CS4102 Algorithms
Spring 2019

**Warm up**

Simplify:

$$1 + 2 + 3 + \cdots + (n-1) + n =$$

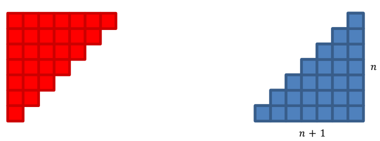
$1 + 2 + 3 + \cdots + 98 + 99 + 100$
$+ \; 100 + 99 + 98 + \cdots + 3 + 2 + 1$
$101 + 101 + 101 + \cdots + 101 + 101 + 101$

$\frac{100(101)}{2}$

1

$$1 + 2 + 3 + \cdots + (n-1) + n =$$

$$\frac{n(n+1)}{2}$$




2

## Today's Keywords

- Divide and Conquer
- Matrix Multiplication
- Strassen's Algorithm
- Sorting
- Quicksort

3

## CLRS Readings

- Chapter 4
- Chapter 7



## Homeworks

- Hw2 due 11pm next Wednesday!
  - Programming (use Python or Java!)
  - Divide and conquer
  - Closest pair of points
  - Note: you will need to write a recursive function in:
    - closest_pair.py or
    - ClosestPair.java



## Matrix Multiplication

$$n\overset{n}{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}} \times \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix}$$

$$= \begin{bmatrix} 2+16+42 & 4+20+48 & 6+24+54 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}$$

$$= \begin{bmatrix} 60 & 72 & 84 \\ 132 & 162 & 192 \\ 204 & 252 & 300 \end{bmatrix}$$

Run time? $O(n^3)$ Lower Bound? $O(n^2)$

## Matrix Multiplication D&C

Multiply $n \times n$ matrices ($A$ and $B$)

Divide:

$$A = \left[\begin{array}{cc|cc} a_1 & a_2 & a_3 & a_4 \\ a_5 & a_6 & a_7 & a_8 \\ \hline a_9 & a_{10} & a_{11} & a_{12} \\ a_{13} & a_{14} & a_{15} & a_{16} \end{array}\right] \qquad B = \left[\begin{array}{cc|cc} b_1 & b_2 & b_3 & b_4 \\ b_5 & b_6 & b_7 & b_8 \\ \hline b_9 & b_{10} & b_{11} & b_{12} \\ b_{13} & b_{14} & b_{15} & b_{16} \end{array}\right]$$

7

## Matrix Multiplication D&C

Multiply $n \times n$ matrices ($A$ and $B$)

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \qquad B = \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}$$

Combine:

$$AB = \begin{bmatrix} A_{1,1}B_{1,1} + A_{1,2}B_{2,1} & A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ A_{2,1}B_{1,1} + A_{2,2}B_{2,1} & A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{bmatrix}$$

How many matrix multiplications?
Size of sub-problems?
How many additions?
Cost of adding two sub-matrices?

Run time? $T(n) = 8T\left(\frac{n}{2}\right) + 4\left(\frac{n}{2}\right)^2$ Cost of additions

8

## Matrix Multiplication D&C

$$T(n) = 8T\left(\frac{n}{2}\right) + 4\left(\frac{n}{2}\right)^2$$

$$T(n) = 8T\left(\frac{n}{2}\right) + n^2$$

$a = 8, b = 2, f(n) = n^2$

Case 1!

$n^{\log_b a} = n^{\log_2 8} = n^3$

$T(n) = \Theta(n^3)$ We need to be more clever… How?

9

## Matrix Multiplication D&C

Multiply $n \times n$ matrices ($A$ and $B$)

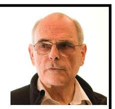
$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \qquad B = \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}$$

$$AB = \begin{bmatrix} A_{1,1}B_{1,1} + A_{1,2}B_{2,1} & A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ A_{2,1}B_{1,1} + A_{2,2}B_{2,1} & A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{bmatrix}$$

Idea: Use a Karatsuba-like technique on this

10

## Strassen's Algorithm



Multiply $n \times n$ matrices ($A$ and $B$)

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \qquad B = \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}$$

Calculate:

$Q_1 = (A_{1,1} + A_{2,2})(B_{1,1} + B_{2,2})$
$Q_2 = (A_{2,1} + A_{2,2})B_{1,1}$
$Q_3 = A_{1,1}(B_{1,2} - B_{2,2})$
$Q_4 = A_{2,2}(B_{2,1} - B_{1,1})$
$Q_5 = (A_{1,1} + A_{1,2})B_{2,2}$
$Q_6 = (A_{2,1} - A_{1,1})(B_{1,1} + B_{1,2})$
$Q_7 = (A_{1,2} - A_{2,2})(B_{2,1} + B_{2,2})$

Find $AB$:

$$\begin{bmatrix} A_{1,1}B_{1,1} + A_{1,2}B_{2,1} & A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ A_{2,1}B_{1,1} + A_{2,2}B_{2,1} & A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{bmatrix} = \begin{bmatrix} Q_1 + Q_4 - Q_5 + Q_7 & Q_3 + Q_5 \\ Q_2 + Q_4 & Q_1 - Q_2 + Q_3 + Q_6 \end{bmatrix}$$

Number Mults.: 7 Number Adds.: 18

$$T(n) = 7T\left(\frac{n}{2}\right) + 18\left(\frac{n}{2}\right)^2$$

11

## Strassen's Algorithm

$$T(n) = 7T\left(\frac{n}{2}\right) + \frac{9}{2}n^2$$

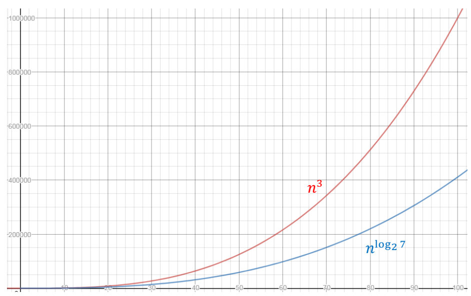
$$a = 7, b = 2, f(n) = \frac{9}{2}n^2$$

Case 1!

