Lecture 25

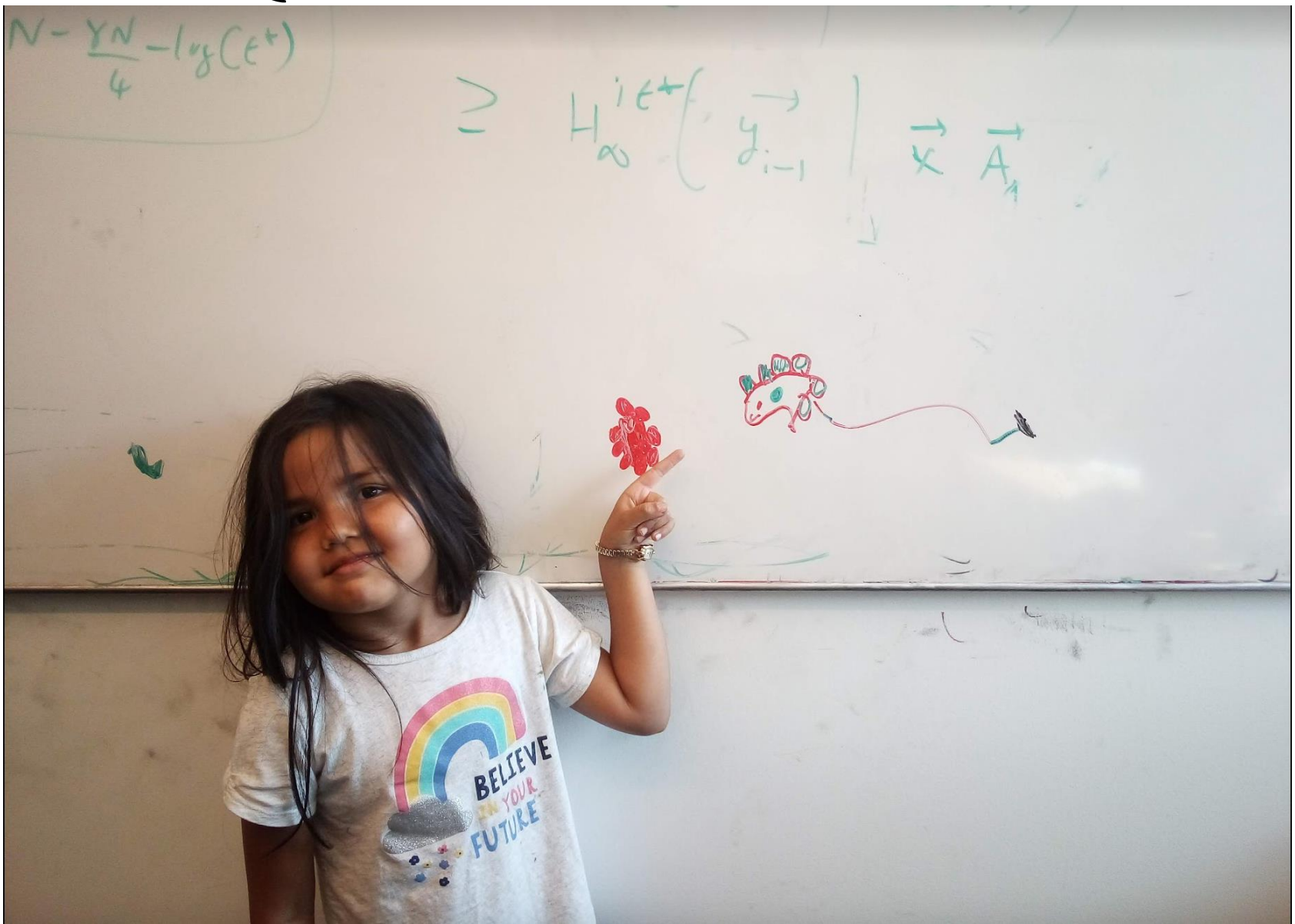
CSE 331

Oct 30, 2024

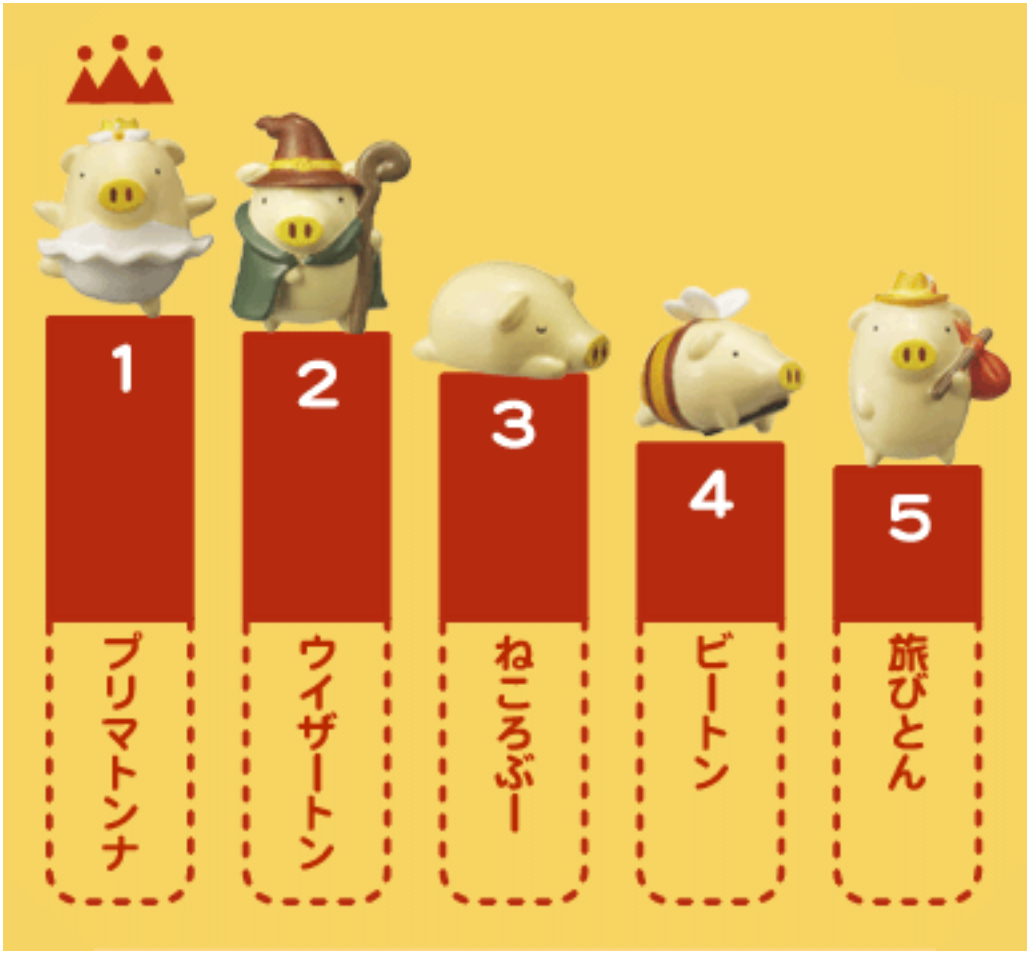
1st project deadline THIS Friday

Tue, Oct 15		**(HW 4 out)**
Wed, Oct 16	Dijkstra's algorithm F24 F23 F22 F21 F19 x²	[KT, Sec 4.4] Week 8 recitation notes
Fri, Oct 18	Correctness of Dijkstra's Algorithm F23 F22 F21 F19 F18 F17 x²	[KT, Sec 4.4] *Reading Assignment:* [KT, Sec 4.4]
