Until now: Analyse recursive algorithms

CSE525 Lec3: Recursion

Debajyoti Bera (M21)

Multiplication: Divide and Conquer

complexity ≈ # single digit multiplication (additions are free)

2341 7892 — 8-digit

× 65 68 17 — 6-digit

left shifting

78×17 + carry

92×17

123
6 5 00

78×68 + carry

92×68

← multiply by 10

Complexity to multiply two n-bit #s $T(n) = 4T(n/2) + O(1) = O(n^2)$

2341 7892

65 6817

Mult divides both number in two halves.

Example divides smaller number asymmetrically.

def Algo P (input x):

trivial / brute force / naive approach

0. How to solve P for very small inputs

1. Suppose magically AlgoP(y) is always correct when $|y| < |x|$

→ write code to solve P on x?

$T(n,m) = 4T(n/2, m/2) + O(1)$

23417892×656817

$= (2341 \times 10^4 + 7892)(65 \times 10^4 + 6817) = O(n^2 m^2)$

$= \underbrace{2341 \times 65}_{P_1} \times 10^8 + \underbrace{(2341 \times 6817}_{P_2} + \underbrace{7892 \times 65}_{P_3}) \times 10^4$

$+ \underbrace{7892 \times 6817}_{P_4}$

left shifting while adding

?

def Mult (A, B): ($A \to n$, $B \to m$) ← add a base case

if n=1, m=1 return AB

write $A = A_1 \times 10^{n/2} + A_2$

write $B = B_1 \times 10^{m/2} + B_2$

→ compute $P_1 = \text{Mult}(A_1, B_1)$, $P_3 = \text{Mult}(A_2, B_1)$

$P_2 = \text{Mult}(A_1, B_2)$, $P_4 = \text{Mult}(A_2, B_2)$

return $P_1 \times 10^{\frac{n}{2}+\frac{m}{2}} + P_2 \times 10^{n/2} + P_3 \times 10^{m/2} + P_4$

Multiplication: Divide and Conquer

Two 2m-bit numbers

$\boxed{\overbrace{a}^{m} \;\big|\; \overbrace{b}^{m}}\; X \qquad Y\; \boxed{c \;\big|\; d}$

$$XY = (10^m a + b)(10^m c + d) = 10^{2m} ac + 10^m (bc + ad) + bd$$

- Recursive algorithm ?
- Time complexity ?

123 456

123 : a   b : 456

two 2m digit numbers

Multiplication: Karatsuba's algorithm

$x_m \cdots x_3\ x_2\ x_1$

$y_m \cdots y_3\ y_2\ y_1$

linear pass

2 rec calls

rec call

X : 2m digit Y

$$XY = (10^m a + b)(10^m c + d) = 10^{2m} ac + 10^m (bc + ad) + bd$$

ac : two m digit numbers

bd : " "

rec call

$T(K)$: complexity to multiply two K-digit numbers

→ Use the identity: bc + ad = ac + bd - (a-b)(c-d)

(a-b)x(c-d) : two numbers ≤ m digit

one more rec call

(assuming subtraction is free)

1. a·c

2. b·d

3. (a-b)·(c-d)

Use the identity: bc + ad = $\underbrace{(a+b)}_{m+1 \text{ digit}}$(c+d) - ac -bd

$T(2m) = 3T(m) +$ cost for addition/subtraction

→ $O(1)$ if add/sub free

$O(m)$ o/w

$T(2m) = 2T(m) + T(m+1) + O(m)$

$= O(\cancel{m^{1.5}})$

$= O(m^{\lg_2 3})$

when cost to addition/subtraction is $O(m)$

- Recursive algorithm ?
- Time complexity ?
- Which identity is better ?
- Proof of correctness?

$\boxed{2}\boxed{3} = 2 \times 10^1 + 3$

Select(A,k) $\rightarrow$ kth smallest element of A

unsorted, n elements

$k=1 \rightarrow$ min $\qquad k=|A|/2 \rightarrow$ median

$k=|A| \rightarrow$ max

Naive solution: Sorting based $\rightarrow O(n\lg n)$

Given an array A[1...n] and an integer k <= n, return the k-*th* small element in A.

def Myselect (A, k):
    basecase (trivial)

Assume Myselect (B, k) is correct if $|B|<n$. } not helpful

Myselect([5, 1, 19, 2, 7, 16, 11, 10, 12, 6], 5):
    Myselect([5, 1, 19, 2, 7, 16, 11, 10, 12], 5) $\rightarrow x$
    If $x > 6$: return ~~6~~ + Add basecase (tricky)
    else $x < 6$: return x

Assume Myselect(B, k) is correct if $|B| < n$.

Myselect([5, ... 12], 4) $\rightarrow x$

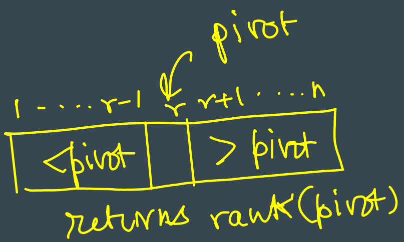
if $x < 6$:

exercise

if $x > 6$:

Select(A,k)

QuickSelect(A,k):



1. If A is small, brute force
2. r = partition(A) // returns rank(pivot)
3. If k < r: return QuickSelect(A,k) A[1...r-1]
4. If k = r: return A[k]
5. If k > r: return QuickSelect(A, __) A[r+1...n], k-r

Time complexity recurrence:

T(n) = ______________________________

T(small) = brute force

T(n) = O(n) // time to partition

$\max_{r=1\ldots n}\{T(r-1),\ T(n-r)\}$

+ ______________ // time during recursion

What is the best way to choose pivot ?

$\max\{\max\{1,2,3\},\ \max\{11,21\}\}$

$= \max\{1,2,3,11,21\}$