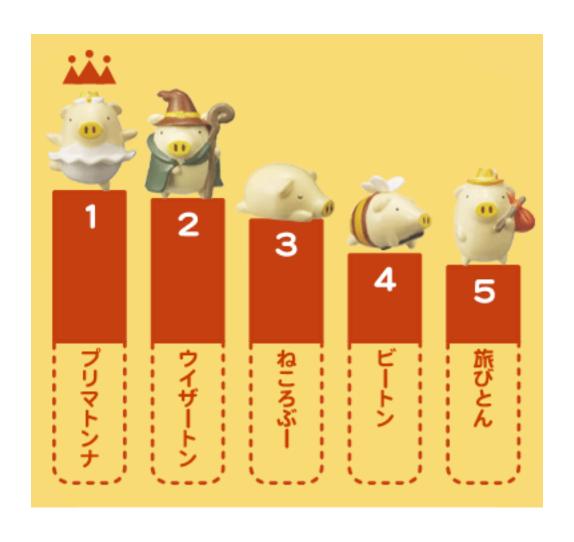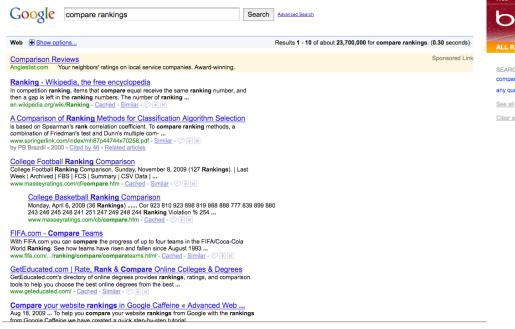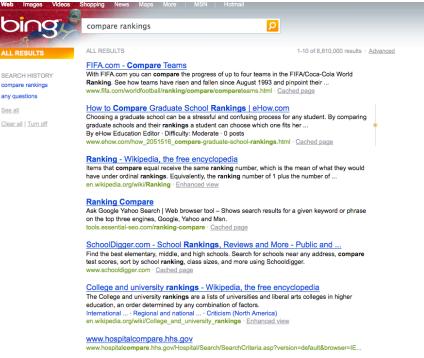# Lecture 24

CSE 331

Oct 25, 2019

Rankings

How close are two rankings?

Rest of today's agenda

Formal problem: Counting inversions

Divide and Conquer algorithm

Solve a harder problem

Input: $a_1, .., a_n$

Output: LIST of all inversions

```
L = φ
for i in 1 to n-1
    for j in i+1 to n
        If a_i > a_j
            add (i,j) to L

return  L
```

Optimal for the listing problem

Example 1: All inversions-- (2i-1,2i)

| 2 | 1 | 3 | 4 | 6 | 5 | 7 | 8 |

Only check (i,i+1) pairs

Q1: Solve listing problem in O(n) time?

Q2: Recursive divide and conquer algorithm to count the number of inversions?

CountInv (a,n)

    if n = 1 return 0

    if n = 2 return $a_1 > a_2$

    $a_L = a_1 , .., a_{[n/2]}$

    $a_R = a_{[n/2]+1} , .., a_n$

    return CountInv(a_L, [n/2]) + CountInv(a_R, n- [n/2])

This can be horribly wrong...

CountInv (a, n)

