L10 - Time Complexity II

Sunday, April 12, 2020 12:11 PM

<u>Time Complexity of recursive Sum of a list of numbers</u>

recSum(Arr, n)

1. if n = 0
2. return 0
3. return recSum(Arr, n-1) + Arr[n]

Recurrence Equation/Relation

$$T_{recSum}(n) = \begin{cases} c1; & if\ n = 0 \\ T_{recSum}(n-1) + c; & if\ n > 0 \end{cases}$$

where c & c1 are positive constants:

Solving for

$$T_{recSum}(n) = T_{recSum}(n-1) + c; \qquad if\ n > 0 \qquad (eq.1)$$

Using equation 1, above, repeatedly,

$T_{recSum}(n) = \underline{T_{recSum}(n - 1)} + c$

Substituting n-1 for n in equation 1

Substituting n-1 for n in eq. 1

$T_{recSum}(n-1) = T_{recSum}(n-1-1) + c; = T_{recSum}(n-2) + c;$

Substituting n-2 for n in eq. 1

$T_{recSum}(n-2) = T_{recSum}(n-2-1) + c; = T_{recSum}(n-3) + c;$

$= [T_{recSum}(n - 2) + c] + c = T_{recSum}(n - 2) + c + c$

$= \underline{T_{recSum}(n - 2)} + 2c$

Substituting n-2 for n in equation 1

$= [T_{recSum}(n - 3) + c] + 2c = T_{recSum}(n - 3) + c + 2c$

$= T_{recSum}(n - 3) + 3c$

...

Let's continue this process of unfolding of recurrence equation/relation for k steps

$= T_{recSum}(n - k) + kc$ (2)

let n - k = 0, therefore k = n
substituting k=n in equation 2

$T_{recSum}(n) = T_{recSum}(n - n) + cn$

$= T_{recSum}(0) + cn$

$T_{recSum}(n) = c1 + cn = cn + c1$

$T_{recSum}(n) = \boldsymbol{O(n)}$

If n= 4, ==> recSum(Arr, 4) -> recSum(Arr, 3) -> recSum(Arr, 2) -> recSum(Arr, 1) -> recSum(Arr, 0)

$$
\begin{aligned}
T_{recSum}(4) &= T_{recSum}(3) + c \\
&= T_{recSum}(2) + c + c \\
&= T_{recSum}(1) + c + c + c \\
&= T_{recSum}(0) + c + c + c + c \\
&= c1 + c + c + c + c
\end{aligned}
$$

If n=4, the recSum function is called 5 times and T_{recSum} is also calculated 5 times.

As such, there is one-to-one correspondence between the number of times recursive code is invoked and the number of steps to calculate the affiliated time complexity. This is important point.

$T_1(n) = \begin{cases} c1; & if\ n = 0 \\ T_1(n-2) + c; & if\ n > 0 \end{cases}$

where c & c1 are positive constants:

Solving for

$$T_1(n) = T_1(n-2) + c; \qquad if\ n > 0 \qquad (eq.3)$$

Using equation 3, above, repeatedly,

$T_1(n) = \underline{T_1(n-2)} + c$

Substituting n-2 for n in equation 3

$= [T1(n-4) + c] + c = T1(n-4) + c + c$

$T_1(n) = \underline{T_1(n-4)} + 2c$

Substituting n-4 for n in equation 3

$= [T_1(n-6) + c] + 2c = T_1(n-6) + c + 2c$

$T_1(n) = \underline{T_1(n-6)} + 3c$

…

$$T_1(n) = T_1(n-k) + (k/2)c \qquad (4)$$

assuming n to be multiple of 2.
let n - k = 0, therefore k = n,
substituting k=n in equation 4

$T_1(n) = T_1(n-n) + (c/2)n$

$= T_1(0) + (c/2)n$

$= c1 + (c/2)n$

$T_1(n) = \boldsymbol{O(n)}$ *Linear Time Complexity*

$$T_m(n) = \begin{cases} c1; & if\ n = 0 \\ T_m(n-1) + c(logn); & if\ n > 0 \end{cases}$$

where c & c1 are positive constants:

Solving for

$$T_m(n) = T_m(n-1) + c(logn); \qquad if\ n > 0 \qquad (eq.5)$$

Using equation 5, above, repeatedly,

$T_m(n) = \underline{T_m(n-1)} + c(\log n)$

Substituting n-1 for n in equation 5, above

$= [T_m(n-2) + c(\log(n-1))] + c(\log n)$

$= \underline{T_m(n-2)} + c(\log(n-1)) + c(\log(n))$

Substituting n-2 for n in equation 5, above $[T_m(n-2) = T_m(n-2-1) + c(\log(n-2))]$;

$= [T_m(n-3) + c(\log(n-2))] + c(\log(n-1)) + c(\log(n))$

$= \underline{T_m(n-3)} + c(\log(n-2)) + c(\log(n-1)) + c(\log(n))$

...

$$= T_m(n-k) + c(\log(n-(k-1))) + c(\log(n-(k-2))) + \dots + c(\log(n-2) + c(\log(n-1)) + c(\log(n)) \qquad (6)$$

let n - k = 0, therefore k = n,
substituting k=n in equation 6

$$T_m(n) = T_m(n-n) + c(\log(n-(n-1))) + c(\log(n-(n-2))) + \dots + c(\log(n-2)) + c(\log(n-1)) + c(\log(n))$$

$$= T_m(0) + c(\log 1) + c(\log 2) + \dots + c(\log(n-2)) + c(\log(n-1)) + c(\log(n))$$

$\log a + \log b = \log ab$

$= T_m(0) + c(\log(1 * 2 * \ldots * (n-2) * (n-1) * n))$

$= T_m(0) + c(\log(n!))$

$= c1 + c(\log(n!))$

$n!$ is upper bounded by n^n $(1! = 1^1, 2! < 2^2\ \ 3! < 3^3\ \ )$

$= c1 + c(\log(n^n)])$

$\log a^b = b\log a$

$= c1 + c(n\log(n))$

$T_m(n) = c1 + c(n\log(n))$

$= \boldsymbol{O(n\log(n))}$