A2 extra office hours: Nov 2 1-2 pm, 3-5 pm
BA 3289

# CSC236 fall 2012

## more complexity: mergesort

Danny Heap
heap@cs.toronto.edu
BA4270 (behind elevators)
http://www.cdf.toronto.edu/~heap/236/F12/
416-978-5899

Using Introduction to the Theory of Computation, Chapter 3

# Outline

divide and conquer (recombine)

using the Master Theorem

Notes

# General case

revisit...

Class of algorithms: partition problem into $b$ *roughly* equal subproblems, solve, and recombine:

$$T(n) = \begin{cases} k & \text{if } n \leq B \\ a_1 T(\lceil n/b \rceil) + a_2 T(\lfloor n/b \rfloor) + f(n) & \text{if } n > B \end{cases}$$

where $B, k > 0$, $a_1, a_2 \geq 0$, and $a_1 + a_2 > 0$. $f(n)$ is the cost of splitting and recombining.

# Master Theorem

(for divide-and-conquer recurrences)

how to efficiently count # of switches

3 217 93 ... 1

1 2 3 .. n

better than $\theta(n^2)$

If $f$ from the previous slide has $f \in \theta(n^d)$, then

$$T(n) = \begin{cases} \theta(n^d) & \text{if } a < b^d \\ \theta(n^d \log n) & \text{if } a = b^d \\ \theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

# Proof sketch

1. Unwind the recurrence, and prove a result for $n = b^k$
2. Prove that $T$ is non-decreasing
3. Extend to all $n$, similar to MergeSort

# multiply lots of bits

## what if they don't fit into a machine instruction?

$$\begin{array}{r} 1101 \\ \times 1011 \\ \hline \end{array}$$

# divide and recombine

## recursively...

| | |
|---|---|
| 11 | 01 |
| $\times$10 | 11 |

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

# compare costs

$n$ $n$-bit additions versus:

1. divide each factor (roughly) in half
2. multiply the halves (recursively, if they're too big)
3. combine the products with shifts and adds

# Gauss's trick

$$n^2 \longrightarrow n^{\log_2 3}$$

$$xy = 2^n \underline{x_1 y_1} + \underline{x_0 y_0} + 2^{n/2}\left(\underbrace{(x_1 + x_0)(y_1 + y_0)} - x_1 y_1 - x_0 y_0\right)$$

# Gauss's payoff

## lose one multiplication

1. divide each factor (roughly) in half
2. sum the halves
3. multiply the sum and the halves Gauss-wise
4. combine the products with shifts and adds

# closest point pairs

see Wikipedia

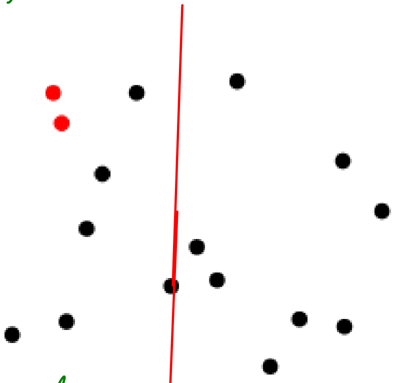
$P = [(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots]$

Brute force $\binom{n}{2} \sim \Theta(n^2)$

Sort $P$ by $x$, $y$

$\rightarrow P_x \quad P_y$

$n \log n$



recursively find min on $L$

recursively find min $R$

# divide-and-conquer v0.1

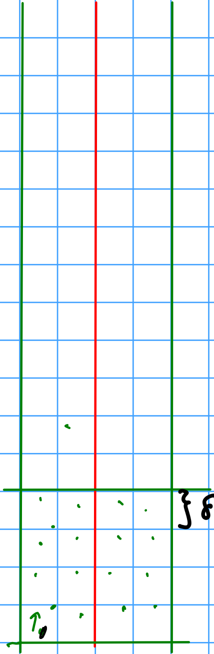
after sorting

$$T(n) = \begin{cases} k & n \leq 3 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + f(n) & n > 3 \end{cases}$$

$\min(L, R) = \delta > 0$

$D_y \rightarrow \Theta(n)$

$\sqrt{\frac{\delta^2}{4} + \frac{\delta^2}{4}} = \frac{\sqrt{2}}{2}\delta$



$\} \delta/2$

# an $\underbrace{n \lg n}$ algorithm

$\text{rec}f(P_x, P_y)$

$P_y$

$\text{rec}f(L_x, L_y)$ ← look at x

$P$ is a set of points

R

1. Construct (sort) $P_x$ and $P_y$ — $n \log n$
2. For each recursive call, construct $L_x, L_y, R_x, R_y$ — $\Theta(n)$
3. Recursively find closest pairs $(l_0, l_1)$ and $(r_0, r_1)$, with minimum distance $\delta$ — $T(n)$
4. $V$ is the vertical line splitting $L$ and $R$, $D$ is the $\delta$-neighbourhood of $V$, and $D_y$ is $D$ ordered by $y$-ordinate — $\Theta(n)$
5. Traverse $D_y$ looking for mininum pairs 15 places apart — $\Theta(n)$
6. Choose the minimum pair from $D_y$ versus $(l_0, l_1)$ and $(r_0, r_1)$.

across V

# Notes