Outline

Wrap up MergeSort complexity

Divide and Conquer

Using the Master Theorem

Notes

Recall MergeSort

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 2n + 1 & \text{if } n > 1 \end{cases}$$

Last time: $T(n) \in \Omega(n \lg n)$

```
MergeSort(A,b,e):
    if b == e:  return
    m = (b + e) / 2
    MergeSort(A,b,m)
    MergeSort(A,m+1,e)
    # merge sorted A[b..m] and A[m+1..e] back into A[b..e]
    for i in [b,...,e]:  B[i] = A[i]
    c = b
    d = m+1
    for in [b,...,e]:
        if d > e or (c <= m and B[c] < B[d]):
            A[i] = B[c]
            c = c + 1
        else:  # d <= e and (c > m or B[c] >= B[d])
            A[i] = B[d]
            d = d + 1
```

Merge Sort

Upper bound on $T(n)$

Now: $T(n) \in O(n \lg n)$

Assume, by same trial + error,

$T(n) \leq 2 \cdot n \cdot \lg(n-1) + 4n - 1$

By C.I.:

Inductive Step: Let $n \in \mathbb{N}$, $n \geq 2$.

Assume $H(n)$: $\forall i \in \mathbb{N}$, $2 \leq i < n$, $T(i) \leq 2i \cdot \lg(i-1) + 4i - 1$

Show $H(n) \rightarrow C(n)$: $T(n) \leq 2n \lg(n-1) + 4n - 1$

Let $n \geq 4$

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 2n + 1$$ (by def. of $T(n)$)

$$\leq 2 \cdot \lceil n/2 \rceil \cdot \lg(\lceil n/2 \rceil - 1) + 4 \cdot \lceil n/2 \rceil - 1 + 2 \cdot \lfloor n/2 \rfloor \cdot \lg(\lfloor n/2 \rfloor - 1) + 4 \cdot \lfloor n/2 \rfloor - 1 + 2n + 1$$

(by $H(n)$ b/c $2 \leq \lfloor n/2 \rfloor, \lceil n/2 \rceil < n$ since $n \geq 4$)

(by $\lfloor n/2 \rfloor - 1 \leq \lceil \frac{n}{2} \rceil - 1 \leq \frac{n-1}{2}$)

$$\leq 2 \cdot \lg\left(\frac{n-1}{2}\right) \cdot \underbrace{\left(\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor\right)}_{n} + 4 \cdot \left(\left\lceil \frac{n}{2} \right\rceil + \lfloor n/2 \rfloor\right) + 2n - 1$$

$$= 2n\left(\lg\left(\frac{n-1}{2}\right) + 1\right) + 4n - 1$$

Merge Sort

Upper bound on $T(n)$

$$= 2n\left(\lg(n-1) - \cancel{\lg 2} \cancel{+1}\right) + 4n - 1$$

$$= 2n\lg(n-1) + 4n - 1$$

So $C(n)$ holds. (for $n \geq 4$)

Remaining: $n=2$ & $n=3$

$$T(2) = \overset{1}{T(1)} + \overset{1}{T(1)} + \overset{4}{2(2)} + 1 = 7 \quad (7 \leq 7) ✓$$

$$2 \cdot 2 \cdot \lg(1) + 4 \cdot 2 - 1 = 7$$

$$T(3) = \overset{7}{T(2)} + \overset{1}{T(1)} + 2 \cdot 3 + 1 = 15 \quad (15 \leq 17) ✓$$

$$2 \cdot 3 \cdot \lg(3-1) + 4 \cdot 3 - 1 = 17$$

So we can conclude $T(n) \leq 2n\lg(n-1) + 4n - 1$
$\forall n \geq 2$

So $T(n) \in O(n\lg n)$.

Divide and Conquer: General Case

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

boundary

# of subprob

b ~~# of~~ size of subproblems

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

where $B, k > 0$, $b > 1$, $a_1, a_2 \geq 0$, and $a = a_1 + a_2 > 0$. $f(n)$ is the cost of splitting and recombining.

