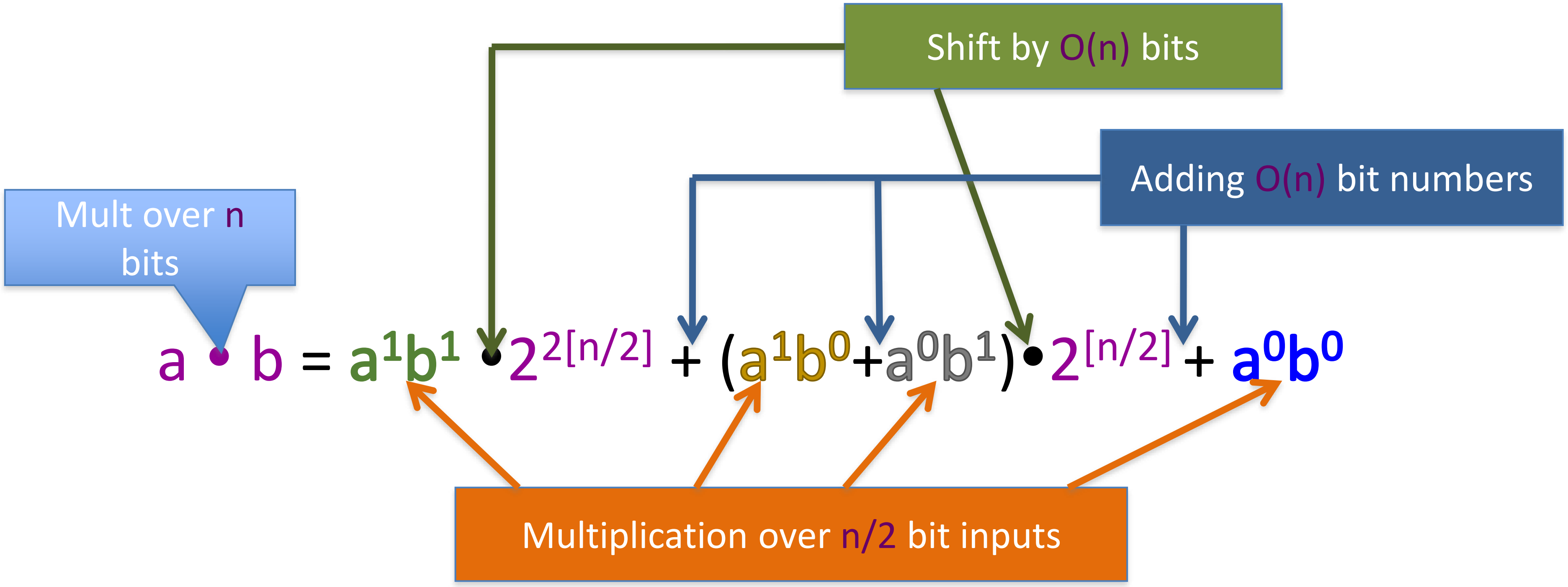
Lecture 27

CSE 331

The current algorithm scheme



$T(n) \leq 4T(n/2) + cn$

$T(1) \leq c$

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

The key identity

$$a^1b^0+a^0b^1= (a^1+a^0)(b^1+b^0) - a^1b^1 - a^0b^0$$

The final algorithm

Input: $a = (a_{n-1},..,a_0)$ and $b = (b_{n-1},...,b_0)$

Mult (a, b)

If $n = 1$ return $a_0 b_0$

$a^1 = a_{n-1},...,a_{[n/2]}$ and $a^0 = a_{[n/2]-1},..., a_0$

Compute b^1 and b^0 from b

$x = a^1 + a^0$ and $y = b^1 + b^0$

Let p = Mult (x, y), D = Mult (a^1, b^1), E = Mult (a^0, b^0)

