CME 323: Distributed Algorithms and Optimization, Spring 2016
`http://stanford.edu/~rezab/dao`.
Instructor: Reza Zadeh, Matroid and Stanford. TAs: Andreas Santucci and Nolan Skochdopole

Lecture 6, 4/13/2016. Scribed by Stephen Kline and Kevin Shaw.

Contents

We cover the following items in this lecture:

- Review - Master Theorem
- Review - Algorithm Analysis (Finding Closest Pair of Points)

Master Theorem

$T(n) = a\,T(\frac{n}{b}) + f(n)$, $a \geq 1$ (# times called), $b \geq 1$ (amount divided)

Cases:

- Case 1 ($c < \log_b(a)$)
  - $f(n) \in O(n^c)$, where $c < \log_b(a)$, i.e. the work we do outside the recursion is less than work in recursion.
  - $\implies T(n) = \Theta(n^{log_b(a)})$
- Case 2 ($c = \log_b(a)$)
  - $f(n) \in O(n^c \log^k n)$, where $k \geq 0$, $c = \log_b(a)$, i.e. the work we do outside the recursion is on par with work in recursion.
  - $\implies T(n) = \Theta(n^c \log^{k+1} n)$
- Case 3 ($c > \log_b(a)$)
  - $f(n) \in \Omega(n^c)$, where $c > \log_b(a)$, i.e. the work we do outside the recursion is greater than work in recursion. This means total work in children should be less than parent.
  - and also, $a\,f(\frac{n}{b}) \leq k\,f(n)$ for $k < 1$, i.e. regularity condition holds.
  - $\implies T(n) \in \Theta(f(n))$

Example Problems:

- EX 1
  - $T(n) = 2\,T(\frac{n}{2}) + O(n)$

  - $a = 2, b = 2, c = 1, k = 0, \log_2(2) = 1 = c \Longrightarrow$ Case 2 $\Longrightarrow T(n) = O(n$
    $log(n))$

- EX 2
  - $T(n) = \frac{1}{2}T(\frac{n}{4}) + O(1)$
  - Master theorem does not apply. The $\frac{1}{2}$ factor doesn't make sense.

- EX 3
  - $T(n) = T(\frac{n}{2}) + 2^n$
  - Exponential outside the recursive call $\Longrightarrow$ Case 3 $\Longrightarrow T(n) = O(2^n)$

- EX 4
  - $T(n) = 16\,T(\frac{n}{4}) + n$
  - $a = 16, b = 4, c = 1, \log_4(16) = 2 > 1 \Longrightarrow$ Case 1 $\Longrightarrow T(n) = O(n^2)$

- EX 5
  - $T(n) = 3\,T(\frac{n}{2}) - n^2$
  - Master theorem does not apply. Cannot do negative work.

- EX 6
  - $T(n) = 2\,T(\frac{n}{2}) + (\frac{n}{n\ \log(n)})$
  - Might consider Case 1. However, f(n) is not poly. Master theorem does not apply.
  - Linear trumps log: Compare $\frac{(\frac{n}{\log(n)})}{n^{\log_2(2)}} = \frac{1}{(\log(n))} < n^{\epsilon}$, for any $\epsilon > 0$

- EX 7
  - $T(n) = 2\,T(\frac{n}{4}) + n^{0.51}$
  - $a = 2, b = 4, c = 0.51, \log_2(4) = \frac{1}{2} < c \Longrightarrow$ Case 3 $\Longrightarrow T(n) = O(n^{0.51})$

- EX 8
  - $T(n) = 2\,T(\frac{n}{4}) + \cos(n)$
  - Master theorem does not apply. Periodicity of cos(n) invalidates regularity condition.

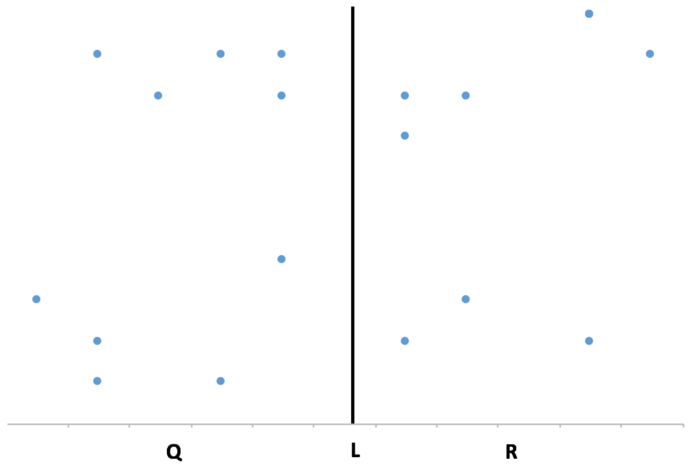
- EX 9
  - $T(n) = 2\,T(\frac{n}{2}) + \log(n)$
  - $f(n)$ is big-O and so Case 1 still applies. Plug in any $c < 1$ and show $n^c < \log(n)$
  - $a = 2, b = 2, c < 1, \log_2(2) = 1 > c \Longrightarrow$ Case 1 $\Longrightarrow T(n) = O(n)$

Algorithm Analysis (Finding Closest Pair of Points)

Closest Pair Problem
Given n points on a plane (2D), output the pair of points that are closest (e.g. L2, Euclidean)
$P_i = (x_i, y_i), i = 1..n$, assume distinct points (x_i, y_i) (algorithm can be tweaked otherwise)