Master Theorem

cost of splitting/combining is a polynomial

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

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

Applying the Master Theorem

MergeSort

$$T(n) = \begin{cases} 1 & n=1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \underbrace{2n+1}_{f(n)} & n>1 \end{cases}$$

a (# of subproblems) $\rightarrow 1+1=2$

b (size of subprob) $\rightarrow 2$

d (exponent from splitting/combining) $\rightarrow 1$

$a = 2$

$b^d = 2$

So by Master Theorem

$\Theta(n^d \lg n) \rightarrow \Theta(n \lg n)$

$T(\lceil \frac{n}{8} \rceil) + 2T(\lfloor \frac{n}{3} \rfloor)$

$b = 3$

$a = 3$

Applying the Master Theorem

RecBinSearch

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ \underbrace{1}_{f(n)} + \max\left(T(\lceil n/2 \rceil), T(\lfloor n/2 \rfloor)\right) & n > 1 \end{cases}$$

$a = ? \rightarrow 1$

$b = ? \rightarrow 2$

$d = ? \rightarrow 0$

$b^d = 2^0 = 1$

$T(n) \in \Theta(n^d \lg n)$

$\rightarrow \Theta(\lg n)$

Closest Point Pairs

see Wikipedia

Given points $p_1 = (x_1, y_1), \ldots, p_n = (x_n, y_n)$, find the pair p_i, p_j s.t. $d(p_i, p_j)$ is minimal.

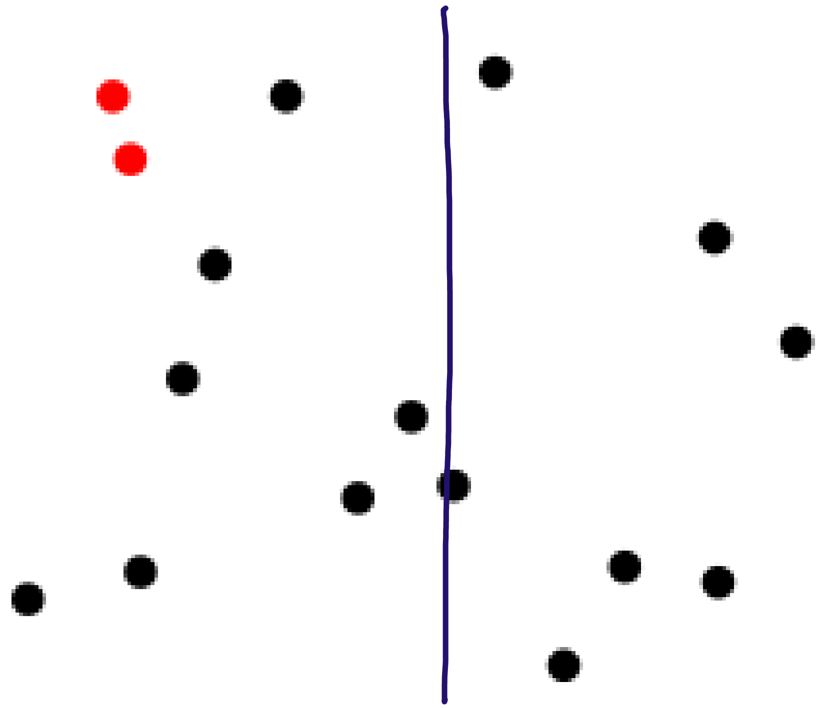
↳ distance from p_i to p_j

<u>Brute force alg:</u>

Compare all pairs, and find min. dist.

$\rightarrow \Theta(n^2)$

$\binom{n}{2}$ comparisons to make.



Assume all points are distinct.

$$T(n) = 2 \cdot T\left(\tfrac{n}{2}\right) + n^d$$

By MT, we can do better if $d < 2$

Dividing and Conquering ClosestPointPairs

Try to do better using D+C.