F = p - D - E

return $D \bullet 2^{2[n/2]} + F \bullet 2^{[n/2]} + E$

$T(1) \leq c$

$T(n) \leq 3T(n/2) + cn$

$O(n^{\log_2 3}) = O(n^{1.59})$ run time

All green operations are O(n) time

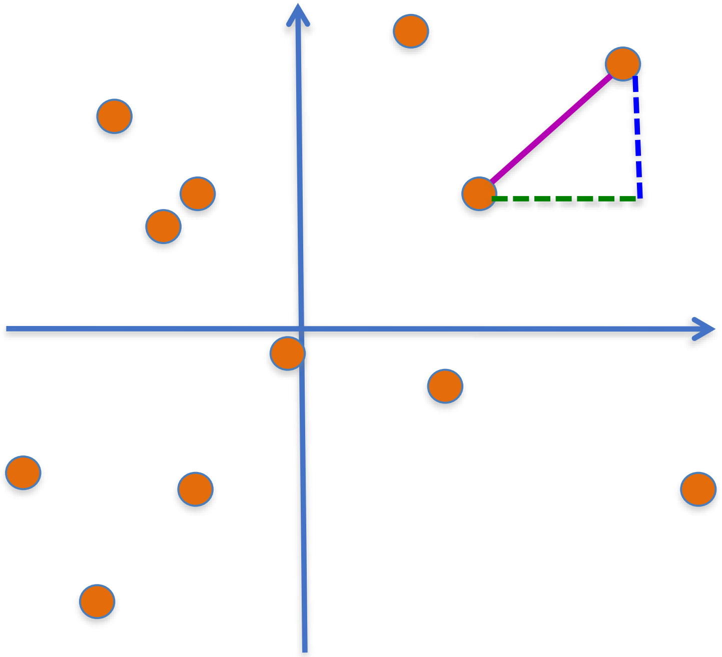
$$a \bullet b = a^1 b^1 \bullet 2^{2[n/2]} + ((a^1+a^0)(b^1+b^0) - a^1 b^1 - a^0 b^0) \bullet 2^{[n/2]} + a^0 b^0$$

Closest pairs of points

Input: n 2-D points $P = \{p_1,\ldots,p_n\}$; $p_i=(x_i,y_i)$

$d(p_i,p_j) = ((x_i-x_j)^2+(y_i-y_j)^2)^{1/2}$

Output: Points p and q that are closest



Closest pairs of points

$O(n^2)$ time algorithm?

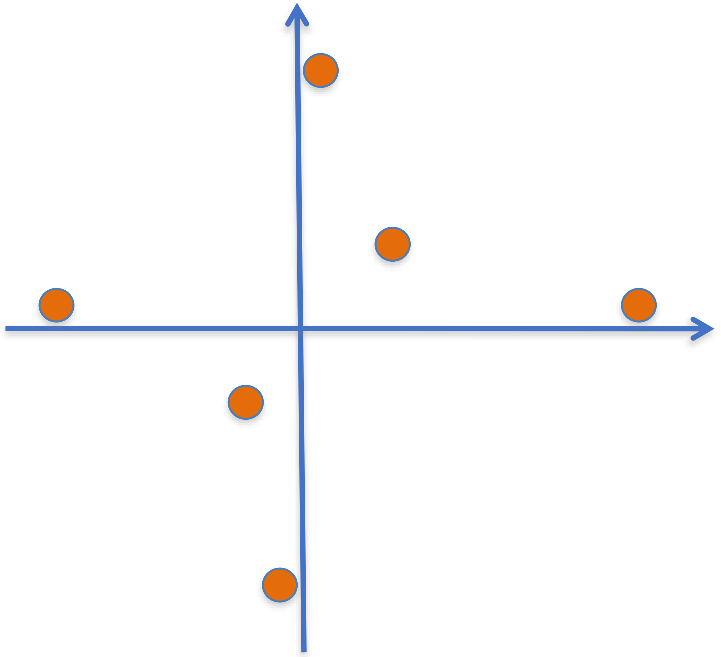
1-D problem in time $O(n \log n)$?

Sorting to rescue in 2-D?

Pick pairs of points closest in x co-ordinate

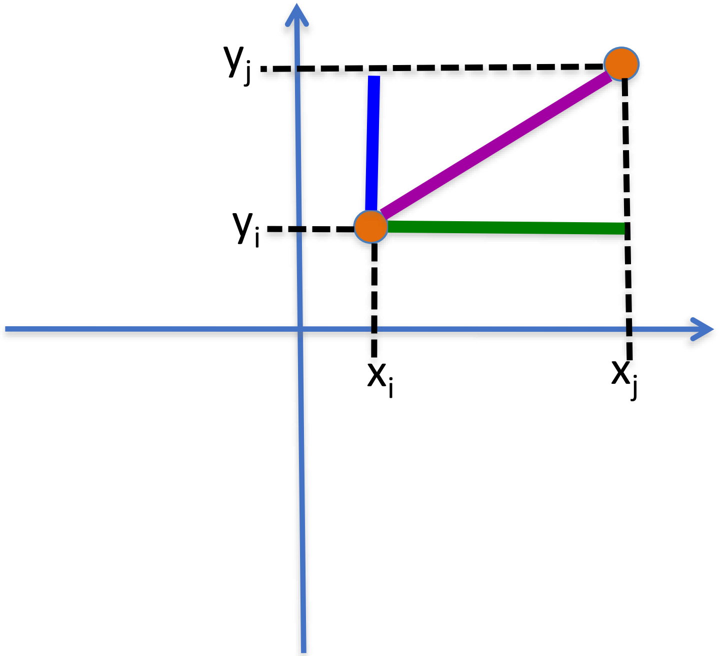
Pick pairs of points closest in y co-ordinate

Choose the better of the two



A property of Euclidean distance

$d(p_i,p_j) = ((x_i-x_j)^2+(y_i-y_j)^2)^{1/2}$



The distance is larger than the x or y-coord difference

Problem definition on the board…