Counting Sort

Recebe inteiros n e k, e um vetor $A[1 \,.\,.\, n]$ onde cada elemento é um inteiro entre 1 e k.

Counting Sort

Recebe inteiros n e k, e um vetor $A[1\,..\,n]$ onde cada elemento é um inteiro entre 1 e k.

Devolve um vetor $B[1\,..\,n]$ com os elementos de $A[1\,..\,n]$ em ordem crescente.

Counting Sort

Recebe inteiros n e k, e um vetor $A[1\,..\,n]$ onde cada elemento é um inteiro entre 1 e k.

Devolve um vetor $B[1\,..\,n]$ com os elementos de $A[1\,..\,n]$ em ordem crescente.

COUNTINGSORT(A, n)

para $i \leftarrow 1$ **até** k **faça**
    $C[i] \leftarrow 0$
para $j \leftarrow 1$ **até** n **faça**
    $C[A[j]] \leftarrow C[A[j]] + 1$
para $i \leftarrow 2$ **até** k **faça**
    $C[i] \leftarrow C[i] + C[i-1]$
para $j \leftarrow n$ **decrescendo até** 1 **faça**
    $B[C[A[j]]] \leftarrow A[j]$
    $C[A[j]] \leftarrow C[A[j]] - 1$
devolva B

Consumo de tempo

linha	consumo na linha
1	$\Theta(k)$
2	$O(k)$
3	$\Theta(n)$
4	$O(n)$
5	$\Theta(k)$
6	$O(k)$
7	$\Theta(n)$
8	$O(n)$
9	$O(n)$
10	$\Theta(1)$
total	????

Consumo de tempo

linha	consumo na linha
1	$\Theta(k)$
2	$\mathrm{O}(k)$
3	$\Theta(n)$
4	$\mathrm{O}(n)$
5	$\Theta(k)$
6	$\mathrm{O}(k)$
7	$\Theta(n)$
8	$\mathrm{O}(n)$
9	$\mathrm{O}(n)$
10	$\Theta(1)$
total	$\Theta(k+n)$

Counting Sort

COUNTINGSORT(A, n)

para $i \leftarrow 1$ **até** k **faça**
  $C[i] \leftarrow 0$
para $j \leftarrow 1$ **até** n **faça**
  $C[A[j]] \leftarrow C[A[j]] + 1$
para $i \leftarrow 2$ **até** k **faça**
  $C[i] \leftarrow C[i] + C[i-1]$
para $j \leftarrow n$ **decrescendo até** 1 **faça**
  $B[C[A[j]]] \leftarrow A[j]$
  $C[A[j]] \leftarrow C[A[j]] - 1$
devolva B

Consumo de tempo: $\Theta(k + n)$

Se $k = \mathrm{O}(n)$, o consumo de tempo é $\Theta(n)$.

Radix Sort

Algoritmo usado para ordenar

- inteiros não-negativos com d dígitos
- cartões perfurados
- registros cuja chave tem vários campos

Radix Sort

Algoritmo usado para ordenar

- inteiros não-negativos com d dígitos
- cartões perfurados
- registros cuja chave tem vários campos

dígito 1: menos significativo
dígito d: mais significativo

Radix Sort

Algoritmo usado para ordenar

- inteiros não-negativos com d dígitos
- cartões perfurados
- registros cuja chave tem vários campos

dígito 1: menos significativo
dígito d: mais significativo

RADIXSORT(A, n, d)
1 **para** $i \leftarrow 1$ **até** d **faça**
2 ORDENE(A, n, i)

Radix Sort

Algoritmo usado para ordenar

- inteiros não-negativos com d dígitos
- cartões perfurados
- registros cuja chave tem vários campos

dígito 1: menos significativo
dígito d: mais significativo

RADIXSORT(A, n, d)
    para $i \leftarrow 1$ **até** d **faça**
        ORDENE(A, n, i)

ORDENE(A, n, i): ordena $A[1 \,.\,.\, n]$ pelo i-ésimo dígito dos números em A por meio de um algoritmo estável.

Estabilidade

Um algoritmo de ordenação é estável se sempre que, inicialmente, $A[i] = A[j]$ para $i < j$, a cópia $A[i]$ termina em uma posição menor do vetor que a cópia $A[j]$.

Estabilidade

Um algoritmo de ordenação é estável se sempre que, inicialmente, $A[i] = A[j]$ para $i < j$, a cópia $A[i]$ termina em uma posição menor do vetor que a cópia $A[j]$.

Isso só é relevante quando temos informação satélite.

Estabilidade

Um algoritmo de ordenação é estável se sempre que, inicialmente, $A[i] = A[j]$ para $i < j$, a cópia $A[i]$ termina em uma posição menor do vetor que a cópia $A[j]$.

Isso só é relevante quando temos informação satélite.

Quais dos algoritmos que vimos são estáveis?

Estabilidade

Um algoritmo de ordenação é estável se sempre que, inicialmente, $A[i] = A[j]$ para $i < j$, a cópia $A[i]$ termina em uma posição menor do vetor que a cópia $A[j]$.

Isso só é relevante quando temos informação satélite.

Quais dos algoritmos que vimos são estáveis?

- inserção direta? seleção direta? bubblesort?
- mergesort?
- quicksort?
- heapsort?
- countingsort?

Consumo de tempo do Radixsort

Depende do algoritmo ORDENE.

Consumo de tempo do Radixsort

Depende do algoritmo ORDENE.

Se cada dígito é um inteiro de 1 a k,
então podemos usar o COUNTINGSORT.

Consumo de tempo do Radixsort

Depende do algoritmo ORDENE.

Se cada dígito é um inteiro de 1 a k,
então podemos usar o COUNTINGSORT.

Neste caso, o consumo de tempo é $\Theta(d(k+n))$.

Consumo de tempo do Radixsort

Depende do algoritmo ORDENE.

Se cada dígito é um inteiro de 1 a k,
então podemos usar o COUNTINGSORT.

Neste caso, o consumo de tempo é $\Theta(d(k+n))$.

Se d é limitado por uma constante (ou seja, se $d = \mathrm{O}(1)$)
e $k = \mathrm{O}(n)$, então o consumo de tempo é $\Theta(n)$.