$$n^{\log_b a} = n^{\log_2 7} \approx n^{2.807}$$

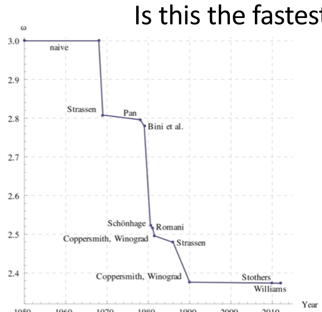
$$T(n) = \Theta\left(n^{\log_2 7}\right) \approx \Theta(n^{2.807})$$

12




13

## Is this the fastest?




Best possible is unknown

May not even exist!

14

## Divide and Conquer, so far

- Mergesort
- Naïve Multiplication
- Karatsuba
- Closest Pair of Points
- Strassen's

What they have in common
Divide: Very easy (i.e. $O(1)$)
Combine: Hard work ($\Omega(n)$)

15

## Quicksort

- Like Mergesort:
  - Divide and conquer
  - $O(n \log n)$ run time (kind of…)
- Unlike Mergesort:
  - Divide step is the hard part
  - *Typically* faster than Mergesort

16

## Quicksort

- Idea: pick a pivot element, recursively sort two sublists around that element
- Divide: select pivot element $p$, Partition($p$)
- Conquer: recursively sort left and right sublists
- Combine: Nothing!

17

## Partition (Divide step)

- Given: a list, a pivot $p$

Start: unordered list

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 5 | 7 | 3 | 12 | 10 | 1 | 2 | 4 | 9 | 6 | 11 |

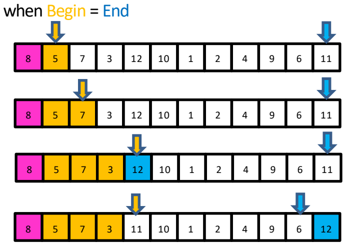
Goal: All elements $< p$ on left, all $> p$ on right

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 7 | 3 | 1 | 2 | 4 | 6 | 8 | 12 | 10 | 9 | 11 |

18

## Partition, Procedure

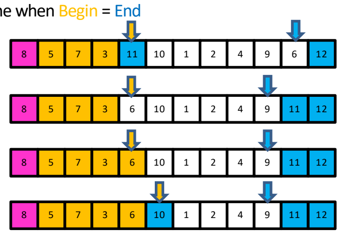
If Begin value < $p$, move Begin right
Else swap Begin value with End value, move End Left
Done when Begin = End




19

## Partition, Procedure

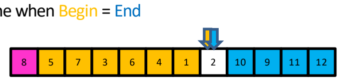
If Begin value < $p$, move Begin right
Else swap Begin value with End value, move End Left
Done when Begin = End




20

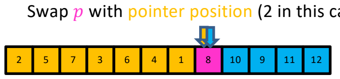
## Partition, Procedure

If Begin value < $p$, move Begin right
Else swap Begin value with End value, move End Left
Done when Begin = End




Case 1: meet at element < $p$

Swap $p$ with pointer position (2 in this case)




21

## Partition, Procedure

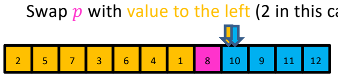
If Begin value < $p$, move Begin right

Else swap Begin value with End value, move End Left

Done when Begin = End


Case 2: meet at element $> p$

Swap $p$ with value to the left (2 in this case)




22

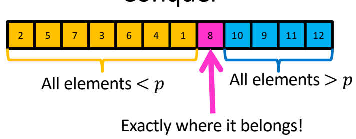
## Partition Summary

1. Put $p$ at beginning of list
2. Put a pointer (Begin) just after $p$, and a pointer (End) at the end of the list
3. While Begin < End:
   1. If Begin value $< p$, move Begin right
   2. Else swap Begin value with End value, move End Left
4. If pointers meet at element $< p$: Swap $p$ with pointer position
5. Else If pointers meet at element $> p$: Swap $p$ with value to the left

Run time? $O(n)$

23

## Conquer




Recursively sort Left and Right sublists

24

## Quicksort Run Time (Best)

- If the pivot is always the median:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 5 | 1 | 3 | 6 | 4 | 7 | 8 | 10 | 9 | 11 | 12 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 3 | 5 | 6 | 4 | 7 | 8 | 9 | 10 | 11 | 12 |

- Then we divide in half each time

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

$$T(n) = O(n \log n)$$

25

## Quicksort Run Time (Worst)

- If the pivot is always at the extreme:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 2 | 3 | 6 | 4 | 7 | 8 | 10 | 9 | 11 | 12 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 5 | 6 | 4 | 7 | 8 | 10 | 9 | 11 | 12 |

- Then we shorten by 1 each time

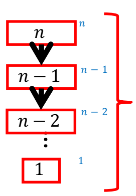
$$T(n) = T(n-1) + n$$

$$T(n) = O(n^2)$$

26

## Quicksort Run Time (Worst)

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




$$T(n) = 1 + 2 + 3 + \cdots + n$$

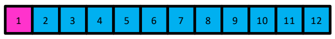
$$T(n) = \frac{n(n+1)}{2}$$

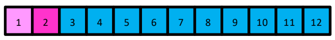
$$T(n) = O(n^2)$$

27

## Quicksort on a (nearly) Sorted List

- First element always yields unbalanced pivot




- So we shorten by 1 each time

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

$$T(n) = O(n^2)$$

28

## Takeaway Question

- How to pick the pivot?

29