Need to be able to split/combine in $\Theta(n^d)$, $d<2$ (ie. $\in o(n^2)$)

Consider P: a set of points

P_x : P sorted by x-coord

P_y : P sorted by y-coord.

Assume P_x/P_y data structure has cross-references

To D+C: Split P vertically (along y-axis) into

Q : leftmost $n/2$ points

R : rightmost $n/2$ points

Note: Q_x, Q_y, R_x, R_y from P_x, P_y in linear time

Recursively find

$q_0, q_1 \rightarrow$ closest points in Q

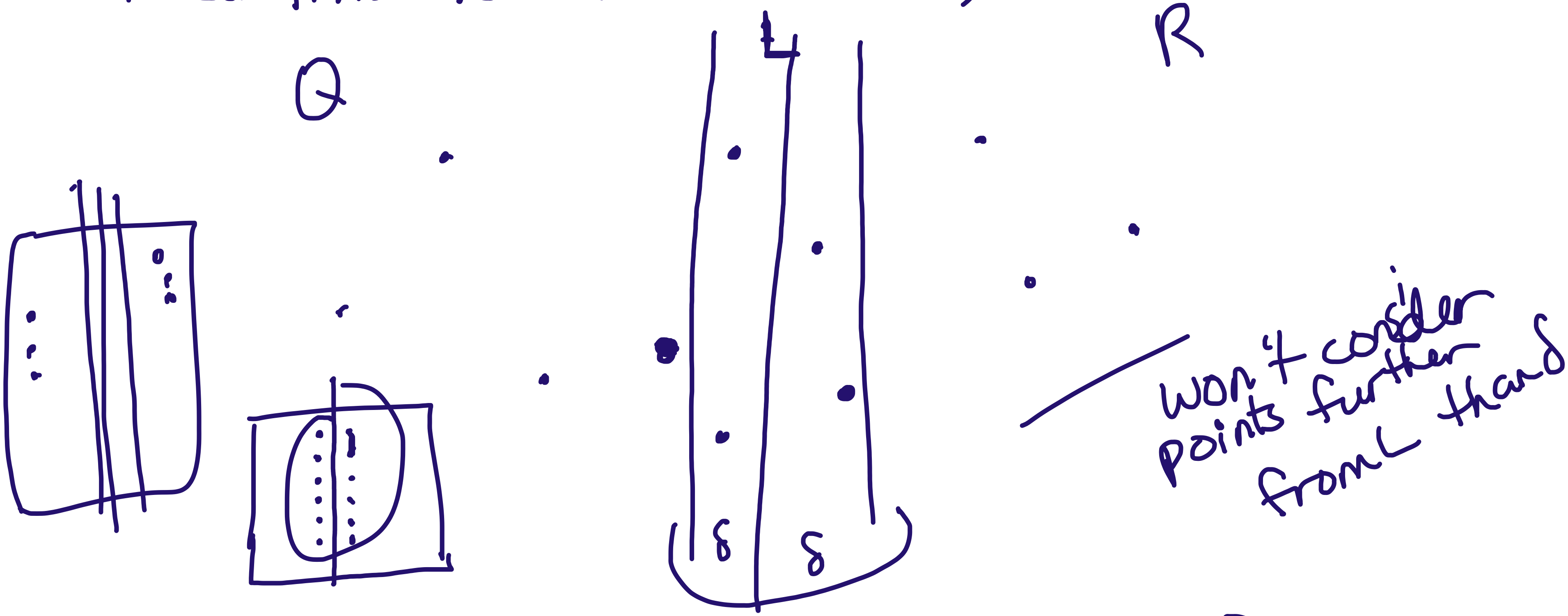
$r_0, r_1 \rightarrow$ closest in R

Dividing and Conquering ClosestPointPairs

Combine: Let $\delta = \min(d(q_0,q_1), d(r_0,r_1))$

Want to find $q \in \underline{\underline{Q}}, r \in R$, s.t. $d(q,r) < \delta$, if any.

