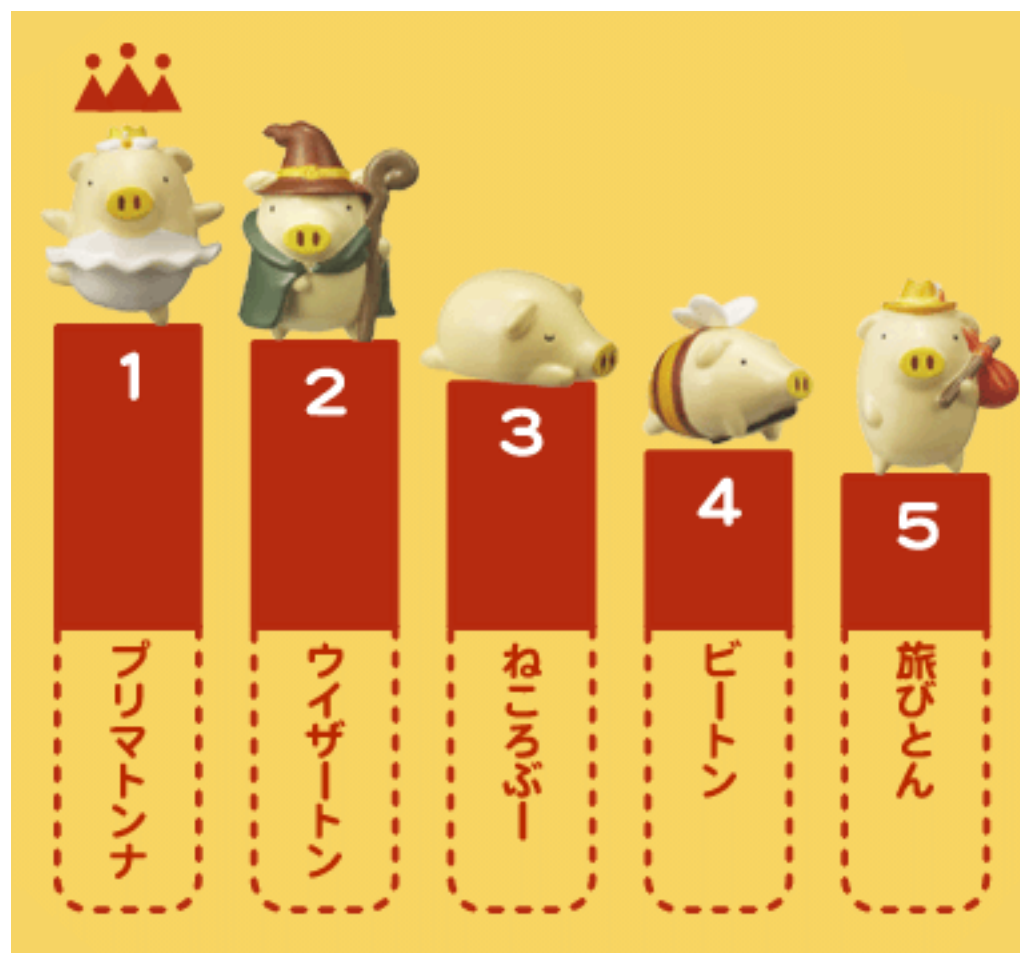


# Lecture 25

CSE 331

# Rankings



# How close are two rankings?



compare two rankings



compare two rankings



[All](#) [News](#) [Shopping](#) [Images](#) [Videos](#) [More](#)

[All](#) [Images](#) [Videos](#) [News](#) [Maps](#) [Shopping](#) [Settings](#)

About 125,000,000 results (0.65 seconds)

All regions [Safe search: moderate](#) [Any time](#)

[https://towardsdatascience.com/rbo-v-s-kendall-tau-to...](https://towardsdatascience.com/rbo-v-s-kendall-tau-to-compare-ranked-lists-of-items/)

<https://stackoverflow.com/questions/13574406/how-to-compare-ranked-lists>

## RBO v/s Kendall Tau to compare ranked lists of items

## How to compare ranked lists - Stack Overflow

Jan 10, 2021 — The Kendall Tau metric also known as Kendall's Correlation method used to check if **two ranked** lists are in agreement.

