Pseudocodice

H. Algoritmi di ordinamento non basati su confronti

H.1. Integersort

Algoritmo *integerSort (array* $A[0..n-1]$*, intero* k*)*

```
/* Ordina un array di interi appartenenti all'intervallo [1,k]          */
Sia Y[1..k] un array                                  // array di contatori
for i ← 1 to k do  Y[i] ← 0                           // azzera i contatori

// Conta le occorrenze nell'array A di ciascun intero appartenente a [1,k]
for i ← 0 to n − 1 do
    x ← A[i]
    Y[x] ← Y[x] + 1

// riempi A con i valori ordinati
j ← 0                        // indice della prossima posizione di A da riempire
for i ← 1 to k do
    // assegna l'intero i alle successive Y[i] posizioni di A
    for h ← 1 to Y[i] do
        A[j] ← i
        j ← j + 1
```

H.2. Bucketsort

Algoritmo *bucketSort (array $A[0..n-1]$, intero k)*
```
/* Ordina un array di record in base a un campo chiave intero appartenente
   all'intervallo [1, k]                                                  */
Sia Y[1..k] un array                                          // array di code
for i ← 1 to k do  Y[i] ← coda vuota

// colloca gli elementi di A in differenti code, in base alle chiavi
for i ← 0 to n − 1 do
    x ← A[i].chiave
    Y[x].enqueue(A[i])

// riempi A con i valori ordinati
j ← 0                        // indice della prossima posizione di A da riempire
for i ← 1 to k do
    // colloca i record con chiave i nelle prossime posizioni di A
    while not Y[i].isEmpty() do
        A[j] ← Y[i].dequeue()
        j ← j + 1
```

H.3. Radixsort in base B ($B > 1$ è una costante fissata, e.g., $B = 10$).

Algoritmo *radixSort (array $A[0..n-1]$)*
```
/* Ordina l'array A secondo un campo chiave intero                        */
t ← 0
while esiste una chiave k in A con k/B^t ≠ 0 do
    bucketSort(A, B, t)
    t ← t + 1
```

Procedura *bucketSort (array $A[0..n-1]$, intero b, intero t)*
```
/* Ordina l'array A secondo la cifra di posizione t nella rappresentazione
   in base b della chiave                                                 */
Sia Y[0..b − 1] un array                                      // array di code
for i ← 0 to b − 1 do  Y[i] ← coda vuota

// colloca gli elementi di A in differenti code, in base alle chiavi
for i ← 0 to n − 1 do
    c ← cifra di posizione t nella rappresentazione in base b di A[i].chiave
    Y[c].enqueue(A[i])

// riempi A con i valori ordinati
j ← 0                        // indice della prossima posizione di A da riempire
for i ← 0 to b − 1 do
    // colloca gli elementi che hanno in posizione t la cifra i nelle prossime
       posizioni di A
    while not Y[i].isEmpty() do
        A[j] ← Y[i].dequeue()
        j ← j + 1
```