```
MERGEINVERSION(A, left, mid, right)
  1. leftMaxIndex = mid − left + 1
  2. rightMaxIndex = right − mid
  3. let L[0 ... n_leftMaxIndex] and R[0 ... n_rightMaxIndex] be new arrays
  4. for i = 0 to n_leftMaxIndex
             L[i] = A[left + i – 1]
  5. for j = 0 to n_rightMaxIndex
             R[j] = A[mid + j]
  6. L[n_leftMaxIndex] = INTEGER_MAX_VALUE
  7. L[n_rightMaxIndex] = INTEGER_MAX_VALUE
  8. let i = 0, j = 0
  9. for k = left to right
  10.     if L[i] ≤ R[j]
  11.         A[k] = L[i]
  12.         i = i + 1
  13.     else
  14.         inversionCount = inversionCount + leftMaxIndex – i
  15.         A[k] = R[j]
  16.         j = j + 1
```

기존 MERGE 메소드에서 MERGEINVERSION 메소드로 변경하고 inversion의 개수를 세는 변수 invers_ionCount 변수를 추가하였다. 이때 inversion은 메소드 내에서 오른쪽 array의 값이 왼쪽 array의 값보다 큰 경우에 생긴다.

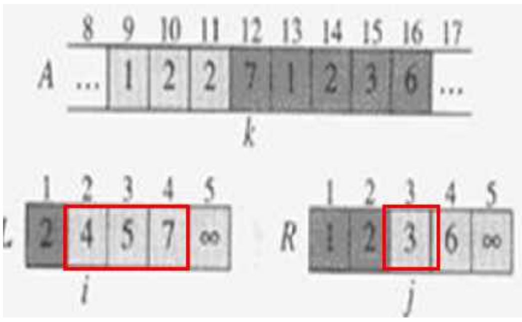

Figure_1

inversion의 개수는 그림 Figure_1 에서처럼 R[j]=3이 L[i]=4보다 작으므로 A[k]에 저장된다. 이때 R[j]는 L[2],L[3],L[4]와 비교를 하므로 inversionCount 가 5 – 2 = 3이다.

시간복잡도

Divide 과정에서 배열의 중간 위치 계산에서 상수 시간이 걸리므로 $D(n) = \Theta(1)$

Conquer 과정에서 2개의 subproblem을 recursive하게 해결하고, 각 subproblem의 크기가 $n/2$이므로 $2T(n/2)$이다.

Combine 과정에서 n개의 숫자를 병합하므로 $C(n) = \Theta(n)$이다. inversionCount를 세는 과정은 단순 계산으로 상수 시간이므로 무시된다.

따라서 $T(n) = \begin{cases} \Theta(1) & (n = 1) \\ 2T(n/2) + \Theta(n) & (n > 1) \end{cases}$ 이다. 이는 기존 MergeSort의 시간복잡도와 같으므로 MERGERINVERSION의 시간복잡도 또한 $\Theta(n \lg n)$이다.