2-4 Inversions

Let $A[1..n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an ***inversion*** of A.

d. Give an algorithm that determines the number of inversions in any permutation on n elements in $\Theta(n \lg n)$ worst-case time. (*Hint:* Modify merge sort.)

Answer :

COUNT-INVERSIONS(A, p, r)

inversions $\leftarrow 0$

if $p < r$

  then $q \leftarrow \lfloor (p + r)/2 \rfloor$

    inversions $\leftarrow$ *inversions* $+$COUNT-INVERSIONS(A, p, q)

    inversions $\leftarrow$ *inversions* $+$COUNT-INVERSIONS$(A, q + 1, r)$

    inversions $\leftarrow$ *inversions* $+$MERGE-INVERSIONS(A, p, q, r)

return *inversions*

MERGE-INVERSIONS(A, p, q, r)

```
n₁ ← q − p + 1
n₂ ← r − q
create arrays L[1 .. n₁ + 1] and R[1 .. n₂ + 1]
for i ← 1 to n₁
    do L[i] ← A[p + i − 1]
for j ← 1 to n₂
    do R[j] ← A[q + j]
L[n₁ + 1] ← ∞
R[n₂ + 1] ← ∞
i ← 1
j ← 1
inversions ← 0
counted ← FALSE
for k ← p to r
    do if counted = FALSE and R[j] < L[i]
          then inversions ← inversions + n₁ − i + 1
               counted ← TRUE
       if L[i] ≤ R[j]
          then A[k] ← L[i]
               i ← i + 1
          else A[k] ← R[j]
               j ← j + 1
               counted ← FALSE
return inversions
```