Design and Analysis of Algorithms I

Divide and Conquer

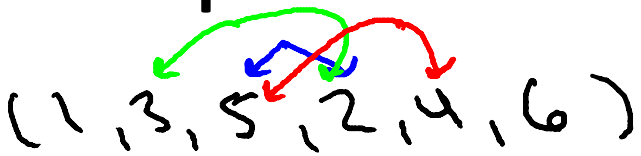
Counting Inversions I

The Problem

Input : array A containing the numbers 1,2,3,..,n in some arbitrary order

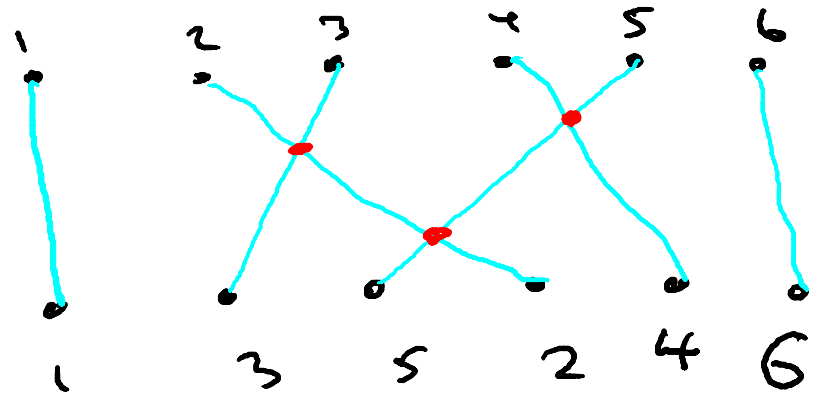
Output : number of inversions = number of pairs (i,j) of array indices with i<j and A[i] > A[j]

Examples and Motivation

Example 

Inversions :

(3,2), (5,2), (5,4)



Motivation : numerical similarity measure between two ranked lists eg: for collaborative filtering

Ter
ver

What is the largest-possible number of inversions that a 6-element array can have?

- 15
- 21
- 36
- 64

In general, $\binom{n}{2} = n(n-1)/2$

High-Level Approach

Brute-force : $\theta(n^2)$ time

Can we do better ? Yes!

KEY IDEA # 1 : Divide + Conquer

Call an inversion (i,j) [with i<j]

Left : if i,j < n/2

Right : if i,j > n/2

Split : if i<=n/2 < j

Note : can compute these recursively

need separate subroutine for these

High-Level Algorithm

Count (array A, length n)

if n=1, return 0

else

X = Count (1st half of A, n/2)

Y = Count (2nd half of A, n/2_

Z = CountSplitInv(A,n) ← CURRENTLY UNIMPLEMENTED

return x+y+z

Goal : implement CountSplitInv in linear (O(n)) time then count will run in O(nlog(n)) time [just like Merge Sort]

Tim Roughgarden