2/27/20

Thursday, February 27, 2020 2:28 PM

Telescoping Method

$$\begin{cases} T(1) = 0 \\ T(n) = T(n-1)+1, & n>1 \end{cases}$$

general case

$T(n)-T(n-1) = 1$

$T(n-1)-T(n-2) = 1$ ← subbing $(n-1)$ for $T(n)$

$T(n-2)-T(n-3)=1$

$\vdots$

Last equation = $T(2)-T(1)=1$

Aligning them side by side

$$(T(n)-\cancel{T(n-1)})+(\cancel{T(n-1)}-\cancel{T(n-2)})+(\cancel{T(n-2)}-\cancel{T(n-3)})+\dots+(\cancel{T(2)}-T(3)) = \underbrace{1+1+1\dots+1}_{n-1}$$

$= T(n)-T(1) = n-1$

$T(n) = T(1)+(n-1)$

$= 0+n-1$

$= n-1$

our equation has soln $T(n)=n-1$

verification (by induction)

basis: $n=1$

$T(1)=1-1=0$

Inductive hypothesis

Assume $T(K)=K-1$

Inductive step:

Must show that $T(K+1)=(K+1)-1$
$=K$

$T(K+1)=T(K)+1$

sub $T(K)$

$T(K+1)=(K-1)+1$

$=K$ ☒

Another ex)

$$\begin{cases} T(1)=0 \\ T(n)=T(n-1)+n, & n>1 \end{cases}$$

system of eq:

$T(n)-T(n-1) = n$

$T(n-1)-T(n-2) = n-1$

$\vdots$

$T(2)-T(1) = 2$

$(T(n)-T(n-1))+(T(n-1)-T(n-2))+\dots$

$T(n) - T(1) = 2+3\dots+(n-1)+n$

$$T(n) = T(1) + \sum_{i=2}^{n} i$$

↑ 0

$$T(n) = \sum_{i=2}^{n} i = \sum_{i=1}^{n} i - 1$$

-1 bc i added one

$= \frac{n(n+1)}{2} - 1$

$= \frac{n(n+1)-2}{2}$

$= \frac{n^2+n-2}{2}$

$$T(n) = \frac{n^2+n-2}{2}$$

Answer: $T(n) = \frac{n^2+n-2}{2}$

verification

Basis $n=1$

$T(1) = \frac{1^2+1-2}{2} = 0$

ind hype: assume

$T(K) = \frac{K^2+K-2}{2}$

ind step must show $T(K+1) = \frac{(K+1)^2+(K+1)-2}{2}$

$T(K+1) = \frac{K^2+2K+1+K+1-2}{2}$

$= \frac{K^2+3K}{2} = \frac{K(K+3)}{2}$

$T(1)=0$

$T(2) = T(1)+2 = 0+2=2$

$T(3) = T(2)+3 = 2+3=5$

$T(4) = T(3)+4 = 5+4=9$

$T(n) = \frac{n^2+n-2}{2}$

$\hookrightarrow T(K+1) = \frac{K^2+2K+1+K+1-2}{2}$

$= \frac{K^2+3K}{2} = \frac{K(K+3)}{2}$

$T(K+1) = T(K) + (K+1)$

$= \frac{K^2+K-2}{2} + (K+1)$

$= \frac{K^2+K-2+2(K+1)}{2}$

$= \frac{K^2+3K}{2}$

$= \frac{K(K+3)}{2}$ ☒

$T(n) = \frac{n^2+n-2}{2}$

$T(1) = 0$

$T(2) = \frac{2^2+2-2}{}$

Suppose that you have array of sorted #s

A(n) - sorted

x - key (determine if it's in array)

for i ← 1 to ~~n~~ do ← use only when we know the # of repetitions

if x == A(i) then

return (true)

n x O(1) = O(n) worst case

use while loop

while x < A(i)

if x = A(i) then

return (true)

OR ⟹

x-found ← false

while !x-found do

if x = A(i) then

x-found ← true

best case

n x O(1) = O(n)

Code analysis: non recursive

best - O(n)

worst = O(n) } O(n)

Linear search

Better ⇓

Binary Search

$A(n/2) \stackrel{?}{=} x$

if $A(\frac{n}{2}) > n$

recursive binary search procedure

procedure BINARY_SEARCH(A(n), x)

mid_value ← A(n/2)

if x < mid_value then

BINARY_SEARCH (A(1...n/2-1), x)

else

Binary_Search (A(n/2+1...n, x)

end-if

end binary search

improve as follows:

procedure Binary_Search (A(n), x)

mid-value ← A(n/2) O(1)

if x == mid-value then O(1)

return (true)

} O(1)

else if x < mid value then
Binary search $(A(1 \ldots n/2 - 1), x)$

else
Binary Search $(A(n/2+1 \ldots n), x)$
endif
end bin search

recursive eq for ↗
will give running time:

$$\begin{cases} T(1) = 1 \\ T(n) = T(n/2) + 1 & n > 1 \end{cases}$$

solve: if using teloscoping → do domain transformation (change eq)

$n = 2^k$

$T(n) = T(2^k) = S(k)$

$T(n/2) = T(2^k/2) = T(2^{k-1}) = S(k-1)$

$T(1) = T(2^0) = S(0) = 1$

Transformed form

$$\begin{cases} S(0) = 1 \\ S(k) = S(k-1) + 1, & k \geq 0 \end{cases}$$

$S(k) - S(k-1) = 1$

$S(k-1) - S(k-2) = 1$

$S(1) - S(0) = 1$

$S(k) \qquad - S(0) = k$

$S(k) = S(0) + k$

$= k + 1$

$n = 2^k \rightarrow \log_2 2^k = k$

$k = \log n$

$S(k) = T(2^k) = T(n) = \log n + 1$

$T(n) = \log n + 1$

verification:

Basis $n = 1$

$T(1) = \log 1 + 1 = 0 + 1 = 1$ ✓

hypothesis: assume $T(k) = \log k + 1$

step: show $T(2k) = \log(2k) + 1$

$T(2k) = T(k) + 1$

$= (\log k + 1) + 1$

$= (\log k + \log 2) + 1$

$= \log(2k) + 1$ ⊠

$\log_a a = 1$

$\log(xy) = \log x + \log y$