CS 3610 - Assignment 1 - Jonathon Meney - 348074

Jonathon Meney

September 2023

1. (3 marks) Fill in the following table to compute the total number of steps (time complexity):

Statement	s/e	Frequency	Total Steps
Algorithm Test (a, x, b, y)	0	-	0
//array a of size x; array b of size y	0	-	0
{	0	-	0
result := 1.0;	1	1	1
for i :=1 to x do	1	$x+1$	$x+1$
{	0	-	0
a[i] = 2 * a[i];	1	x	x
result := result * a[i];	1	x	x
for j := 1 to y do	1	$x(y+1)$	$xy+x$
{	0	-	0
result := result * b[j];	1	xy	xy
for k := 1 to xy do	1	$xy(xy+1)$	x^2y^2+xy
}	0	-	0
result := result + (i*x) +(j*y);	1	$xy(xy)$	x^2y^2
}	0	-	0
}	0	-	0
}	0	-	0
return result;	1	1	1
}	0	-	0
Total			$2x^2y^2+3xy+4x+3$

2. (3 marks) Given a function $f(n) = 10n^2 + 4n + 2$, compute a function $g(n)$ such that $f(n) \leq c * g(n)$ for $n \leq 5$ and a positive constant c. Please note that $f(n) = O(g(n))$ if and only if the equation is satisfied.

   The function $g(n) = n^2$ and the constant $c = 11$ satisfy this equation since, $10n^2 + 4n + 2 \leq 11n^2$ for all $n \leq 5$

   (a) What would be the Big-Oh representation of the given function f(n).

      The Big-Oh representation $f(n) = 10n2 + 4n + 2$ is $O(n^2)$ since n^2 is the fastest growing term in $f(n)$ and $g(n) = n^2$.

   (b) What minimum possible value of constant c satisfies the equation.

      The minimum possible value of constant c is 11.

3. (5 marks = 3+2) Solve the following recurrence relation. Also, prove that $T(n)$ is $O(3^n)$.

$$T(n) = \begin{cases} 2 & \text{if } n = 0 \\ 3T(n-1) + 2 & \text{if } n > 0 \end{cases}$$

$$\begin{aligned}
T(n) &= 3T(n-1) + 2 \\
&= 3[3T(n-2) + 2] + 2 \\
&= 3(3)(T(n-2)) + 2(3) + 2 \\
&= 3(3)(3T(n-3) + 2) + 2(3) + 2 \\
&= 3(3)(3)T(n-3) + 2(3)(3) + 2(3) + 2 \\
&= \ldots \\
&= 3^n T(0) + 2(3^{n-1}) + \ldots + 2(3^2) + 2(3^1) + 2(3^0) \\
&= 2(3^n) + 2(3^{n-1}) + \ldots + 2(3^2) + 2(3^1) + 2(3^0) \\
&= \sum_{k=0}^{n} 2(3)^k = 3^{n+1} - 1
\end{aligned}$$

$T(n)$ is $O(3^n)$ because $3 * 3^n - 1 \leq 3 * 3^n$ using $g(n) = 3^n$.

4. (3 marks) Fill in the following table to compute the total number of steps (time complexity). Show the recurrence relationship that represents the time complexity of the given algorithm.

Statement	s/e	Frequency		Total Steps	
		$n = 0$	$n > 0$	$n = 0$	$n > 0$
Algorithm RTest (n)	0	-	-	0	0
{	0	-	-	0	0
if (n==0) then	1	1	1	1	1
score := 1**n	1	1	1	1	1
return score;	1	1	1	1	1
else	0	-	-	0	0
score := n;	1	1	1	1	1
return n * RTest(n-1);	$1 + x$	0	1	0	$1 + x$
}	0	-	-	0	0
Total				4	$4 + x$

$$t_{RTest}(n) = \begin{cases} 4 & \text{if } n = 0 \\ t_{RTest}(n-1) + 4 & \text{if } n > 0 \end{cases}$$

5. (6 marks = 3+3) You are given a Binary Search algorithm that divides the given list of elements in half (line 13) and makes two recursive calls (i to $mid - 1$; $mid + 1$ to l) using if-elseif-else statements (lines 14-17). Rewrite the algorithm to incorporate the following functionality:

   (a) You partition the given list such that the two sub lists are of sizes one-third and two-third of the original size respectively.

   See definition of mid below.

   (b) Replace if-elseif-else with if-else i.e. reduce the number of comparisons by one for each call to the given algorithm.

   See below changes. Total comparisons was reduced from 4 to 3.

```
1   Algorithm BinSearch(a, i, l, x)
2   // Given an array a[i : l] of elements in nondecreasing
3   // order, 1 ≤ i ≤ l, determine whether x is present, and
4   // if so, return j such that x = a[j]; else return 0.
5   {
6       mid := i + ⌊(l − i)/3⌋
7
8       if (x = a[mid]) then return mid
9
10      if (i >= l) then return 0
11
12      if (x < a[mid]) then return BinSearch(a, i, mid − 1, x)
13      else return BinSearch(a, mid + 1, l, x)
14  }
```