Design Paradigm: Divide and Conquer

- Finding Rank - Merge Sort
- Karatsuba Algorithm for Integers Multiplication
- Counting Inversions
- Finding Closest Pair in Plane

Imdad ullah Khan

Inversions

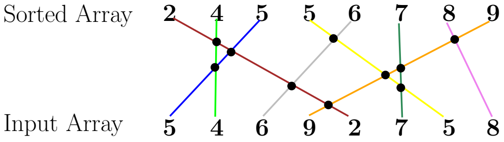
Inversions in an array A of numbers are the out-of-order pairs

Pairs of indices (i, j) such that $i < j$ and $A[i] > A[j]$

$A =$

1	2	3	4	5	6	7	8
5	4	6	9	2	7	5	8

Inversions: $\{(1,2),(1,5),(2,5),(3,5),(3,7),(4,5),(4,6),(4,7),(4,8),(6,7)\}$



Crossing Points (black dots) represent inversions

Inversions

Inversions in an array A of numbers are the out-of-order pairs

Pairs of indices (i, j) such that $i < j$ and $A[i] > A[j]$

Number of inversions is a measure of (dis)sorted-ness of array

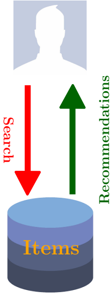
$\triangleright$ An array is sorted if there are zero inversions

Recall which sorting algorithm is better when A has few inversions?

Applications

- Collaborative filtering
- Rank voting theory

Recommendation Systems



Products, news, friends, websites,movies, courses

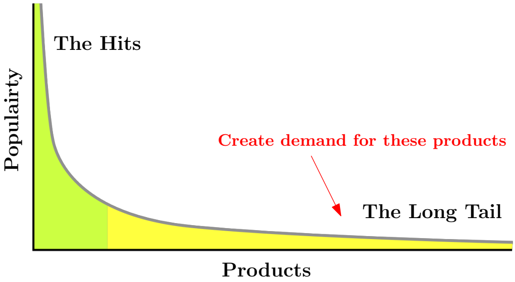
The Web, they say, is leaving the era of search and entering one of discovery. What's the difference? Search is what you do when you're looking for something. Discovery is when something wonderful that you didn't know existed, or didn't know how to ask for, finds you.

J. O'Brien, Nov 20, 2006 The race to create a 'smart' Google

Recommendation Systems

Retailers cannot shelve everything

- ▷ Online retailers and digital content providers have millions of products



Near zero-cost dissemination of information about products

Necessitates information filtering (customization and recommendation)

Recommendation Systems

Filtering can be

- Hand-Curated: ▷ Chef's specials, editor's picks, favorites
- Simple aggregates: ▷ Top 10, Trending, Recent uploads
- Customized to individual users: ▷ Recommendation Systems

Recommendation Systems: Problem Formulation

- n users - $\{u_1, \ldots, u_n\}$ and m items - $\{p_1, \ldots, p_m\}$
- Utility Matrix U: $n \times m$ matrix row/column for each user/item
- $U(i,j)$: rating of user i for item j

	p_1	p_2	p_3						p_j									p_m
u_1	1		2	1		4			2		3	2		5				2
u_2		1				2		1			2			1				3
u_3		1	1	2				1							1		2	
				3		2			5			2			3	4		
		1			2									5				
u_i			3	2	1		4	5	$\boxed{?}$	1		3	1		2			1
			4													4		
			5			1										5		
		1		4							1	3		5		1	2	
u_n			3		1	1		2	1					4				5

$U(i,j)$ could be

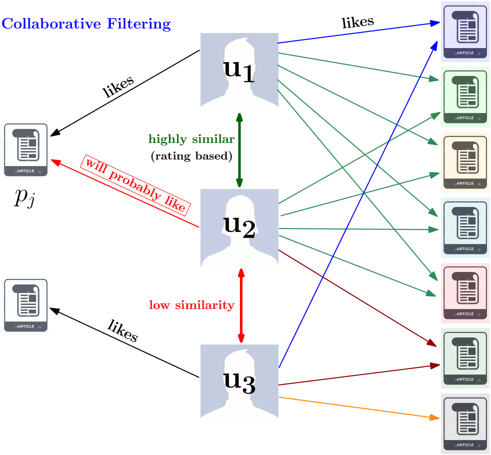
- $0-5$ stars
- $\in [0,1]$
- $\in \{0,1\}$

