Nov 8

Collaborative Filtering (Netflix)

Each user ≡ a ranking on movies / shows

Hypothesis: User 1 is "close" to user 2 if user 1's ranking is "close" to user 2's ranking.

Assumption: Each user ranks all movies / shows (unrealis

Input: A ranking $a = a_1, \ldots, a_n$ (permutation of $\{1, \ldots,$

$1, 2, \ldots, n$ ← true ranking

Output: Number of inversions

Recall: (i,j) is an inversion if (1) $i < j$ AND (2) $a_i > a_j$.

Ex 1: $a = (1, 2, \ldots, n)$; #inversions $= 0$ ← smallest possible value. ← none of the pairs (i,j) $i<j$ is an inversio

Ex 2: $a = (n, n-1, \ldots, 1)$; Every pair is an inversion ⇒ #inversions $= \binom{n}{2}$ ← max possible value.

Group Q: Brute force algo? $O(n^2)$

→ check all $\binom{n}{2}$ pairs (i,j) s.t. $i<j$ (& if $a_i > a_j$ then increase count) ⇒ $O(n^2)$

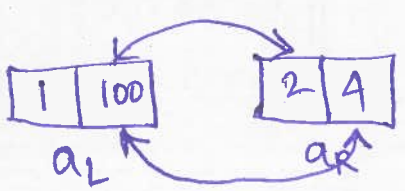
OBS: If you had to output a list of ALL inversions, then brute force algo is optimal: since Ex 2 has output size $\Omega(n^2)$

GOAL: Count #inversions faster than $O(n^2)$

⇒ CANNOT compute all inversions.

Divide and Conquer Algo:

Issue:

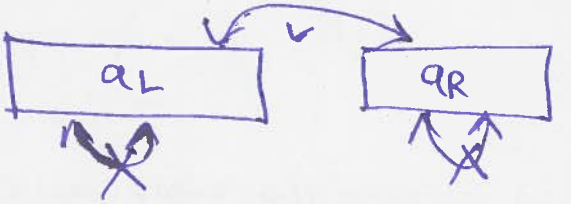


Algo will NOT count the crossing inversions

```
CountInv (a, n)
If n = 1   return 0
If n = 2   return a1 > a2
aL = a1, ..., a⌈n/2⌉
aR = a⌈n/2⌉+1, ..., an-1
return CountInv(aL, ⌈n/2⌉) + CountInv(aR, n-⌈n/2⌉)
```

Problem case: What if all the inversions were crossing inversions?

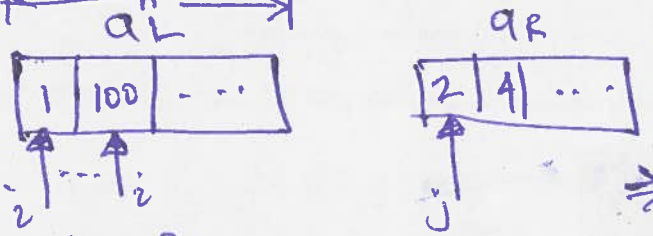
Consider the "bad" case: where all inversions are crossing inversion



⇒ Both a_L & a_R are sorted

Claim: The above case can be solved in $O(n)$ time.

Ex.



Case 1: location i in a_L < location j in a_R

⇒ (i,j) is not an inversion

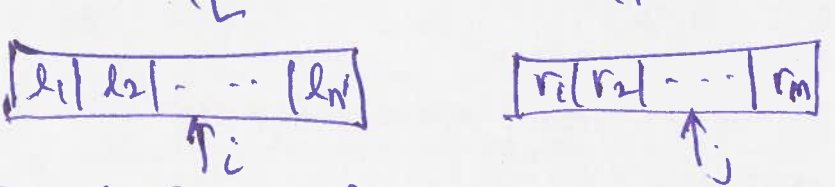
⇒ (i,j') $\forall j' \geq j$ is not an inversion ⇒ advance i++

Case 2: location i in a_L > location j in a_R

⇒ (i,j) is an inversion ⇒ (i',j) $\forall i' \geq i$ are all inversions.

⇒ $(n'-i+1)$ inversions + advance j.

In general:



Goal: Count all pairs (i,j) s.t $l_i > r_j$

MergeCount (a_L, a_R)

0. $c = 0$
1. $i, j = 1$
2. While $i \leq n'$ and $j \leq m$
   - If $l_i \leq r_j$: Append l_i to output; i++
   - else: c+= $n'-i+1$; Append j to output; j++
3. return c

2.5 Append whatever remains to the output

Modify this algo to also sort the merged a_L & a_R