Mohammad Ali Zahir (40077619)

COMP 352: Data Structures and Algorithms

27/01/2020

Assignment 1

1. Function changeConsecutiveString(st)
   Define an arr[n]
   Define an empty string str=" "

   Initialize int count = 0

   If n =0, then (1)
   arr[n] =""

   else if n=1, then
   str = arr[0]

   else if
   for i ← 1 to n do
      if arr[i] != arr [i-1]
         str = str + arr[i-1]

   if count != 1 (2)
      str = str + count
      end if statement (2)

   if count =1 (3)
            end for loop

else
count++
end if statement (3)

end for loop
end if statement (1)

return str
end function

a) **The time complexity is O(n)**
b) **The space complexity is O(n)**

2.

i. Function arrayDifference(arr[])
Create an array of size n (int a1 = new int[n])

int min ← 0
int max ← 0

for i ← 0 to n -1 do
increment i for each iteration of the for loop

if (Math.abs(a1[i+1] –a1[i]) < Math.abs(a1[min+1] – a1[min]))

min ← i

else if (Math.abs(a1[i+1] –a1[i]) > Math.abs(a1[max+1] – a1[max]))

max ← i

end for loop

print ("The two conductive indices with smallest difference between their values are: index "+min+" and index "+(min+1)+", storing values "+arr[min]+" and "+arr[min+1])

print ("The two conductive indices with largest difference between their values are: index "+max+" and index "+(max+1)+", storing values "+arr[max]+" and "+arr[max+1])

ii. As we go through the for loop, we always get the difference with the next element of the array. If by looping through the array, the maximum difference updates we can update it and store in the max variable instantly. The same applies to the minimum difference.

iii. The time complexity would be O(n).

The pseudocode has one for loop which goes trough 0 to n-1.

Since the other operations like storing the value of max and min can be done with a time complexity of 1.

O(n-1). If we ignore constants, O(n).

iv. The stack growth is the maximum amount of space that the algorithm that the algorithm. The space used for this algorithm is the number of elements that are present in our array. **Since this number cannot change in any circumstance, we can say that the stack growth is constant (space complexity is O(n)).**

3.

a) **$n^{15}\log n + n^9$ is $O(n^9 \log n)$**

$f(n) = n^{15}\log n + n^9$

By assuming, that $n^9\log n$ is the right answer, we should have

$n^{15}\log n \leq n^9\log n$ (Assume base is 10)

For n = 1, this would be 0, hence the inequality suffices for n=1. However, it has to be proven for $n \geq 1$.

Doing n =2: $9864.15 \leq 154.12$

As we can see this is not true, hence this is **disproved.**

b) **$f(n) = 15^7n^5 + 5n^4 + 8000000n^2 + n$ is $\Theta(n^3)$**

Based on the definition of Big-O, and the fact that this is a polynomial.

$O(n^5)$ for $n \geq 1$. This is the worst case that is possible

$\Omega(n)$ for $n \geq 1$. This is the best case possible

This means that: $n \leq f(n) \leq n^5$

Hence, this is not n^3 , and this is **disproved**

c) **$f(n) = n^n$ is $\Omega(n!)$**

For this to be true, $n^n \geq n!$

LHS : n x n x n x ……..n (n times)

RHS: n x (n-1) x (n-2) x (n- (n-1))

For the LHS, we can see that n never decreases. However, for the RHS, n decreases each time by 1 so logically LHS ≥ RHS.

Hence this is **proved.**

d) **$f(n) = 0.01n^9 + 800000n^7$ is $O(n^9)$**

Since this is a polynomial, this is already **proved** in class. However, to prove this, we must do

$0.01n^9 \leq 0.01n^9$ for $n \geq 0$ $(0 \leq 0)$

$800000n^7 \leq 800000n^9$ for $n \geq 0$ $(0 \leq 0)$

If we do, n= 2, for example in the second one.

102 400 000 ≤ 409 600 000.

e) **$f(n) = n^{14} + 0.0000001n^5$ is $\Omega(n^{13})$**

For this to be true, $n^{14} + 0.0000001n^5 \geq n^{13}$

Since obviously the highest degree of coefficient is on the LHS

$n^{14} \geq n^{13}$. Big O is $O(n^{14})$.

By definition, Big – Omega has to be better than Big O. Since n^{13} is better than n^{14} , this is **proved**.

f) **$f(n) = n!$ is $O(3^n)$**

From the definition of Big-O, if we assume the answer is correct,

$n! \leq 3^n$

If we use n = 7, for example:

$5040 \leq 2187$

This is already proven to be false. Hence, this is **disproved**.