Couting Inversions

The Problem of Counting Inversions

- 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 : 1, 3, 5, 2, 4, 6

Inversions :

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

Motivation:

Numerical similarity between two ranked lists. e.g collaborative filtering

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

a) 15
b) 21
c) 36
d) 64

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

a) 15 n choose 2, 6 choose 2 = 15
b) 21
c) 36
d) 64

Brute Force Approach

- Example : 1, 3, 5, 2, 4, 6

Can We Do Better ?

- Divide and Conquer Approach

High Level Algorithm

- Divide inversion pairs in three types

- Left
- Right
- Split

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

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

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

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

Example

- Consider merging B = 1, 3, 5   C = 2, 4, 6

Pseudocode for Merge

```
C = output [length = n]
A = 1st sorted array [n/2]
B = 2nd sorted array [n/2]
i = 1
j = 1
```

```
for k = 1 to n
        if A(i) ≤ B(j)
                C(k) = A(i)
                i++
        else
                C(k) = B(j)
                j++
end
```

```
CountInversions(A, l, r)
{
    If(l == r)  return 0
    m = (l+r)/2
    (B, left) = CountInversions(A, l, m)
    (C, right) = CountInversions(A, m+1, r)
    (A, Split) = CountSplitInvPairs(B, C)
    return (A,(left + right+ split))
}
```

Pseudocode for CountSplitInvPairs

```
C = output [length = n]
A = 1st sorted array [n/2]
B = 2nd sorted array [n/2]
count = 0
i = 1
j = 1
```

```
for k = 1 to n
        if A(i) ≤ B(j)
                C(k) = A(i)
                i++
        else
                C(k) = B(j)
                j++
                count += (n/2)-i+1

return (C, count )
```

Dry Run

- 3, 6, 2, 5, 8, 1

Dry Run

- 3, 6, 2, 5, 8, 1

- Left Inversion Pairs: (3,2), (6,2)
- Right Inversion Pairs: (5,1), (8,1)
- Split Inversion Pairs: (3,1), (6,5), (6,1) (2,1)

- Total 8 inversion pairs

Dry run 3, 6, 2, 5, 8, 1

```
CountInversions(A, l, r)  A = [6, 2]
{
    If(l == r)  return 0
    m = (l+r)/2
    ([2],0)(B, left) = CountInversions(A, l, m)
    ([6],0)(C, right) = CountInversions(A, m+1, r)
    ([2,6],1)(A, Split) = CountSplitInvPairs(B, C)
    return (A,(left + right+ split)) [2,6], 0+0+1 = 1
}
```

Dry run   3, 6, 2, 5, 8, 1

```
CountInversions(A, l, r)  A = [3, 6, 2]
{
    If(l == r)  return 0
    m = (l+r)/2
    ([3],0)(B, left) = CountInversions(A, l, m)
    ([6,2],1)(C, right) = CountInversions(A, m+1, r)
    ([1,3,6],1)(A, Split) = CountSplitInvPairs(B, C)
    return (A,(left + right+ split)) [1,3,6], 0+1+1 = 2
}
```

Dry run 3, 6, 2, 5, 8, 1

```
CountInversions(A, l, r)  A = [5,8,1]
{
    If(l == r)  return 0
    m = (l+r)/2
    ([5],0)(B, left) = CountInversions(A, l, m)
    ([8,1],1)(C, right) = CountInversions(A, m+1, r)
    ([1,5,8],1)(A, Split) = CountSplitInvPairs(B, C)
    return (A,(left + right+ split)) [1,5,8], 0+1+1 = 2
}
```

Dry run 3, 6, 2, 5, 8,1

```
CountInversions(A, l, r)  A = [3,6,2,5,8,1]
{
    If(l == r)  return 0
    m = (l+r)/2
    ([2,3,6],2)(B, left) = CountInversions(A, l, m)
    ([1,5,8],2)(C, right) = CountInversions(A, m+1, r)
    ([1,2,3,5,6,8],4)(A, Split) = CountSplitInvPairs(B, C)
    return (A,(left + right+ split)) [1,2,3,5,6,8], 2+2+4 = 8
}
```