Need this to not be $\Theta(n^2)$.



Let $S = \{$points that are $< \delta$ from $L\}$

S_x, S_y in linear time

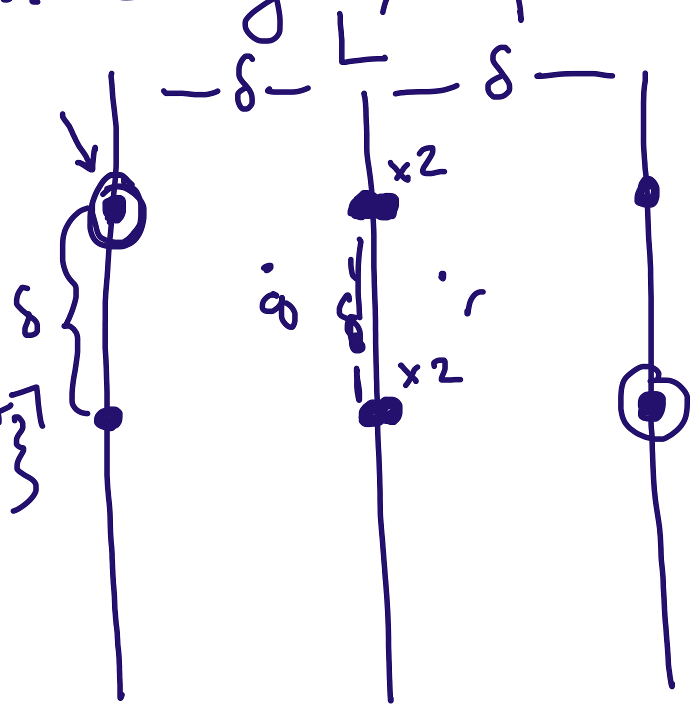
Dividing and Conquering ClosestPointPairs

Problem: It looks like we might have to check $\underline{\underline{\Theta(n^2)}}$ pairs to combine...

Actually... only 7 per point $\rightarrow$ $\underline{7n}$

Q

R



$S_y = \{, \ldots, p, \text{next } 7 \ldots\}$

8 points, including the one we're looking at.

7 is enough!

Dividing and Conquering ClosestPointPairs

Dividing and Conquering ClosestPointPairs

Dividing and Conquering ClosestPointPairs

Algorithm for ClosestPointPairs

Assuming P_x, P_y are sorted $\rightarrow \Theta(n \lg n)$

ClosestPairRec(P_x, P_y):

    if |P| <= 3:

        find closest points by brute force   k

    else:

        construct Q_x, Q_y, R_x, R_y $\Theta(n)$

        (q_0,q_1) = ClosestPairRec(Q_x, Q_y) $T(\frac{n}{2})$

        (r_0,r_1) = ClosestPairRec(R_x, R_y) $T(\frac{n}{2})$

c δ = min(d(q_0,q_1), d(r_0,r_1))

        ~~m~~ L = average of rightmost x-coordinate in Q

                and leftmost x-coordinate in R

        construct S_x, S_y $\Theta(n)$

        for each s $\in$ S_y: $\Theta(n)$

            compute distance to next ~~15~~ 7 points in S_y

              and let (s_0,s_1) be closest pair found

15 is in both directions

Recurrence for ClosestPointPairs

Recurrence?

$$T(n) = \begin{cases} k & n \le 3 \\ 2T(\frac{n}{2}) + \underline{\Theta(n)} & n > 3 \end{cases}$$

$\Theta(n^d)$

Apply MT:

$a \to 2$

$b \to 2$ $\qquad 2 = 2^1$

$d \to 1$

By MT, $T(n) \in \Theta(n \lg n)$.

Applying the Master Theorem

If combining was $\Theta(n^2)$

$a = 2$ $\quad d = 2 \quad$ $2 < 2^2$

$b = 2$

So $\Theta(n^2)$ instead

of $\Theta(n \lg n)$.

Notes