Day 11

DIVIDE AND CONQUER

CLOSEST PAIR

Review of big-oh, big-omega and big-theta

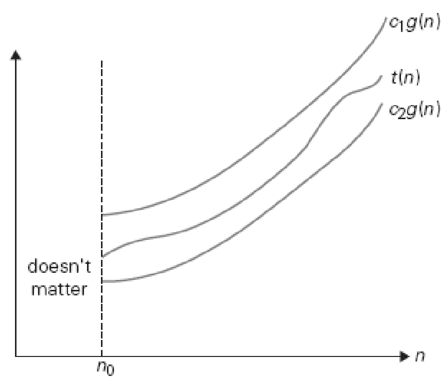


FIGURE 2.3 Big-theta notation: $t(n) \in \Theta(g(n))$.

Review of big-oh, big-omega and big-theta

Big-O (O):
$T(n) <= cg(n)$ for all $n >= n_0$

Big-Omega (Ω):
$T(n) <= cg(n)$ for all $n >= n_0$

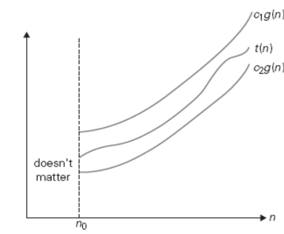


FIGURE 2.3 Big-theta notation: $t(n) \in \Theta(g(n))$.

Big-Theta (Θ):
$c_1g(n) <= T(n) <= c_2g(n)$ for all $n >= n_0$

Example: Show ½n(n-1) is $\Theta(n^2)$

Divide and conquer

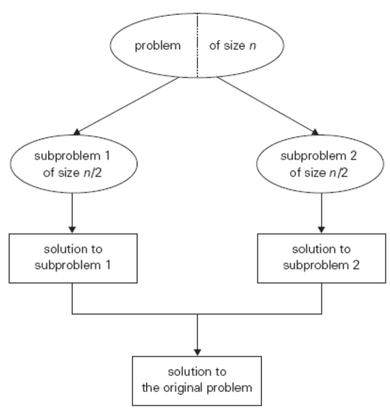


FIGURE 5.1 Divide-and-conquer technique (typical case).

Divide and conquer

Why is *quicksort* a divide and conquer algorithm?

Why is *mergesort* a divide and conquer algorithm?

Closest-Pair Problem

Find the two closest points in a set of *n* points.

Example applications:

- Film/music recommendations
- Air traffic control
- Clustering in general

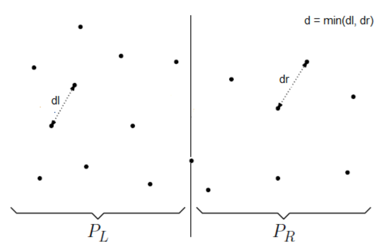
Closest-Pair Problem

- Brute-force algorithm (2D case)

ALGORITHM *BruteForceClosestPair*(P)
//Finds distance between two closest points in the plane by brute force
//Input: A list P of n ($n \geq 2$) points $p_1(x_1, y_1), \ldots, p_n(x_n, y_n)$
//Output: The distance between the closest pair of points
$d \leftarrow \infty$
for $i \leftarrow 1$ **to** $n - 1$ **do**
  for $j \leftarrow i + 1$ **to** n **do**
    $d \leftarrow \min(d, sqrt((x_i - x_j)^2 + (y_i - y_j)^2))$ //*sqrt* is square root
return d

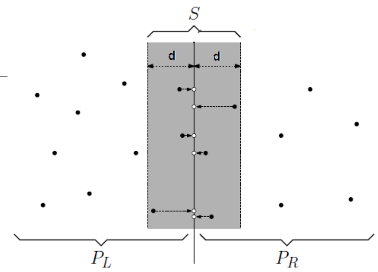
- What to count?
- Can you think of an optimization to the algorithm?

Divide-and-Conquer Closest Pair



1. Sort points by x-coordinate
2. Then sort by y-coordinate
3. Let *m* be the median of x-coordinates
4. Let P_L be the points to the left side of *m*, including *m*
5. Let P_R by the points to the right side of *m*
6. Recursively find the closest pair among P_L and P_R
7. Let d_L be the minimum in P_L
8. Let d_R be the minimum in P_R
9. Let *d* be the minimum of d_L and d_R

Divide-and-Conquer Closest Pair



Not quite done.

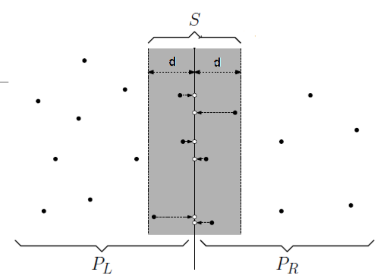
Need to look for potentially shorter pairs between P_L and P_R

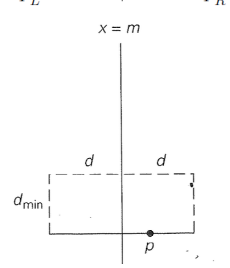
We only need to consider a strip S centered on m.

The width of S is $2d$.

Divide-and-Conquer Closest Pair

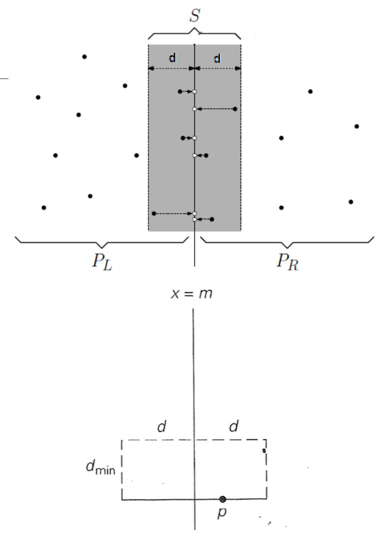
To further improve efficiency, for each point in S, we only need to consider points that are at most d away on the y-axis.





Divide-and-Conquer Closest Pair

How many points can be in the lower rectangle?



Divide-and-Conquer Closest Pair

How many points can be in each square?

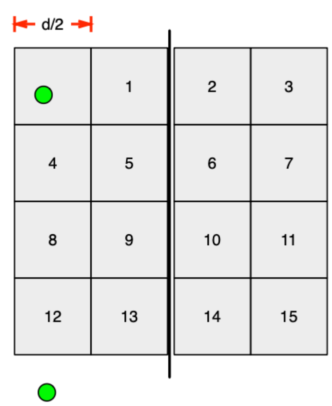


Image source: Carl Kingsford, https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/closepoints.pdf

Divide-and-Conquer Closest Pair Runtime

Sorting: 2*n log(n)

Splitting: n

$T(n) = 2\,T(n/2) + f(n)$

Master theorem: $a = 2$, $b = 2$, $f(n) = n^1$, i.e. $d = 1$

$a = b^d$

$T(n) = \Theta(n^d \log n)$, i.e. $\Theta(n \log n)$