CSE 421
Introduction to Algorithms

Richard Anderson

Lecture 11, Winter 2024

Divide and Conquer

1

Announcements

- Divide and Conquer and Recurrences
  - ~~Recurrence Techniques~~
  - ~~Fast Matrix Multiplication~~
  - ~~Counting Inversions (5.3)~~
  - ~~Closest Pair (5.4)~~
  - Integer Multiplication (5.5)
  - Quicksort and Median Finding
- Dynamic Programming
- Midterm, Friday, February 9

$$k^{\log_k n} = n$$

$$\log_k n = \frac{\log_2 n}{\log_2 k}$$

$$k^{\log_2 n} = n^{\log_2 k}$$

$$\sum_{i=0}^{n} x^i = \frac{1 - x^{n+1}}{1 - x}$$

2

Integer Arithmetic

```
    9715480283945084383094856701043643845790217965702956767
  + 1242431098234099057329075097179898430928779579277597977
  ---------------------------------------------------------
```

Runtime for standard algorithm to add two n digit numbers:

```
    2095067093034680994318596846868779409766717133476767930
  X 5920175091777634709677679342929097012308956679993010921
  ---------------------------------------------------------
```

Runtime for standard algorithm to multiply two n digit numbers:

3

Recursive Multiplication Algorithm (First attempt)

$x = x_1 2^{n/2} + x_0$

$y = y_1 2^{n/2} + y_0$

$xy = (x_1 2^{n/2} + x_0)(y_1 2^{n/2} + y_0)$

$= x_1 y_1 2^n + (x_1 y_0 + x_0 y_1) 2^{n/2} + x_0 y_0$

Recurrence:

Run time:

4

Simple algebra

$x = x_1 2^{n/2} + x_0$

$y = y_1 2^{n/2} + y_0$

$xy = x_1 y_1 2^n + (x_1 y_0 + x_0 y_1) 2^{n/2} + x_0 y_0$

$p = (x_1 + x_0)(y_1 + y_0) = x_1 y_1 + x_1 y_0 + x_0 y_1 + x_0 y_0$

5

Karatsuba's Algorithm

Multiply n-digit integers x and y

Let $x = x_1 2^{n/2} + x_0$ and $y = y_1 2^{n/2} + y_0$

Recursively compute

$a = x_1 y_1$

$b = x_0 y_0$

$p = (x_1 + x_0)(y_1 + y_0)$

Return $a2^n + (p - a - b)2^{n/2} + b$

Recurrence: $T(n) = 3T(n/2) + cn$

$\log_2 3 = 1.58496250073\ldots$

6

Quicksort [Tony Hoare, 1959]

QuickSort(S):

1. Pick an element *v* in **S**. This is the ***pivot*** value.
2. Partition **S**-{*v*} into two disjoint subsets, $\mathbf{S}_1$ and $\mathbf{S}_2$ such that:
   - elements in $\mathbf{S}_1$ are all < *v*
   - elements in $\mathbf{S}_2$ are all > *v*
3. Return concatenation of QuickSort($\mathbf{S}_1$), *v*, QuickSort($\mathbf{S}_2$)

Recursion ends if Quicksort( ) receives an array of length 0 or 1.

7

Quicksort – worst case

- Pivot selection: choose first element
- Sort [1,2,3,4,5,6,. . . N]

8

Quicksort - pragmatics

- Pivot selection rules
  – Median of first, middle, and last
  – Choose random element
- In place implementation
- Algorithm engineering for partitioning
- Recursion cutoff for small problems

9

Average case analysis for Quicksort

- All inputs equally likely
  – Or random elements used for pivot
  – Or input is randomly shuffled

- QS(n) = average number of comparisons for Quicksort on input of size n.

10

Building a recurrence

Pivot chosen at random. The chance of having i elements less than the pivot is 1/n.

$$T(n) = (n-1) + \frac{1}{n}\sum_{i=0}^{n-1}(T(i) + T(n-i-1)).$$

$$T(n) = (n-1) + \frac{2}{n}\sum_{i=1}^{n-1} T(i)$$

Solution: T(N) ≈ 2 n ln n

11

Computing the Median

- Given n numbers, find the number of rank n/2
- One approach is sorting
  – Sort the elements, and choose the middle one
  – Can you do better?

12

Problem generalization

- *Selection*, given n numbers and an integer k, find the k-th largest

13

Select(A, k)

```
Select(A, k){
        Choose element x from A
        S1 = {y in A | y < x}
        S2 = {y in A | y > x}
        S3 = {y in A | y = x}
        if (|S2| >= k)
                return Select(S2, k)
        else if (|S2| + |S3| >= k)
                return x
        else
                return Select(S1, k - |S2| - |S3|)
}
```

S_1	S_3	S_2

14

Randomized Selection

- Choose the element at random
- Analysis can show that the algorithm has expected run time O(n)

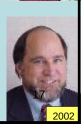
15

Deterministic Selection

- What is the run time of select if we can guarantee that choose finds an x such that $|S_1| < 3n/4$ and $|S_2| < 3n/4$ in O(n) time

16

BFPRT Algorithm

- A very clever choose algorithm . . .

Split into n/5 sets of size 5
M be the set of medians of these sets
Let x be the median of M

BFPRT runtime

$|S_1| < 3n/4$, $|S_2| < 3n/4$

Split into n/5 sets of size 5
M be the set of medians of these sets
x be the median of M
Construct S_1 and S_2
Recursive call in S_1 or S_2

18

BFPRT Recurrence

$T(n) \leq T(3n/4) + T(n/5) + c\,n$

Prove that $T(n) \leq 20\,c\,n$

19

A theoretical aside

- How many comparisons are needed in the worst case to find the median?
- BFPRT showed that this is at most 18 n
- Best known results in 3 n (but its complicated)
- The lower bound was shown to be at least 2 n by Bent and John
  - Improved to 2.01 n by Zwick

20