I have **two** lists of ranked items. Each item has an rank and an associated score. The score has decided the rank. The **two** lists can contains (and usually do) different items, that is their intersection can be empty. I need measures to **compare** such **rankings**. Are there well-known algorithms (in literature or real-world systems) to do so ?

[https://stackoverflow.com/questions/how-to-compar...](https://stackoverflow.com/questions/how-to-compare-ranked-lists)

[https://stackoverflow.com/questions/9149345/ranking-algorithms-to-compare-rankin...](https://stackoverflow.com/questions/9149345/ranking-algorithms-to-compare-rankings)

## How to compare ranked lists - Stack Overflow

## Ranking algorithms to compare "Rankings" - Stack Overflow

Nov 26, 2012 — Cavnar & Trenkle have a nice and simple measure of the **two ranked** lists. The Wilcoxon ranked-sum test gives a measure of ...

Ranking algorithms to compare "Rankings" Ask Question Asked 10 years ago. ... Is there an algorithm that allows to rank items based on the difference of the position of those items in **two** rankings but also "weighted" with the position of a one Player that goes from position 2 to 1?

[3 answers](#) · Top answer: This question has never been answered before, but

# (A Very Simple) Collaborative Filtering Example

Each user: a ranking of movies/shows on Netflix.

1. Movie-X 2. Movie-Y 3. Movie-Z

Assumption: Each user ranks all movies/shows on Netflix.

User 1	User 2	User 3
1	3	1
2	2	3
3	1	2

Hypothesis: A user is close to another user if their rankings are close.

Rankings

---

## Problem Formulation

**Input:** A ranking  $a_1, \dots, a_i, a_j, \dots, a_n$ . ( i.e., a permutation of 1, 2, ..., n)

*Implicit assumption:* 1, 2, ..., n is the “true” ranking (i.e., you compare other rankings to this ranking).

**Output:** The number of inversions.

**Inversion:** (i, j) is an inversion if

1.  $i < j$  AND 2.  $a_i > a_j$

# (A Very Simple) Collaborative Filtering Example

Each user: a ranking of movies/shows on Netflix.

Assumption: Each user ranks all movies/shows on Netflix.

Hypothesis: A user is close to another user if their rankings are close.

1. Movie-X 2. Movie-Y 3. Movie-Z

User 1	User 2	User 3
1	3	1
2	2	3
3	1	2

Rankings

---

## Example 1:

User 2: how many inversions?

Answer: every pair is an inversion.

# (A Very Simple) Collaborative Filtering Example

Each user: a ranking of movies/shows on Netflix.

Assumption: Each user ranks all movies/shows on Netflix.

Hypothesis: A user is close to another user if their rankings are close.

1. Movie-X 2. Movie-Y 3. Movie-Z

User 1	User 2	User 3
1	3	1
2	2	3
3	1	2

Rankings

---

## Example 1:

User 2: how many inversions?

Answer: every pair is an inversion.

# (A Very Simple) Collaborative Filtering Example

Each user: a ranking of movies/shows on Netflix.

Assumption: Each user ranks all movies/shows on Netflix.

Hypothesis: A user is close to another user if their rankings are close.

1. Movie-X 2. Movie-Y 3. Movie-Z

User 1	User 2	User 3
1	3	1
2	2	3
3	1	2

Rankings

---

## Example 1:

User 2: how many inversions?

Answer: every pair is an inversion.

Number of inversions =  $\binom{3}{2} = 3$ , inversions =  $\{(1, 2), (1, 3), (2, 3)\}$ .

