Module 3 Recurrence Relations Assignment

Tyler Kim
tkj9ep

October 11, 2022

Response 1

1. The algorithm would take in two integer inputs, *int start* and *int end*, and would return an integer. The algorithm would check if $end - start == 2$, if so the algorithm would call f([0], [1]). If f([0], [1]) == -1, then the algorithm returns the parameter *start* and if f([0],[1]) == 1, then the algorithm returns the parameter *end*. Else, the algorithm would calculate mid by taking the size, n, and dividing it by 2. If n is even, the algorithm would check if the left or right side of the array is bigger by calling f([0..mid], [mid..end]). If f([0..mid], [mid..end]) == -1, then the algorithm would recursively call on itself with $start = 0$ and $end =$ mid. If the function resulted in 1, then the algorithm will call on [mid..end] of array instead. If n is odd, then the function would check if the left half and right half (excluding the exact middle value) of the array is bigger or not. If calling f([0..mid],[mid+1..end]) is 0, then *mid* must be the index. If not, then the algorithm will proceed the recursion calls like previously described. 2.

Response 2

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

Unroll the Recurrence
Let d denote level of unrolling

$d = 1$: $T(n) = T(n-1) + n$
$d = 2$: $T(n) = [T(n-2) + (n-1)] + n = T(n-2) + 2n - 1$
$d = 3$: $T(n) = [T(n-3) + (n-2)] + 2n - 1 = T(n-3) + 3n - 3$
$d = 4$: $T(n) = [T(n-4) + (n-4)] + 3n - 3 = T(n-4) + 4n - 7$

General Pattern: $T(n) = T(n-d) + dn - (2^{d-1} - 1)$

The base case when $T(1)$ is reached when $n - d = 1$.
Solve for d:
$n - d = 1$
$-d = 1 - n$
$d = n - 1$

Plug d back in:
$T(n) = T(n - (n-1)) + (n-1)n - (2^{n-1-1} - 1)$
$T(n) = T(1) + n^2 - n - 2^{n-1} + 1$
$T(n) = n^2 - n - 2^{n-2} + 1 = \Theta(2^n)$
$\therefore \Theta(2^n)$

Response 3

Response 4

Given: $T(n) = 2T(\frac{n}{4}) + 1$

Apply Master Theorem:
A = 2, B = 4, $f(n) = 1$
$k = \frac{\log 2}{\log 4} = \frac{1}{2}$
Compare $f(n) = 1$ to $n^{\frac{1}{2}}$
Since $f(n) = O(n^{\frac{1}{2}-\epsilon})$ where $\epsilon = \frac{1}{2}$, Case 1 applies:
$T(n) \in \Theta(n^{\frac{1}{2}})$
The solution must be $T(n) = \Theta(n)$ since $k = \frac{1}{2}$ and rounds up to 1

Response 5

Given: $T(n) = 2T(\frac{n}{4}) + \sqrt{n}$

Apply Master Theorem:
A = 2, B = 4, $f(n) = \sqrt{n}$
$k = \frac{\log 2}{\log 4} = \frac{1}{2}$
Compare $f(n) = \sqrt{n}$ to $n^{\frac{1}{2}}$
Since $f(n) = \sqrt{n}$ is equal to $n^k = n^{\frac{1}{2}}$, then we apply Case 2:
$T(n) = \Theta f(n) log(n) = \Theta(n^{\frac{1}{2}} log(n^{\frac{1}{2}}))$
$\therefore \Theta(nlog(n))$

Response 6

Given: $T(n) = 2T(\frac{n}{2}) + n$

Apply Master Theorem:
A = 2, B = 4, $f(n) = n$
$k = \frac{\log 2}{\log 4} = \frac{1}{2}$
Compare $f(n) = n$ to $n^{\frac{1}{2}}$
Since $n^{\frac{1}{2}-\epsilon}$ results in $\epsilon = \frac{1}{2}$, and $cf(n) \geq n^{\frac{1}{2}}$, apply Case 3:
$T(n) \in \Theta(f(n))$.
$\therefore \Theta(n)$

Response 7

Given: $T(n) = 2T(\frac{n}{4}) + n^2$

Apply Master Theorem:
A = 2, B = 4, $f(n) = n^2$
$k = \frac{\log 2}{\log 4} = \frac{1}{2}$
Compare $f(n) = n^2$ and $n^{\frac{1}{2}}$
Since $n^{\frac{1}{2}+\epsilon}$ results in $\epsilon = 1.5$ and $cf(n) \geq n^{\frac{1}{2}}$, apply Case 3:
$T(n) \in \Theta f(n)$.
$\therefore \Theta(f(n))$