2/17/15: Divide & Conquer

1. divide into sub-problems (barely overlapping)
2. combine together

MergeSort

Merge

input: $c_1, ..., c_m, d_1, .., d_n$
init: $i = 1, j = 1$
begin
  while $i \leq m$ and $j \leq n$:
    _if_ $c_i \leq d_j$:
      append c_i to b
      $i = i + 1$
    else
      append d_j to b
      $j = j + 1$
  end while

  _if_ $i = m + 1$:
    append $d_j, ..., d_n$ to b
  else
    append $c_i, ..., c_m$ to b
end

MergeSort

input: $a_1, ..., a_n$
begin
  _if_ $n - 2$:
    return $\text{sort}(a_1, a_2)$
  else:
    $c_1, ..., c_{\frac{n}{2}} = MergeSort(a_1, ..., a_{\frac{n}{2}})$
    $d_1, ..., d_{\frac{n}{2}} = MergeSort(a_{\frac{n}{2}}, ..., a_n)$
    return $Merge(c, d)$
end

Time: $O(n + m)$

Prove by induction that $T(n) \leq cnlog_2(n)$

- Base Case: $n = 2$
  - we know $T(2) \leq c \leq c2log_2(n) = c$
  - ✓

- Induction Step: assume True for $\frac{n}{2}$

$$T(n) \leq 2T(\frac{n}{2}) + cn$$

1. $\leq 2c(\frac{n}{2})log_2(\frac{n}{2}) + cn$ (by induction)
2. $= cn(log_2(n) - 1) + cn$
3. $= cnlog_2(n) - cn + cn = cnlog_2(n)$
4. ✓

Example: Inversions

- **def**: an *inversion* is a pair $i < j$ s.t. $a_i > a_j$
- **def:** a *cross-inversion* is an $i \leq \frac{n}{2}, j > \frac{n}{2}$ with $a_i > a_j$

My rank	d	a	c	b	e
James	c	a	e	d	b

- count # of inversions - pairs out of order: *5 inversions*
- rename s.t. my order is $1, 2, 3, ...$ and other order is $a_{1},a_{2
- yeilds:

My rank	1	2	3	4	5
James	3	2	5	1	4

- $w =$ CountInversions$(a_1, ..., a_n)$
- $x =$ CountInversions$(a_1, ..., a_{\frac{n}{2}})$
- $y =$ CountInversions$(a_{\frac{n}{2}+1}, ..., a_n)$
- $z =$ CountCrossInversions$(a_1, ..., a_{\frac{n}{2}}, a_{\frac{n}{2}+1}, ..., a_n)$

$$w = x + y + z$$

- here, we are going to compute more than we need to make our problem easier $->$*Modified Merge Sort*
- sort and count number of inversions

$(b_1, ..., b_n)$	$w =$ CountInversions(a_1, a_n)
$(c_1, ..., c_{\frac{n}{2}})$	$x =$ CountInversions$(a_1, ..., a_{\frac{n}{2}})$
$(d_1, ..., d_{\frac{n}{2}})$	$y =$ CountInversions$(a_{\frac{n}{2}}, ..., a_n)$
$(b_1, ..., b_n)$	$y =$ CountInversions$(c_1, ..., c_{\frac{n}{2}}, d_1, ..., d_{\frac{n}{2}})$

$$(b_1, ..., b_n), 2$$

CountCrossInversions

```
input: c_1, ..., c_m, d_1, ..., d_n
init: i = j = 1, z = 0
begin
  while i ≤ m ∧ j ≤ n:
    if c_i ≤ d_j
      append c_i to b
      i = i + 1
    else
      append d_j to b
      j = j + 1
      z = z + m − i + 1
  endwhile

  if i = m + 1
    append d_j, ..., d_n to b
  else
    append c_i, ..., c_m to b

  return b, z
end
```

Closest Pair

- **input:** $(x_1, y_1), ..., (x_n, y_n)$
  - assume x_i's are distinct
- **init:** $i = 1, j = 1$
- **strategy**
  - find i and j s.t. $dist((x_i, y_i), (x_j, y_j))$ is as small as possible
  - can compare all pairs, pick smallest in $O(n^2)$
  - we will do this in $O(n \log n)$

Divide plane into right (R) and left (L)

1. sort so $x_1 < x_2 < ... < x_n$
2. set $L = \{1, ..., \frac{n}{2}\}$, $R = \{\frac{n}{2} + 1, ..., n\}$
3. Find closest pair in L. Let δ_L be its distance.
4. Find closest pair in R. Let δ_R be its distance.
5. $\delta = min(\delta_L, \delta_R)$
   - only care about closest pair bridging L and R
6. will find closes L/R pair if its distance is $< \delta$
   - only need to look at points within δ of center line
   - only need to compare $\frac{L}{R}$ pairs within δ of y-coordinate

- sort points on y-coordinates
- for each point in L, only need to compare to the 10 points in R with closest y components
- linear time to compare L,R
- b/c each point compared to only 10 others

Idea

- we only need to look at points withing δ of the center line.
- we only need to compare L/R pairs within δ of y-coordinate.
- Sort points on y-coordinates.
- For each point in L, only need to compare to the 10 points in R with closest y-coordinates
- So, takes linear time to compare L and R.

Explanation

- at most 1 point per box.
- For every point on left, are at most 10 boxes on right with points at distance $\leq \delta$
- after sort, linear time to scan 10 proximal boxes, but sort take $log(n)$
- $T(n) \leq 2T(\frac{n}{2}) + cn\log_2(n)$
- $T(n) \leq cn(\log_2(n))^2)$
- but wait! can sort on y-coordinates up front (pay $n\log n$ once), and then keep track of order when split.
- $T(n) \leq 2T(\frac{n}{2}) + cn \leq cn\log_2 n$

Test on 2/24/15

- test will be split into 2 rooms
- content
  - memorization-heavy
  - 3 problems, pretty much straight out of class or PSETS
  - memorize problems from PSETS and class
  - follow directions carefully, test is one thing where no name is on every page. only first.