If prediction for $U(i,j)$ is high, then recommend product j to user i

User-User Collaborative Filtering

Collaboratively filter (personalize) ratings using only the rating matrix U

User i will 'like' item j, if other users **similar** to i rate j higher



Inversions: Application in Collaborative Filtering

Pairs of indices (i, j) such that $i < j$ and $A[i] > A[j]$

Inversions is a measure of distance/similarity between two users

- Sort row of u_x by ratings of u_y
- Inversions in u_x row is distance between u_x and u_y

u_1 :

p_1	p_2	p_3	p_4	p_5	p_6	p_7
2	8	6	1	9	7	3

u_2 :

p_1	p_2	p_3	p_4	p_5	p_6	p_7
7	4	1	9	2	1	8

u_2' :

p_4	p_1	p_7	p_3	p_6	p_2	p_5
9	7	8	1	1	4	2

u_3 :

p_1	p_2	p_3	p_4	p_5	p_6	p_7
2	7	6	1	9	8	2

u_3' :

p_4	p_1	p_7	p_3	p_6	p_2	p_5
1	2	2	6	8	7	9

Is u_2 closer to u_1 or u_3?

Does u_2' have more inversions or u_3'?

Counting Inversions: Algorithm

Pairs of indices (i, j) such that $i < j$ and $A[i] > A[j]$

Input: An array A of n numbers
Output: Number of inversions in A

Algorithm Counting Inversions - Brute force algorithm

$count \leftarrow 0$
for $i = 1$ to n **do**
  for $j = i + 1$ to n **do**
    if $A[i] > A[j]$ **then**
      $count \leftarrow count + 1$

- Correct by definition
- $\binom{n}{2}$ index pairs, number of comparisons is $O(n^2)$

Can we do better?

Counting Inversions: Divide & Conquer

2	3	8	5	4	10	9	12	7	18	15	25

Divide the list into two halves

2	3	8	5	4	10

9	12	7	18	15	25

Recursively count inversions in each half

2	3	8	5	4	10

$Left{-}Left$ =8−5, 8−4, 5−4

9	12	7	18	15	25

$Right{-}Right$: 9−7, 12−7, 18−15

Count inversions where a_i and a_j are in different halves

Left-Right Inversions

2	3	8	5	4	10

9	12	7	18	15	25

Return total inversions count

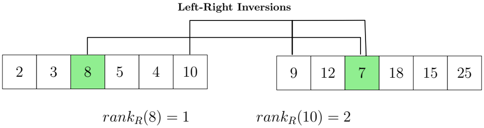
Counting Inversions: Divide & Conquer

- Divide the array into left and right halves
- Find left-left and right-right inversions recursively
- How to find left-right inversions?

How many L-R inversions a given element $x \in L$ is involved in?

Exactly the number of elements in R smaller than x, $\quad \mathbf{rank_R(x)}$

Finding L-R inversions is equivalent to finding ranks of all elements of L in R



L and R sorted $\implies$ can find $rank_R(x)\ \forall\ x \in L$ (L-R inversions) in n steps

sorting L and R removes LL and RR inversions, LR inversions remain intact

Solution: First count LL and RR inversions, then sort L and R

Counting Inversions: Divide & Conquer

Algorithm Counting Inversions

function COUNTINVERSIONS(A) $\triangleright$ returns both sorted A and number of inversion in A

  if $|A| = 1$ **then**

    return $(A, 0)$

  $L \leftarrow A[1, \ldots, n/2]$

  $R \leftarrow A[n/2 + 1, \ldots, n]$

  $(sortedL, LL_{inv}) \leftarrow$ COUNTINVERSIONS(L)

  $(sortedR, RR_{inv}) \leftarrow$ COUNTINVERSIONS(R)

  $LR_{inv} \leftarrow$ SUM(FINDRANKS($sortedL, sortedR$)) $\triangleright$ n steps

  return (MERGE($sortedL, sortedR$), $LL_{inv} + RR_{inv} + LR_{inv}$) $\triangleright$ n steps

Counting Inversions: Recurrence Relation

The recurrence for runtime $T(n)$ on input size n is:

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + 2n & \text{if } n \geq 2 \\ 1 & \text{else} \end{cases}$$

$$T(n) = 2n\log(n)$$

much better than $O(n^2)$