Mon, Oct 21	Minimum Spanning Tree F23 F22 F21 F19 F18 F17 x²	[KT, Sec 4.5]
Tue, Oct 22		**(HW 4 in, HW 5 out)**
Wed, Oct 23	Cut Property Lemma F23 F22 F21 F19 F18 F17 x²	[KT, Sec 4.5] *Reading Assignment:* [KT, Sec 4.5, 4.6]
Fri, Oct 25	Mergesort F23 F22 F21 F19 F18 F17 x²	[KT, Sec 5.1]
Mon, Oct 28	Solving recurrence relations F23 F22 F21 F19 F18 F17 x²	[KT, Sec 5.1]
Tue, Oct 29		**(HW 5 in)**
Wed, Oct 30	Counting Inversions F23 F22 F21 F19 F18 F17 x²	[KT, Sec 5.3]
Fri, Nov 1	Multiplying large integers F23 F22 F21 F19 F18 F17 x²	[KT, Sec 5.5] **(Project (Problems 1 & 2 `Coding`) in)** *Reading Assignment:* Unraveling the mystery behind the identity
Mon, Nov 4	Closest Pair of Points F23 F22 F21 F19 F18 F17 x²	[KT, Sec 5.4] **(Project (Problems 1 & 2 `Reflection`) in)**

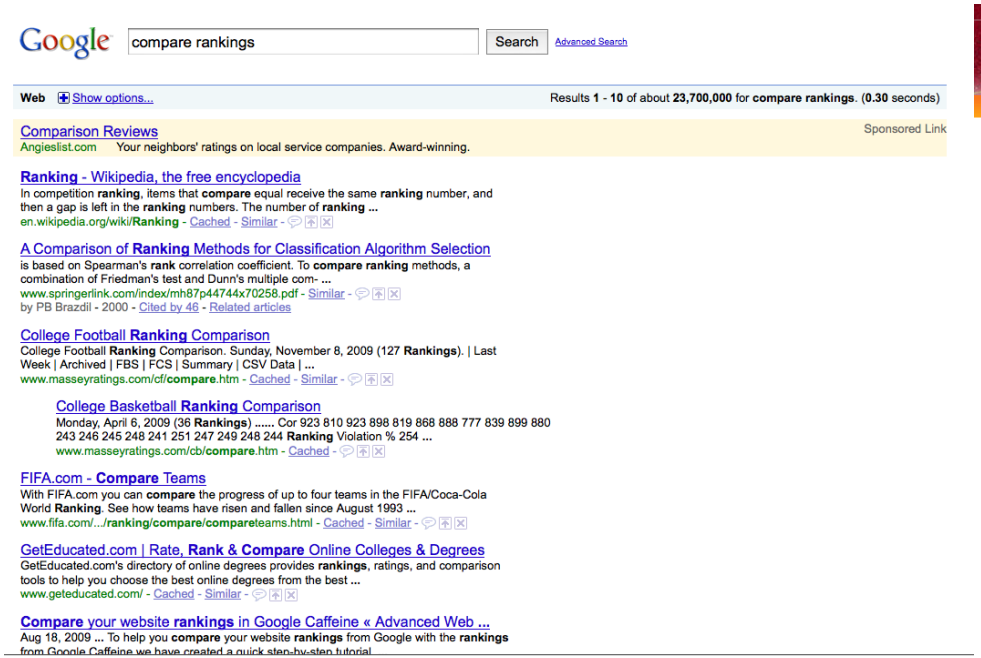
Questions/Comments?

