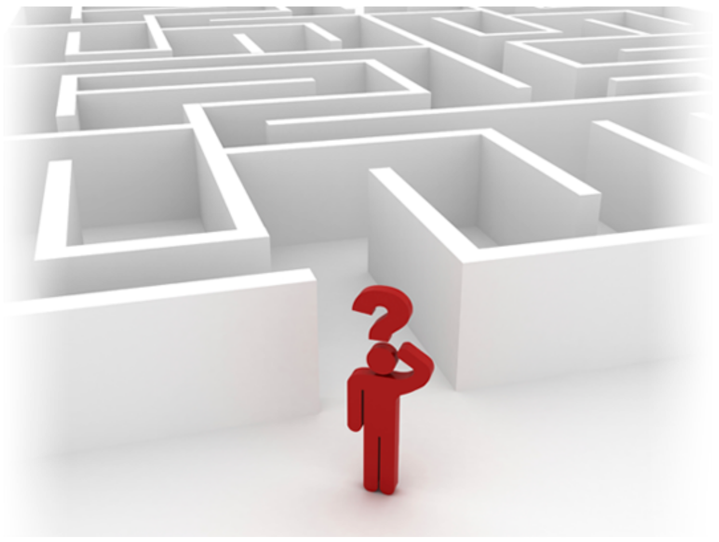


Design and Analysis of Algorithms I

Divide and Conquer

Counting Inversions II

Piggybacking on Merge Sort

KEY IDEA # 2 : have recursive calls both count inversions and sort.
[i.e. , piggy back 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
    (B,X) = Sort-and-Count(1st half of A, n/2)
    (C,Y) = Sort-and-Count(2nd half of A, n/2)
    (D,Z) = 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 CountSplitInv in linear (O(n)) time

=> then Count will run in O(nlog(n)) time [just like Merge Sort]

Pseudocode for Merge:

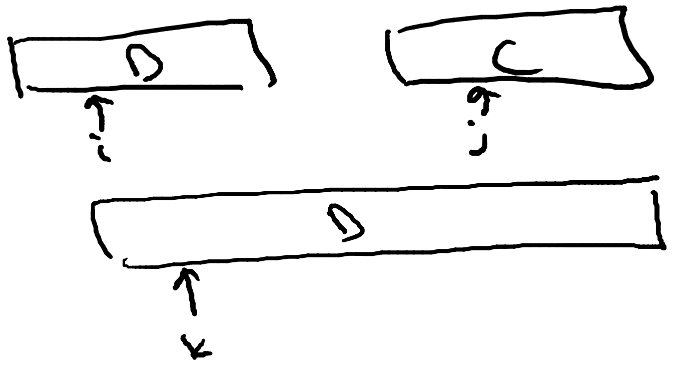
D = output [length = n]

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

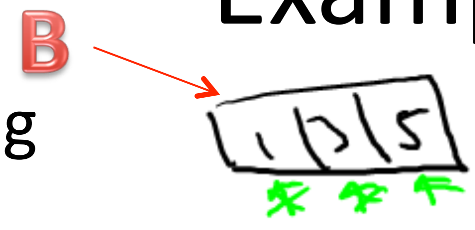
Ter
ver

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

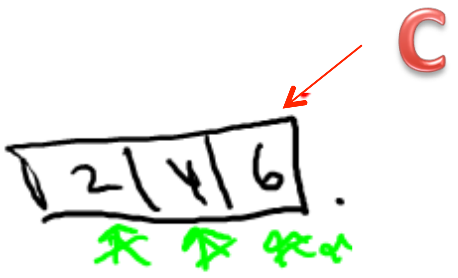
- ○ B has the smallest element of A, C the second-smallest, B, the third-smallest, and so on.
- [✓ ○ All elements of B are less than all elements of C.]
- ○ All elements of B are greater than all elements of C.
- ○ There is not enough information to answer this question.

Example

Consider merging

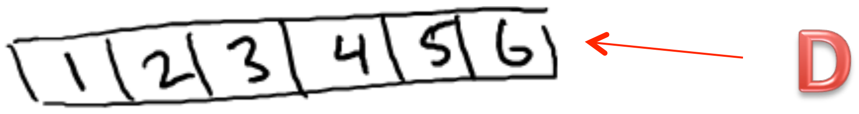


and



.

Output :



⇒When 2 copied to output, discover the split inversions (3,2) and (5,2)

⇒ when 4 copied to output, discover (5,4)

Tim Roughgarden

General Claim

Claim the split inversions involving an element y of the 2nnd array C are precisely the numbers left in the 1st array B when y is copied to the output D.

for example: 如果2 involve，则需要split inversion (3,2) (5,2) remaining part in first array
只要second array比first array element先放进output则有split inversion 发生，否则没有

Proof : Let x be an element of the 1st array B.

1. if x copied to output D before y, then $x < y$

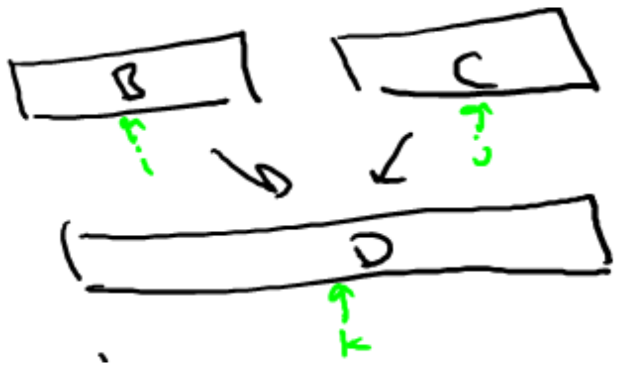
$\Rightarrow$ no inversions involving x and y

2. If y copied to output D before x, then $y < x$

=> X and y are a (split) inversion. Q.E.D

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

Run time of subroutine : $\overset{\text{merge}}{O(n)} + \overset{\text{running total}}{O(n)} = O(n)$

=> Sort_and_Count runs in O(nlog(n)) time [just like Merge Sort]

Tim Roughgarden