Design and Analysis of Algorithms I

Divide and Conquer

Counting Inversions II

Piggybacking on Merge Sort

KEY IDEA #2: have recursive calls both count inversions and sort.

[i.e., piggyback on Merge Sort]

Motivation: Merge subroutine naturally uncovers split inversions [as we'll see].

High-Level Algorithm (revised)

Sort-and-Count(array A, length n)
  if n=1 return 0
  else
    (0, x) = Sort-and-Count(1st half of A, n/2)
    (0, y) = Sort-and-Count(2nd half of A, n/2)
    (0, z) = Merge-and-CountSplitInv(A, n)
    return x+y+z

Sorted version of 1st half

Sorted version of 2nd half

Sorted version of A

currently unimplemented

Goal: implement Merge-and-CountSplitInv in linear (O(n)) time => then Sort-and-Count will run in O(n log n) time [just like merge sort].

Pseudocode for Merge:

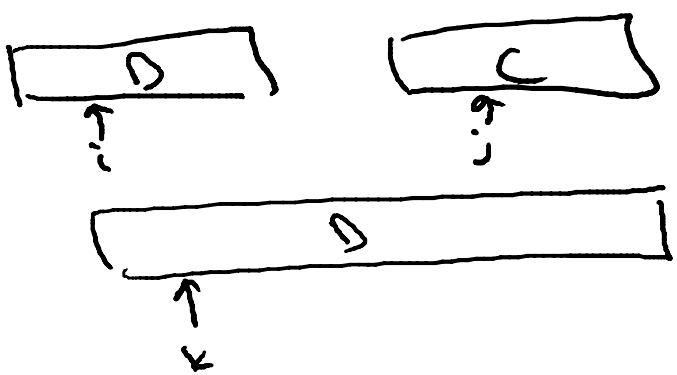
D = output [length = n]

B = 1st sorted array [n/2]

C = 2nd sorted array [n/2]

i = 1

j = 1




```
for k = 1 to n
        if B(i) < C(j)
                D(k) = B(i)
                i++
        else [C(j) < B(i)]
                D(k) = C(j)
                j++
end
```

(ignores end cases)

Tim Roughgarden

Suppose the input array A has no split inversions. What is the relationship between the sorted subarrays B and C?

- B has the smallest element of A, C the second-smallest, B, the third-smallest, and so on.
- All elements of B are less than all elements of C.
- All elements of B are greater than all elements of C.
- There is not enough information to answer this question.

Example

Consider merging $\boxed{1 \mid 3 \mid 5}$ and $\boxed{2 \mid 4 \mid 6}$.

Output: $\boxed{1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6}$

=> when 2 copied to output, discover the split inversions (3,2) and (5,2)

=> when 4 copied to output, discover (5,4)

General Claim

<u>Claim</u>: the split inversions involving an element y of the 2nd array C are precisely the numbers left in the 1st array B when y is copied to the output D.

<u>Proof</u>: let x be an element of the 1st array B.

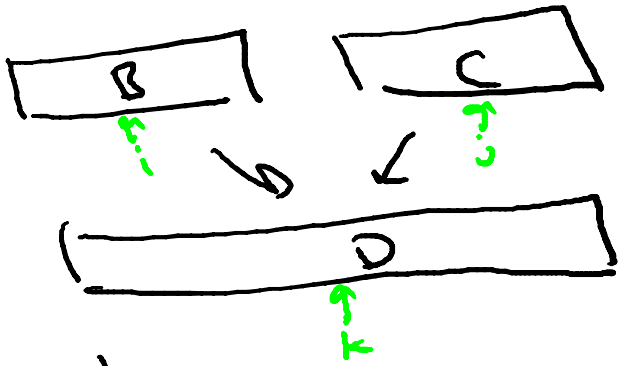
(1) if x copied to output D before y, then $x < y$
$\Rightarrow$ no inversion involving x & y

(2) if y copied to output D before x, then $y < x$
$\Rightarrow$ x & y are a (split) inversion

QED!

Merge_and_CountSplitInv

- while merging the two sorted subarrays, keep running total of number of split inversions
- when element of 2nd array C gets copied to output D, increment total by number of elements remaining in 1st array B



<u>Run time of subroutine</u>: $\underbrace{O(n)}_{\text{Merge}} + \underbrace{O(n)}_{\text{running total}} = O(n)$

$\Rightarrow$ Sort_and_Count runs in $O(n \log n)$ time (just like Merge Sort)