Analysis of Divide-and-Conquer Algorithms

The divide-and-conquer paradigm (Ch. 2)
• divide the problem into a number of subproblems
• conquer the subproblems by solving them
• combine the subproblem solutions to get the solution to the problem

add all these steps at the first level to get recurrence relation for $T(n)$

Example: Merge-Sort: an optimal sorting algorithm
• divide the $n$-element input sequence to be sorted into two $n/2$-element subsequences.
• conquer the subproblems recursively using merge sort.
• combine the resulting two sorted $n/2$-element sequences by merging.

Divide (lgn + 1 levels)

2 8 1 5 4 3 7 6

Why are there $\lg n + 1$ levels? Because $\lg n + 1$ is the number of steps it takes to divide $n$ by 2 until the size of the result is <= 1

How long does it take to find the midpoint of an array? $D(n) = \Theta(1)$

Analyzing Merge-Sort

Merge-Sort(A, p, q)
1. if $p < q$ then
2. $q = \lceil (p+q)/2 \rceil$
3. Merge-Sort(A, p, q)
4. Merge-Sort(A, q+1, r)
5. Merge(A, p, q, r)

Initial call:
Merge-Sort(A, 1, length(A))

The Merge subroutine takes linear time to merge $n$ elements that are divided into two sorted arrays of $n/2$ elements each.

$T(n) = \Theta(n)$

Recurrence for worst-case running time for Merge-Sort

$T(n) = \begin{cases} \Theta(n) & \text{if } n = 1 \\ 2T(n/2) + \Theta(n) & \text{otherwise} \end{cases}$

Recursion Tree for Merge-Sort

The value of $c$ is the cost of dividing an array of $n$ elements into two subarrays.

Recursion for Merge-Sort

$T(n) = \begin{cases} \cdot cn & \text{if } n = 1 \\ c & \text{otherwise} \end{cases}$

Analyzing Merge-Sort

$h = \lg n + 1$ levels

2 8 1 5 4 3 7 6

Recurrence for worst-case running time for Merge-Sort

$aT(n/b) + D(n) + C(n)$

• $a = 2$ (two subproblems)
• $n/b = n/2$ (each subproblem has size approx $n/2$)
• $D(n) = \Theta(1)$ (just compute midpoint of array)
• $C(n) = \Theta(n)$ (merging can be done by scanning sorted subarrays)

Analyzing Merge-Sort

2 8 1 5 4 3 7 6

Why are there $\lg n + 1$ levels? Because $\lg n + 1$ is the number of steps it takes to divide $n$ by 2 until the size of the result is <= 1

How long does it take to find the midpoint of an array? $D(n) = \Theta(1)$

Analyzing Merge-Sort

2 8 1 5 4 3 7 6