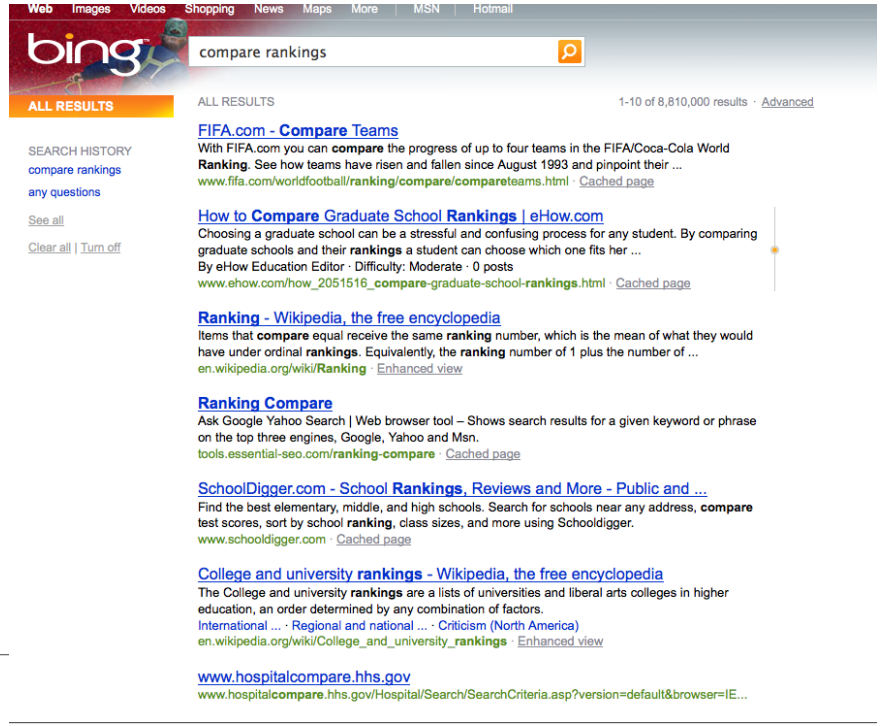


Rankings



How close are two rankings?





