Pseudocodice

C. Ordinamento per fusione (Merge–Sort)

C.1. Fusione (merge) di due array ordinati

```
Algoritmo merge (array B[0..ℓ_B − 1], array C[0..ℓ_C − 1]) → array
  Sia X[0..ℓ_B + ℓ_C − 1] un array
  i_1 ← 0                                  // prossima posizione da considerare in B
  i_2 ← 0                                  // prossima posizione da considerare in C
  k ← 0                                      // prossima posizione da riempire in X
  while i_1 < ℓ_B and i_2 < ℓ_C do    // non raggiunta né la fine di B né quella di C
      if B[i_1] ≤ C[i_2] then
          X[k] ← B[i_1]                        // preleva il prossimo elemento da B
          i_1 ← i_1 + 1
      else
          X[k] ← C[i_2]                        // preleva il prossimo elemento da C
          i_2 ← i_2 + 1
      k ← k + 1
  if i_1 < ℓ_B then
      for j ← i_1 to ℓ_B − 1 do            // copia in X gli elementi rimasti in B
          X[k] ← B[j]
          k ← k + 1
  else
      for j ← i_2 to ℓ_C − 1 do            // copia in X gli elementi rimasti in C
          X[k] ← C[j]
          k ← k + 1
  return X
```

C.2. Schema dell'ordinamento per fusione (utilizza C.1)

```
Algoritmo mergeSort (array A[0..n − 1])
  if n > 1 then
      m ← n/2
      B ← A[0..m − 1]
      C ← A[m..n − 1]
      mergeSort(B)
      mergeSort(C)
      A ← merge(B, C)
```

C.3. Ordinamento per fusione

Al posto degli array ausiliari B e C (che richiedono uso di memoria e di tempo per il trasferimento dei dati) si utilizza l'array A stesso con indici che delimitano le parti da ordinare. Si utilizza un array ausiliario X per le operazioni di merge. X potrebbe essere definito localmente alla procedura *merge*, in quanto viene utilizzato solo da essa. In questo modo tuttavia occorrerebbe un nuovo array ausiliario ad ogni chiamata di *merge*. Per evitare ciò, l'array ausiliario viene definito a livello globale.

```
Algoritmo mergeSort (array A[0..n − 1])
  Sia X[0..n − 1] un array
  mergeSort(A, 0, n, X)

Procedura mergeSort (array A, indice i, indice f, array X)
                          /* Ordina A[i..f − 1] utilizzando X come array ausiliario */
  if f − i > 1 then
      m ← (i + f)/2
      mergeSort(A, i, m, X)
      mergeSort(A, m, f, X)
      merge(A, i, m, f, X)

Procedura merge (array A, indice i, indice m, indice f, array X)
    /* Merge tra A[i..m − 1] e A[m..f − 1] utilizzando X come array ausiliario */

  i1 ← i          // Prima parte: merge di A[i..m − 1] e A[m..f − 1] in X[0..f − i − 1]
  i2 ← m
  k ← 0
  while i1 < m and i2 < f do
      if A[i1] ≤ A[i2] then
          X[k] ← A[i1]
          i1 ← i1 + 1
      else
          X[k] ← A[i2]
          i2 ← i2 + 1
      k ← k + 1
  if i1 < m then
      for j ← i1 to m − 1 do
          X[k] ← A[j]
          k ← k + 1
  else
      for j ← i2 to f − 1 do
          X[k] ← A[j]
          k ← k + 1
  for k ← 0 to f − i − 1 do       // Seconda parte: copia il risultato in A[i..f − 1]
      A[i + k] ← X[k]
```