Naive: check all pairs and choose minimum, $O(n^2)$
Lower Bound: $\Omega(n)$

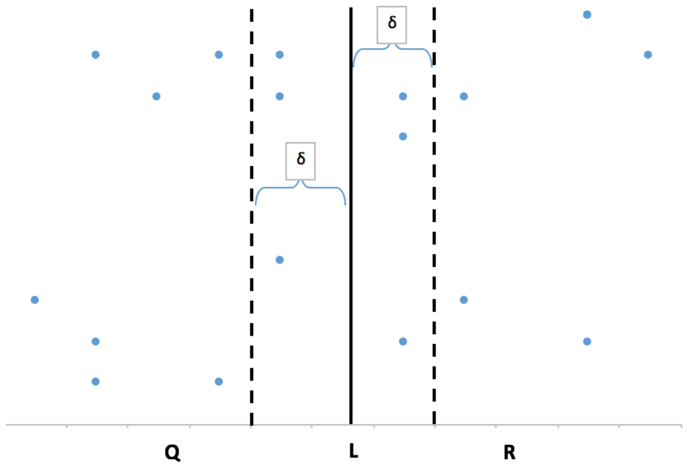
Figure 1:



Basic steps:
Sort by x coord and split on median
Problem scenario: closest pair is $q \in Q, r \in R$

Assume you have a way to check points across median:
$\delta_Q = mindistance(q_1, q_2 \in Q)$
$\delta_R = mindistance(r_1, r_2 \in R)$
$\delta = min(\delta_Q, \delta_R)$

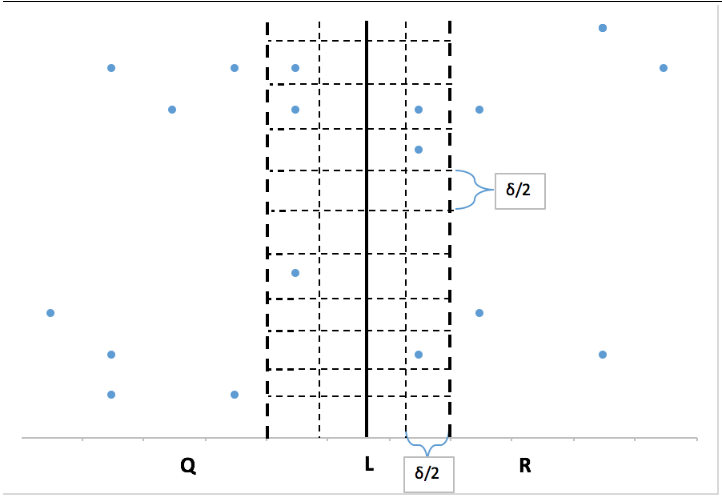
Figure 2:



Look at slab of width $2 * \delta$ centered at L
S is points in a slab, pairwise checking $O(n^2)$

For each point $p \in S$, only need to check $O(1)$ points

Figure 3:



Claim:
There is at most 1 point in each box

Proof:
Suppose not...

$q_1, q_2 \in$ box
$\delta(q_1, q_2) \leq (\frac{\delta}{2})\sqrt{(}2) < \delta \implies$ contradiction

Back to Figure 3 (above)...
Suppose we have a point we need to check. Anything 3 "rows" beyond has distance $\geq \frac{3}{2}\delta > \delta$, so we do not need to check anything beyond those "rows". Because there is at most one point per box, we only need to check at most 27 points for each point (at most 12 in the rows above, at most 3 in the same row and at most 12 in the rows below). Note that this analysis can be made tighter, but we are happy with just showing $O(1)$ work right now.

Idea: Sort L by y coordinate (can be done ahead of time), S_y.
For each point in S, check next 15 points in list. We are still making all the checks, we just are getting rid of some of the replicated work. Now for each point we only do constant work.

Psuedo-Code:

```
**Closest-Pair**($P$)
Construct $P_x$ and $P_y$ ($O(n\log(n))$ time)
($p_0^*$, $p_1^*$) = Closest-Pair-Rec($P_x$, $P_y$)

**Closest-Pair-Rec**($P_x$, $P_y$)
**if** $|P| \leq 3$ **then**
   find closest pair by measuring all pairwise distances
**end if**

Construct $Q_x$, $Q_y$, $R_x$, $R_y$ ($O(n)$ time)
($q_0^*$, $q_1^*$) = Closest-Pair-Rec($Q_x$, $Q_y$)
($r_0^*$, $r_1^*$) = Closest-Pair-Rec($R_x$, $R_y$)

$\delta = min(d((q_0^*, q_1^*)), d(r_0^*, r_1^*)))$
$x^*$ = maximum x-coordinate of a point in set Q
$L = \{(x, y) : x = x^*\}$
$S$ = points in $P$ within distance $\delta$ of $L$.

Construct $S_y$ ($O(n)$ time)
For each pint $s \in S_y$, compute distance from $s$
to each of next 15 points in $S_y$
Let $s, s'$ be pair achieving minimum of these distances
($O(n)$ time)
```

```
if $d(s, s') < \delta$ then
    Return $(s, s')$
else if $\mathrm{d}(q_0^*,\ q_1^*) < \mathrm{d}(r_0^*,\ r_1^*)$ then
    Return $(q_0^*, q_1^*)$
else
    Return $(r_0^*, r_1^*)$
end if
```

Analysis
Pre-Work: $O(n\ \log(n))$, due to the sort
Recursion Work: $T(n) = 2\,T(\frac{n}{2}) + O(n)$ due to the pre-sort, touching all elements
$\implies O(n\ \log(n))$

References

[1] J. Kleinberg and E. Tardos. *Algorithm Design.* Pearson Education, Inc., 2006.