# (A Very Simple) Collaborative Filtering Example

Each user: a ranking of movies/shows on Netflix.

Assumption: Each user ranks all movies/shows on Netflix.

Hypothesis: A user is close to another user if their rankings are close.

1. Movie-X 2. Movie-Y 3. Movie-Z

User 1	User 2	User 3
1	3	1
2	2	3
3	1	2

Rankings

---

## Example 1:

User 2: how many inversions?

Answer: every pair is an inversion.

Number of inversions =  $\binom{3}{2} = 3$ , inversions =  $\{(1, 2), (1, 3), (2, 3)\}$ .

User 1: How many inversions?



# (A Very Simple) Collaborative Filtering Example

Each user: a ranking of movies/shows on Netflix.

Assumption: Each user ranks all movies/shows on Netflix.

Hypothesis: A user is close to another user if their rankings are close.

1. Movie-X 2. Movie-Y 3. Movie-Z

User 1	User 2	User 3
1	3	1
2	2	3
3	1	2

Rankings

---

## Example 1:

User 2: how many inversions?

Answer: every pair is an inversion.

Number of inversions =  $\binom{3}{2} = 3$ , inversions =  $\{(1, 2), (1, 3), (2, 3)\}$ .

User 1: How many inversions?

Answer: one inversion: (2, 3).

# (A Very Simple) Collaborative Filtering Example

Each user: a ranking of movies/shows on Netflix.

Assumption: Each user ranks all movies/shows on Netflix.

Hypothesis: A user is close to another user if their rankings are close.

1. Movie-X 2. Movie-Y 3. Movie-Z

User 1	User 2	User 3
1	3	1
2	2	3
3	1	2

Rankings

---

## Example 2:

$A = (1, 2, \dots, n)$ .

How many inversions? 0

If  $a_1, \dots, a_i, a_j, \dots, a_n$  are sorted, then no inversions.

## Example 3:

$A = (n, \dots, 1)$ .

How many inversions?  $\binom{n}{2}$


$$0 \leq \# \text{ inversions} \leq \binom{n}{2}$$

# Solve a harder problem

Input:  $a_1, \dots, a_n$

Output: LIST of all inversions

```
L =  $\phi$ 
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-- $(2i-1, 2i)$



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  $a_1 > a_2$

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

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

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  $a_1 > a_2$ 
```

```
 $a_L = a_1, \dots, a_{\lfloor n/2 \rfloor}$ 
```

```
 $a_R = a_{\lfloor n/2 \rfloor + 1}, \dots, a_n$ 
```

```
return CountInv( $a_L, \lfloor n/2 \rfloor$ ) + CountInv( $a_R, n - \lfloor n/2 \rfloor$ )
```

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  $a_1 > a_2$

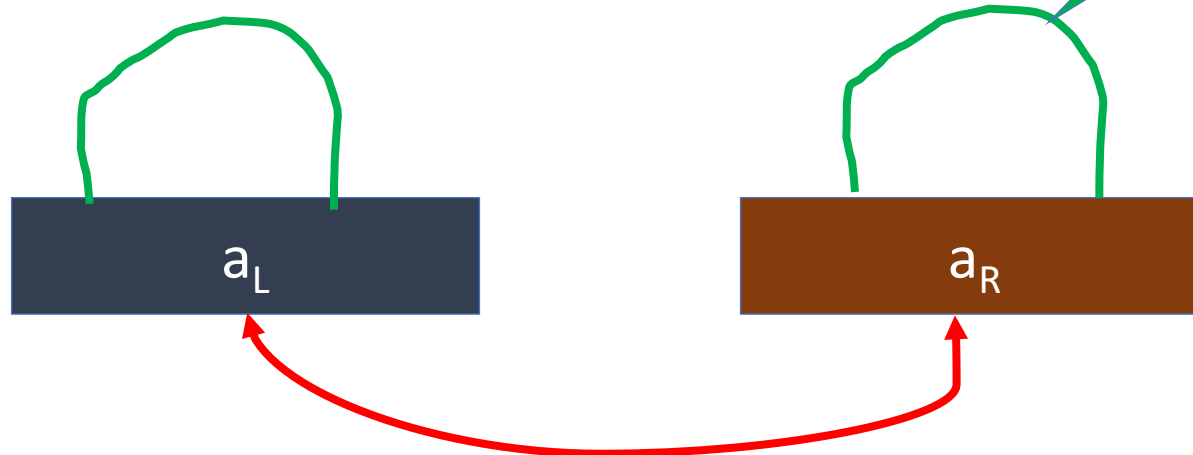
$a_L = a_1, \dots, a_{\lfloor n/2 \rfloor}$