  if $n = 1$ return 0

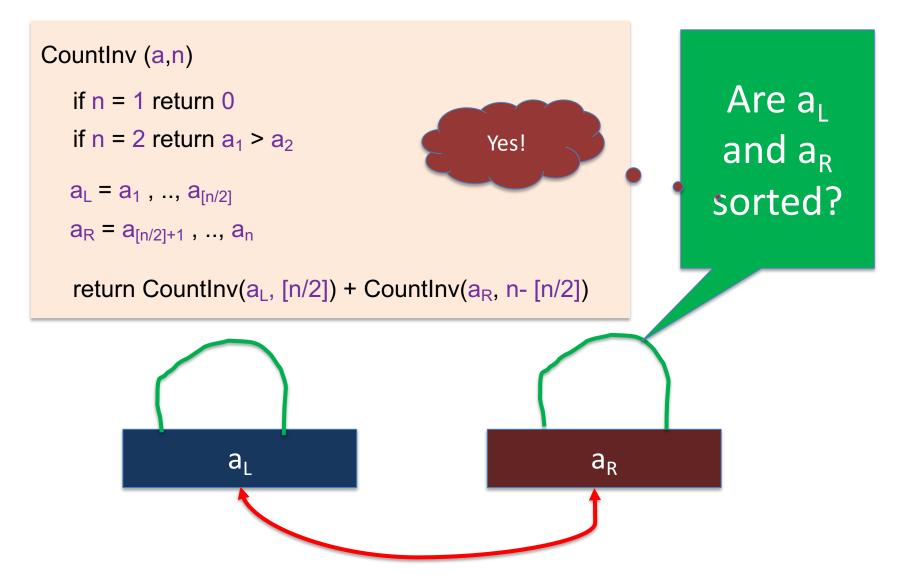 if $n = 2$ return $a_1 > a_2$

  $a_L = a_1$, .., $a_{[n/2]}$

  $a_R = a_{[n/2]+1}$, .., a_n

  return CountInv(a_L, $[n/2]$) + CountInv(a_R, $n- [n/2]$)

Example where instance has non-zero (can be $\Omega(n^2)$) inversions and algo returns 0?

| 5 | 6 | 1 | 2 |

All 4 "crossing" pairs are inversions

Bad case: "crossing inversions"

CountInv (a,n)

    if n = 1 return 0

    if n = 2 return $a_1 > a_2$

    $a_L = a_1 , .., a_{[n/2]}$

    $a_R = a_{[n/2]+1} , .., a_n$

    return CountInv(a_L, [n/2]) + CountInv(a_R, n- [n/2])

Yes!

Are a_L and a_R sorted?

a_L

a_R

Example 2: Solving the bad case

5	6	…..

a_L

1

a_R

a_L is sorted

First element is a_L is larger than first/only element in a_R

O(1) algorithm to count number of inversions?

return size of a_L

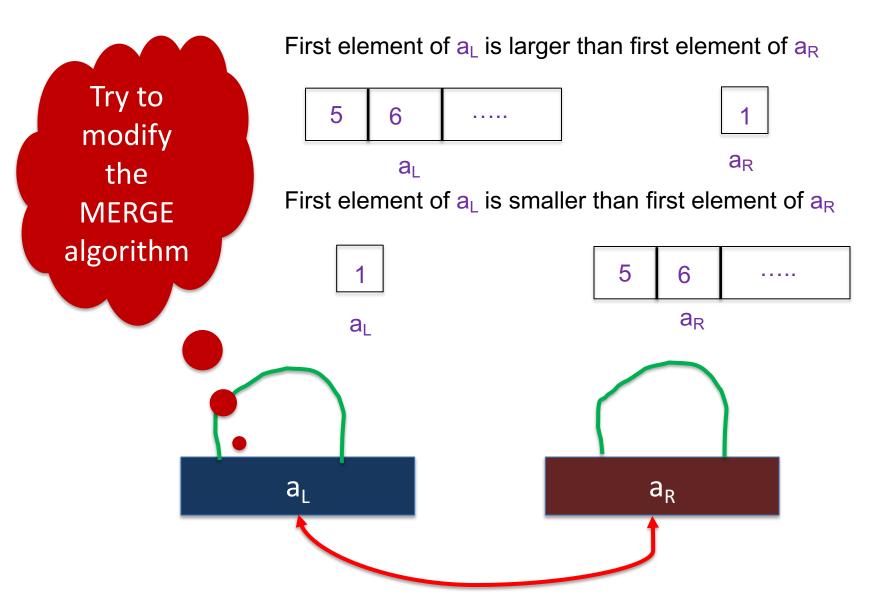# Example 3: Solving the bad case

1

a_L

5	6	…..

a_R

---

a_R is sorted

First/only element is a_L is smaller than first element in a_R

O(1) algorithm to count number of inversions?

