Jordan Allard

Part 1: Set up recurrences, use the Master Theorem

Consider the following pseudo code sketches. For each, state its recurrence, using $T(n)$ to denote the maximum number of steps the function makes on an input of size n. In particular, (1) next to the pseudo code above, highlight its non-recursive parts and estimate their running time in big-Oh notation, and (2) next to the pseudo code above, highlight the recursive parts and estimate their running times using the function $T()$. Then, evaluate it using the Master Theorem.

(a) Function F (A – an array of size n):

   if $n = 1$: return $A[1]$

   use $O(n)$ steps and compute arrays B, C, and D of size $\lfloor n/3 \rfloor$ each

   $x = F(B)$

   $y = F(C)$

   $z = F(D)$

   return $x + y + z$

(i) Set up a recurrence for this pseudo code sketch. Do not worry about the rounding. Do not forget the base case.

$T(n) = 3T(n/3) + O(n) = 3(3T(n/9) + c\cdot\frac{n}{3}) + cn = 9T(n/9) + 2cn$

$T(n) \leq 3^k T(n/3^k) + kcn$

$T(n) = \begin{cases} O(1) & \text{if } n=1 \\ 3T(n/3) + O(n) & \text{if } n>1 \end{cases}$

(ii) Analyze the recurrence using the Master Theorem.

$a = 3,\ b = 3,\ f(n) = O(n)$

Case 2 applies $\rightarrow$ $O(n) = \Theta(n^1)$

$T(n) = \boxed{\Theta(n \log n)}$

A_1 B_2 A_2 B_1

3 8 5 2

24 10

34

(b) Function G (A – an array of size n):

if $n = 1$: return $A[1]$
use $O(n)$ steps and compute arrays B and C of size $\lfloor n/3 \rfloor$ each
$x = F(B)$
$y = F(C)$
return $x - y$

(i) Set up a recurrence for this pseudo code sketch. Do not worry about the rounding. Do not forget the base case.

$$T(n) = \begin{cases} O(1) & \text{if } n=1 \\ 2T(n/3) + O(n) & \text{if } n>1 \end{cases}$$

(ii) Analyze the recurrence using the Master Theorem.

$a=2,\ b=3,\ f(n)=O(n)$

$n^{(\log 2/\log 3)+\epsilon}$ Case 3 applies $\rightarrow$ $T(n) = \boxed{\Theta(n)}$

$2f(\frac{n}{3}) \leq c f(n)$ $c = \frac{2}{3}$

(c) Function H (A – an array of size n):

if $n = 1$: return $A[1]$
use $O(n)$ steps and compute arrays B, C, D, and E of size $\lfloor n/3 \rfloor$ each
$x = H(B)$
$y = H(C)$
$z = H(D)$
$w = H(E)$
return $x + y + z + w$

(i) Set up a recurrence for this pseudo code sketch. Do not worry about the rounding. Do not forget the base case.

$$T(n) = \begin{cases} O(1) & \text{if } n=1 \\ 4T(n/3) + O(n) & \text{if } n>1 \end{cases}$$

(ii) Analyze the recurrence using the Master Theorem.

$a=4,\ b=3,\ f(n)=O(n)$

$\boxed{\Theta\left(n^{(\log 4/\log 3)}\right)}$

$n^{\log 4/\log 3} = n^{1+\epsilon}$

Jordan Allard

ALGORITHMS, FALL '23, ACTIVITY 6: DIVIDE-AND-CONQUER

Demonstrate how the Karatsuba-Ofman algorithm works for the following input: $A = [1, 2, 3, 4]$ and $B = [5, 0, 7, 9]$.

```
LongMultiply(array A, array B - both of length n):
    if n=1: return A[1]*B[1], as a string of digits
    let m = n/2 (rounded down)
    let A1 = A[1...m] and A2 = A[m+1...n]
    let B1 = B[1...m] and B2 = B[m+1...n]
    let C = LongMultiply(A1,B1)
    let D = LongMultiply(A2,B2)
    let A12 = LongAdd(A1,A2)
    let B12 = LongAdd(B1,B2)
    let G = LongMultiply(A12,B12)
    let EF = LongSubtract(G,LongAdd(C,D))
    let Cpadded be C with 10^{2(n-m)} zeros added at the end
    let EFpadded be EF with 10^{(n-m)} zeros added at the end
    return LongAdd(Cpadded,EFpadded,D)
```

In the following, it is ok to write the arrays as numbers and not as string arrays (for clarity):

(a) What is AB (without running the algorithm)? 6 267 486

(b) Trace the algorithm (do not trace through the recursion – merely state the result of the recursive call). In particular, state the values of the following variables:

$A1$	12	$A2$	34
$B1$	50	$B2$	79
C	600	D	2686
$A12$	46	$B12$	129
G	5934	EF	2648
$Cpadded$	6000000	$EFpadded$	264800
return	6 267 486		

(c) Is the return value correct? Yes