$a_R = a_{\lfloor n/2 \rfloor + 1}, \dots, a_n$

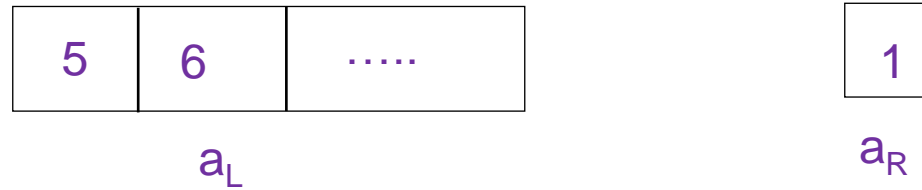
return CountInv( $a_L$ ,  $\lfloor n/2 \rfloor$ ) + CountInv( $a_R$ ,  $n - \lfloor n/2 \rfloor$ )

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

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

Try to  
modify  
the  
MERGE  
algorithm

First element of  $a_L$  is larger than first element of  $a_R$



$a_L$



$a_R$

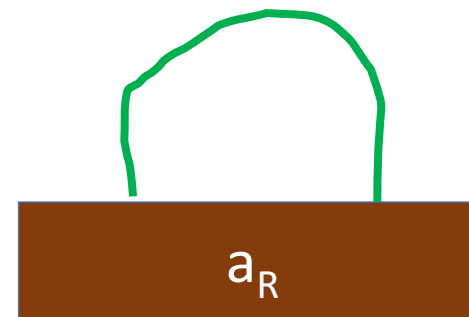
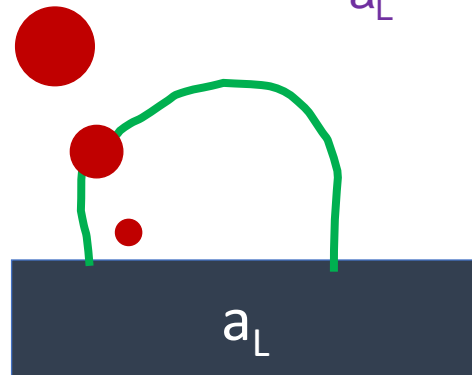
First element of  $a_L$  is smaller than first element of  $a_R$



$a_L$



$a_R$



# 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

# MergeSortCount algorithm

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

Output: Numbers in sorted order+ #inversion

```
MergeSortCount( a, n )  
  If  $n = 1$  return ( 0 ,  $a_1$ )  
  If  $n = 2$  return (  $a_1 > a_2$ ,  $\min(a_1, a_2)$ ;  $\max(a_1, a_2)$ )  
  
   $a_L = a_1, \dots, a_{n/2}$     $a_R = a_{n/2+1}, \dots, a_n$   
  
   $(c_L, a_L) = \text{MergeSortCount}(a_L, n/2)$   
  
   $(c_R, a_R) = \text{MergeSortCount}(a_R, n/2)$   
  
   $(c, a) = \text{MERGE-COUNT}(a_L, a_R)$   
  
  return (  $c + c_L + c_R, a$  )
```

Counts #crossing-inversions+  
MERGE