CS:3330, Algorithms

HW06 Recurrences

Logan Zweifel September 26, 2021

1 A_k

Consider the following recurrence:

$$a_k = \begin{cases} 0 & \text{if } k\text{=}0 \\ a_{k-1} + 3k + 1 & \text{if } k\text{>}\ 0 \end{cases}$$

a) Write out the first six terms of the recurrence.

$$\begin{aligned} a_0 &= 0 = 0 = 0 \\ a_1 &= a_0 + 3(1) + 1 = 0 + 3 + 1 = 4 \\ a_2 &= a_1 + 3(2) + 1 = 4 + 6 + 1 = 11 \\ a_3 &= a_2 + 3(3) + 1 = 11 + 9 + 1 = 21 \\ a_4 &= a_3 + 3(4) + 1 = 21 + 12 + 1 = 34 \\ a_5 &= a_4 + 3(5) + 1 = 34 + 15 + 1 = 50 \\ a_6 &= a_5 + 3(6) + 1 = 50 + 18 + 1 = 69 \end{aligned}$$

b) Make a guess for the explicit formula for a_k.

The first pattern that I see in the first terms denoted in part a of the problem is that there is a summation from 1 to k taking place, $[1+2+3+\cdots+k]$. I saw this because every a_{k-1} term can be expressed as the sum of all the terms lower than it, thus there is a summation from 1 to k. Per example A.1 in Appendix A of the textbook,

$$1 + 2 + \cdots + k = \frac{k(k+1)}{2}$$

In addition, this summation is multiplied by 3 on every term. Using this information, the $a_{k-1} + 3k$ portion of the recurrence is taken care of. The +1 in every term of the recurrence can just be represented as k as adding 1 to itself k times will just equal k. Therefore the first half of the explicit formula is,

$$3[1 + 2 + 3 + \cdots + k] + k$$

The following is the work for simplifying the first half of the explicit equation using substitution for the summation from 1 to k stated earlier.

$$\begin{aligned}
3[1+2+3+\cdots+k]+k &= 3\frac{k(k+1)}{2}+k \\
&= \frac{3k(k+1)}{2}+k \\
&= \frac{3k^2+3k}{2}+k \\
&= \frac{3k^2+3k}{2}+\frac{2k}{2} \\
&= \frac{3k^2+5k}{2}
\end{aligned}$$

This gives the final explicit formula for a_k as,

$$3[1+2+3+\cdots+k]+k=\frac{3k^2+5k}{2}$$

c) Prove your guess is correct using induction.

I show that for all positive integers k, that

$$3[1+2+3+\cdots+k]+k=\frac{3k^2+5k}{2}$$

Induction base: For n=1,

$$\begin{aligned}
3[1]+1 &= \frac{3(1)^2+5(1)}{2} \\
4 &= \frac{3+5}{2} \\
4 &= 4
\end{aligned}$$

Induction Hypothesis: Assume, for an arbitrary positive integer n, that

$$3[1+2+3+\cdots+k]+k=\frac{3k^2+5k}{2}$$

Induction step: Must show that

$$3[1+2+3+\cdots+(k+1)]+(k+1)=\frac{3(k+1)^2+5(k+1)}{2}$$

First, I am going to simplify the right side of this equation.

$$\begin{aligned}
\frac{3(k+1)^2+5(k+1)}{2} &= \frac{3k^2+6k+3+5k+5}{2} \\
&= \frac{3k^2+5k+6k+8}{2} \\
&= \frac{3k^2+5k}{2}+\frac{6k+8}{2} \\
&= \frac{3k^2+5k}{2}+3k+4
\end{aligned}$$

Now, I will show that the left side of the of the induction hypothesis for $k+1$ is equal to the simplified right side.

$$\begin{aligned}
3[1+2+3+\cdots+(k+1)]+(k+1) &= 3[1+2+\cdots+k]+k+3(k+1)+1 \\
&= \frac{3k^2+5k}{2}+3(k+1)+1 \\
&= \frac{3k^2+5k}{2}+3k+3+1 \\
&= \frac{3k^2+5k}{2}+3k+4
\end{aligned}$$

Inbetween the first and second equations above, the induction hypothesis was substituted into the equation. The induction hypothesis is true for $k+1$ and therefore it is has been proven via induction.

2 Master Theorem

Use the Master Theorem to solve the recurrence

$$W(n) = 4W(\frac{n}{2}) + n$$

The following is the definition of the Master Theorem. If $f(n) \in \Theta(n^d)$, then

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

For this problem, following the defintion of the Master Theorem, $a = 4$, $b = 2$, $f(n) = n$ and $d = 1$ as that is the degree of $f(n)$. $b^d \equiv 2^1 = 2$. In this situation, the third case of the Master theorem applies as $a > b^d \equiv 4 > 2$. Therefore,

$$W(n) \in \Theta(n^{\log_b a}) \equiv W(n) \in \Theta(n^2)$$

because $\log_b a \equiv log_4 2 = 2$.

3 Find Largest Index

a) Write the pseudocode for a divide-and-conquer algorithm that finds an index for the largest element in a list of n numbers.

function FINDLARGEST($S, low, high$) ▷ initial parameters are (S, 0, n)
  if $right - left == 1$ **then**
    return left
  end if
  $middle = \lfloor (low + high)/2 \rfloor$
  $maxLeft =$ FINDLARGEST(S, low, middle)
  $maxRight =$ FINDLARGEST(S, middle, high)
  if $S[maxLeft] \geq S[maxRight]$ **then**
    return $maxLeft$
  else
    return $maxRight$
  end if
end function

b) What will be your algorithms output for lists with several elements of largest value

In the event there are multiple largest values in the list, the index returned will be of the leftmost/earliest in the list largest value.

c) Set up a recurrence relation for the number of key comparisons made by your algorithm

The recurrence relation for the algorithm written in part a of this problem is:

$$\begin{aligned} T(n) &= 2 * T(\frac{n}{2}) + 1 \\ T(1) &= 0 \end{aligned}$$

The values for this recurrence relation were identified because for every recursion, there are 2 smaller instances created with the size $\frac{n}{2}$, and the +1 was identified because there is only 1 simple comparison needed between splitting a larger instance to a smaller one and combining the solutions of smaller instances with each other. The initial condition was identified as $T(1) = 0$ as when n=1 the base/termination case is triggered before any recursion or comparisons to key elements are executed.

d) Solve the recurrence relation set up in the previous part.

Similar to problem 2, and thus following the definition stated in it, I will be using the master theorem to solve the recurrence relation from part c. Per the definition of the Master Theorem, $a = 2$, $b = 2$, $f(n) = 1$ and $d = 0$ as that is the degree of $f(n)$. $b^d \equiv 2^0 = 1$. In this situation, the third case of the Master theorem applies as $a > b^d \equiv 2 > 1$. Therefore,

$$T(n) \in \Theta(n^{\log_b a}) \equiv T(n) \in \Theta(n)$$

because $\log_b a \equiv log_2 2 = 1$.