = \frac{1}{2}$ from power 0 to $(n-1)$. Hence, it can be written as
$T(0) = 10 \cdot (\frac{1}{2})^n + \sum_{i=1}^{n-1} (\frac{1}{2})^i$ (Check the geometric series from the literature.)

The summation will result in $\frac{(\frac{1}{2})^n - 1}{\frac{1}{2} - 1}$
Hence, the function $T(n) = 2 + 8 \cdot (\frac{1}{2})^n$

c) For this question, the closed form of the recurrence relation can be written as $T(n) = \prod_{i=0}^{k-1} (\frac{n}{2^i})$, where $k = \log_2 n$, since the assumption allows one to say $n = 2^k$.

Hence, $T(n) = \frac{n^k}{2^{\frac{k(k-1)}{2}}}$, where $k = \log_2 n$.

It can be refined into $T(n) = (\frac{n}{2^{\frac{(k-1)}{2}}})^k$, where $k = \log_2 n$.

The improved form of the function is $T(n) = (\frac{n}{2^{\frac{(\log_2 n - 1)}{2}}})^{\log_2 n}$

The final form of the function is $T(n) = n^{\frac{1+\log_2 n}{2}}$

– By backward substition method,

$T(n) = n \cdot T(n/2)$
$T(n/2) = \frac{n}{2} \cdot T(n/4)$
$T(n/4) = \frac{n}{4} \cdot T(n/8)$
It continues until $T(1)$ is reached. (After $k \leftarrow \log_2 n$ iterations.)
$T(2) = 2 \cdot T(1)$

Swapping every equation from $T(1)$ to $T(n-1)$ with the corresponding equation will result in;
$T(n) = T(1) \cdot n \cdot \frac{n}{2} \cdot \frac{n}{4} \cdot ... \cdot 2$

Plug $T(1)$ into the equation;

$T(n) = 1 \cdot n \cdot \frac{n}{2} \cdot \frac{n}{4} \cdot ... \cdot 2$

The equation can be refined as;
$T(n) = \prod_{i=0}^{k-1} (\frac{n}{2^i})$
Hence, $T(n) = \frac{n^k}{2^{(\sum_{i=0}^{k-1} i)}}$

$T(n) = \frac{n^k}{2^{\frac{k(k-1)}{2}}}$, where $k = \log_2 n$.
Improved form of the refined function is;
$T(n) = (\frac{n}{2^{\frac{(k-1)}{2}}})^k$, where $k = \log_2 n$
Replacing k in the previous function results in;

$T(n) = (\frac{n}{2^{\frac{(\log_2 n-1)}{2}}})^{\log_2 n}$, which implies to $T(n) = n^{\frac{1+\log_2 n}{2}}$