Rest of today's agenda

Formal problem: Counting inversions

Divide and Conquer algorithm

Problem definition on the board…

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
```

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

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])

Can be horribly wrong in general

CountInv (a,n)

if $n = 1$ return 0

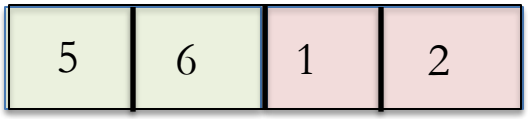
if $n = 2$ return $a_1 > a_2$

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

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

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?



All 4 "crossing" pairs are inversions

Bad case: "crossing inversions"

CountInv (a,n)

if n = 1 return 0

if n = 2 return $a_1 > a_2$

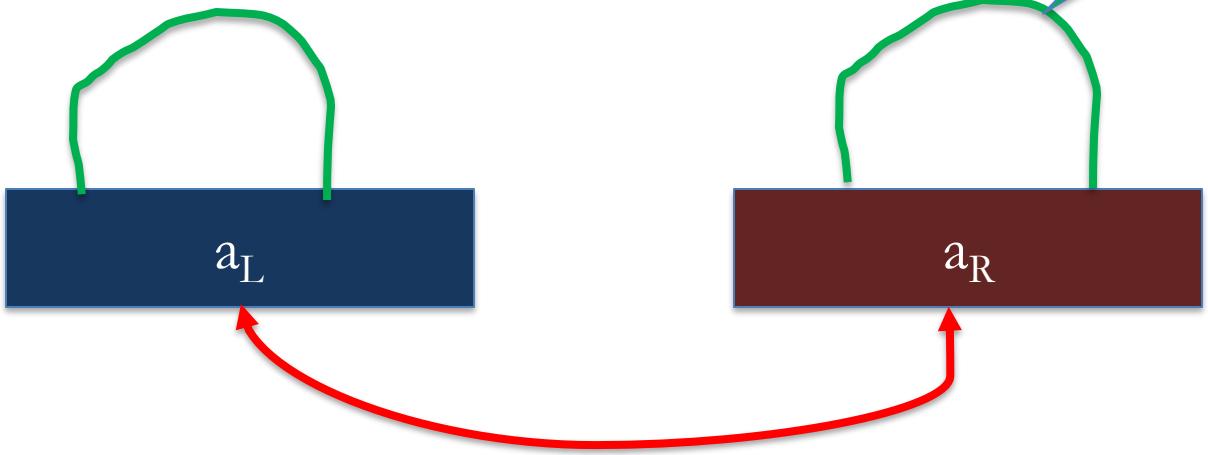
$a_L = a_1, .., a_{[n/2]}$

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

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

Yes!

Are a_L and a_R sorted?



Example 2: Solving the bad case

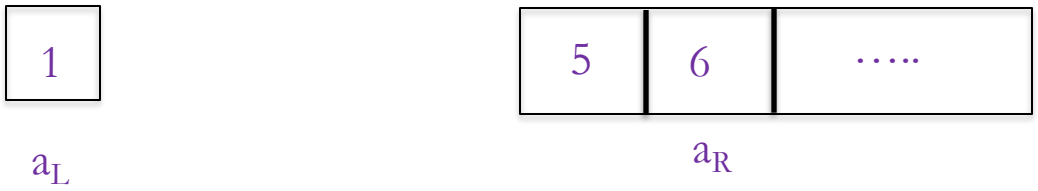
a_L is sorted

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



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?

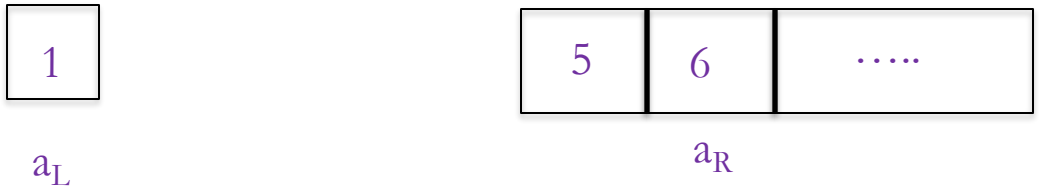
return 0

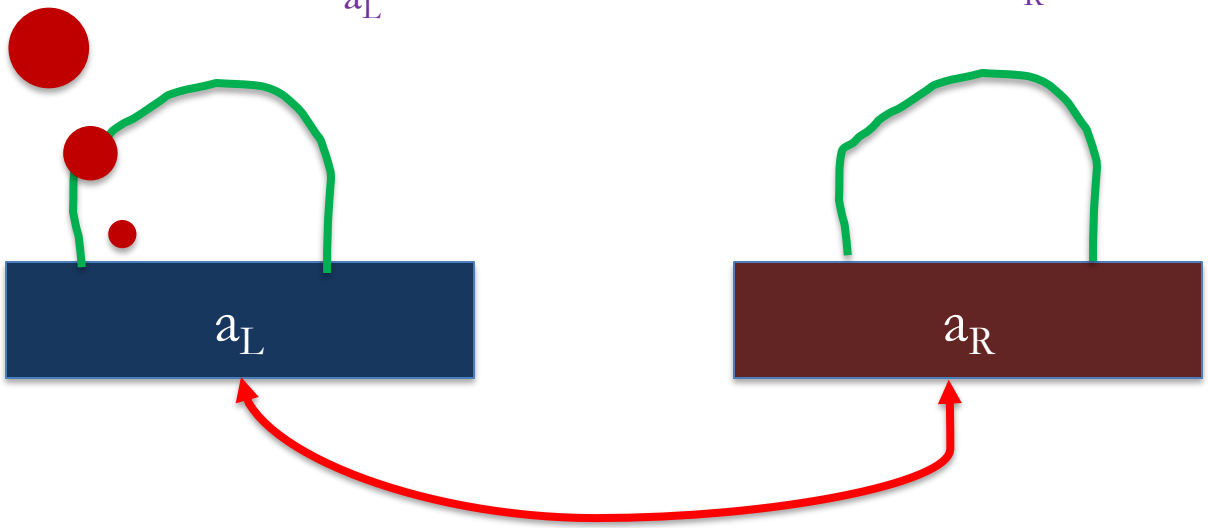
Solving the bad case

Try to modify the MERGE algorithm

First element of a_L is larger than first element of a_R

First element of a_L is smaller than first element of 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

MergeSortCount algorithm

Input: $a_1, a_2, \ldots, a_n$

Output: Numbers in sorted order+ #inversion

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))

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

(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)

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

T(2) = c

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

O(n log n) time

O(n)

Counts #crossing-inversions+ MERGE

MERGE-COUNT(a_L,a_R)

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

```
c = 0
i,j = 1
while i ≤ n' and j ≤ m
        if l_i ≤ r_j
            add l_i to output
            i ++
        else
            add r_j to output
            j ++
            c += n'- i +1
Output any remaining items
return c
```