---

return 0

Solving the bad case

First element of a_L is larger than first element of a_R

| 5 | 6 | ….. |

a_L

| 1 |

a_R

First element of a_L is smaller than first element of a_R

| 1 |

a_L

| 5 | 6 | ….. |

a_R

Try to modify the MERGE algorithm

a_L

a_R

Divide and Conquer

Divide up the problem into at least two sub-problems

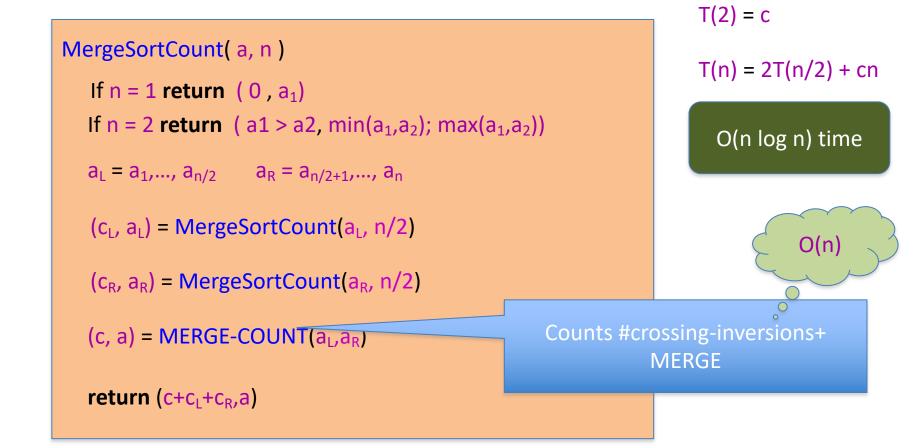Solve all sub-problems: Mergesort

Recursively solve the sub-problems

Solve stronger sub-problems: Inversions

"Patch up" the solutions to the sub-problems for the final solution
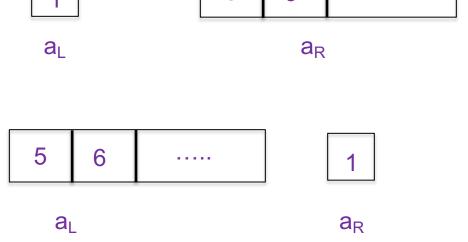
Mergesort-Count algorithm

Input: $a_1, a_2, ..., a_n$

Output: Numbers in sorted order+ #inversion

$T(2) = c$

$T(n) = 2T(n/2) + cn$

O(n log n) time

O(n)

MergeSortCount(a, n)

  If n = 1 **return** (0 , a_1)

  If n = 2 **return** (a1 > a2, min(a_1,a_2); max(a_1,a_2))

  $a_L = a_1,..., a_{n/2}$ $a_R = a_{n/2+1},..., a_n$

  (c_L, a_L) = MergeSortCount(a_L, n/2)

  (c_R, a_R) = MergeSortCount(a_R, n/2)

  (c, a) = MERGE-COUNT(a_L,a_R)

Counts #crossing-inversions+ MERGE

  return ($c+c_L+c_R$,a)

MERGE-COUNT(a_L, a_R)

$a_L = l_1, \ldots, l_{n'}$ $a_R = r_1, \ldots, r_m$

c = 0

i,j = 1

while i ≤ n' and j ≤ m

    if l_i < r_j

      i ++

      add l_i to output

    else

      add r_j to output

      j ++

      c += n'- i +1

Output any remaining items

return c

1

a_L

5	6	…..

a_R

5	6	…..

a_L

1

a_R