Algoritmos de Ordenamiento

Héctor Ferrada
académicos

Instituto de Informática
Universidad Austral de Chile
INFO088 - Estructuras de Datos y Algoritmos

2do Semestre, 2019

Burbuja

- En este algoritmo ordena los n elementos de la lista en n pasos.
- Los elementos mas grandes se van ubicando al final de la lista (o arreglo) quedando la parte final de la lista ordenada.
- Así, dada **la parte inicial de la lista** (desordenada) el elemento mayor de esta sublista ***burbujea*** hasta **su posición final**.
- Este proceso se repite n veces hasta que la lista queda complemanente ordenada.

Burbuja

- En este algoritmo ordena los n elementos de la lista en n pasos.
- Los elementos mas grandes se van ubicando al final de la lista (o arreglo) quedando la parte final de la lista ordenada.
- Así, dada **la parte inicial de la lista** (desordenada) el elemento mayor de esta sublista ***burbujea*** hasta **su posición final**.
- Este proceso se repite n veces hasta que la lista queda complemanente ordenada.

Ejemplo, $lista = [57, 46, 15, 70, 57, 41, 45, 21]$

Paso 1:

$lista = [57,\ 46,\ 15, 70, 57, 41, 45, 21]$

$lista = [46,\ 57,\ 15,\ 70, 57, 41, 45, 21]$

$lista = [46, 15,\ 57,\ 70,\ 57, 41, 45, 21]$

$lista = [46, 15, 57,\ 70,\ 57,\ 41, 45, 21]$

$lista = [46, 15, 57, 57,\ 70,\ 41,\ 45, 21]$

$lista = [46, 15, 57, 57, 41,\ 70,\ 45,\ 21]$

$lista = [46, 15, 57, 57, 41, 45,\ 70,\ 21]$

$lista = [46, 15, 57, 57, 41, 45, 21,\ 70]$

Burbuja

Paso 2:

$$lista = [46, 15, 57, 57, 41, 45, 21, 70]$$
$$lista = [15, 46, 57, 57, 41, 45, 21, 70]$$
$$lista = [15, 46, 57, 57, 41, 45, 21, 70]$$
$$lista = [15, 46, 57, 57, 41, 45, 21, 70]$$
$$lista = [15, 46, 57, 41, 57, 45, 21, 70]$$
$$lista = [15, 46, 57, 41, 45, 57, 21, 70]$$
$$lista = [15, 46, 57, 41, 45, 21, 57, 70]$$

Burbuja

Paso 2:

$$lista = [46, 15, 57, 57, 41, 45, 21, 70]$$
$$lista = [15, 46, 57, 57, 41, 45, 21, 70]$$
$$lista = [15, 46, 57, 57, 41, 45, 21, 70]$$
$$lista = [15, 46, 57, 57, 41, 45, 21, 70]$$
$$lista = [15, 46, 57, 41, 57, 45, 21, 70]$$
$$lista = [15, 46, 57, 41, 45, 57, 21, 70]$$
$$lista = [15, 46, 57, 41, 45, 21, 57, 70]$$

Paso 3:

$$lista = [15, 46, 57, 41, 45, 21, 57, 70]$$
$$lista = [15, 46, 57, 41, 45, 21, 57, 70]$$
$$lista = [15, 46, 57, 41, 45, 21, 57, 70]$$
$$lista = [15, 46, 41, 57, 45, 21, 57, 70]$$
$$lista = [15, 46, 41, 45, 57, 21, 57, 70]$$
$$lista = [15, 46, 41, 45, 21, 57, 57, 70]$$

Burbuja

Paso 4:

$$lista = [15, 46, 41, 45, 21, 57, 57, 70]$$
$$lista = [15, 46, 41\ 45, 21, 57, 57, 70]$$
$$lista = [15, 41\ 46, 45\ 21, 57, 57, 70]$$
$$lista = [15, 41, 45\ 46, 21, 57, 57, 70]$$
$$lista = [15, 41, 45, 21, 46, 57, 57, 70]$$

Burbuja

Paso 4:

$$lista = [15, 46, 41, 45, 21, 57, 57, 70]$$
$$lista = [15, 46, 41\ 45, 21, 57, 57, 70]$$
$$lista = [15, 41\ 46, 45\ 21, 57, 57, 70]$$
$$lista = [15, 41, 45\ 46, 21, 57, 57, 70]$$
$$lista = [15, 41, 45, 21, 46, 57, 57, 70]$$

Paso 5:

$$lista = [15, 41, 45, 21, 46, 57, 57, 70]$$
$$lista = [15, 41, 45, 21, 46, 57, 57, 70]$$
$$lista = [15, 41, 45, 21, 46, 57, 57, 70]$$
$$lista = [15, 41, 21, 45, 46, 57, 57, 70]$$

Burbuja

Paso 4:

$$lista = [15, 46, 41, 45, 21, 57, 57, 70]$$
$$lista = [15, 46, 41\ 45, 21, 57, 57, 70]$$
$$lista = [15, 41\ 46, 45\ 21, 57, 57, 70]$$
$$lista = [15, 41, 45\ 46, 21, 57, 57, 70]$$
$$lista = [15, 41, 45, 21, 46, 57, 57, 70]$$

Paso 5:

$$lista = [15, 41, 45, 21, 46, 57, 57, 70]$$
$$lista = [15, 41, 45, 21, 46, 57, 57, 70]$$
$$lista = [15, 41, 45, 21, 46, 57, 57, 70]$$
$$lista = [15, 41, 21, 45, 46, 57, 57, 70]$$

Paso 6:

$$lista = [15, 41, 21, 45, 46, 57, 57, 70]$$
$$lista = [15, 41, 21\ 45, 46, 57, 57, 70]$$
$$lista = [15, 21, 41, 45, 46, 57, 57, 70]$$

Burbuja

Paso 4:

$$lista = [15, 46, 41, 45, 21, 57, 57, 70]$$
$$lista = [15, 46, 41\ 45, 21, 57, 57, 70]$$
$$lista = [15, 41\ 46, 45\ 21, 57, 57, 70]$$
$$lista = [15, 41, 45\ 46, 21, 57, 57, 70]$$
$$lista = [15, 41, 45, 21, 46, 57, 57, 70]$$

Paso 5:

$$lista = [15, 41, 45, 21, 46, 57, 57, 70]$$
$$lista = [15, 41, 45, 21, 46, 57, 57, 70]$$
$$lista = [15, 41, 45, 21, 46, 57, 57, 70]$$
$$lista = [15, 41, 21, 45, 46, 57, 57, 70]$$

Paso 6:

$$lista = [15, 41, 21, 45, 46, 57, 57, 70]$$
$$lista = [15, 41, 21\ 45, 46, 57, 57, 70]$$
$$lista = [15, 21, 41, 45, 46, 57, 57, 70]$$

Paso 7:

$$lista = [15, 21, 41, 45, 46, 57, 57, 70]$$
$$lista = [15, 21, 41, 45, 46, 57, 57, 70]$$

Burbuja - Pseudocódigo

Input: un arreglo de n elementos $A[1..n]$
Output: A ordenado ascendentemente

bubleSort$(A, n)\{$
  for $i = 1$ to n $\{$
    for $j = 2$ to $n - i + 1$ $\{$
      if $(A[j] < A[j-1])$ $\{$
        $temp = A[j]$
        $A[j] = A[j-1]$
        $A[j-1] = temp$
      $\}$
    $\}$
  $\}$
$\}$

Burbuja - Pseudocódigo

Input: un arreglo de n elementos $A[1..n]$
Output: A ordenado ascendentemente

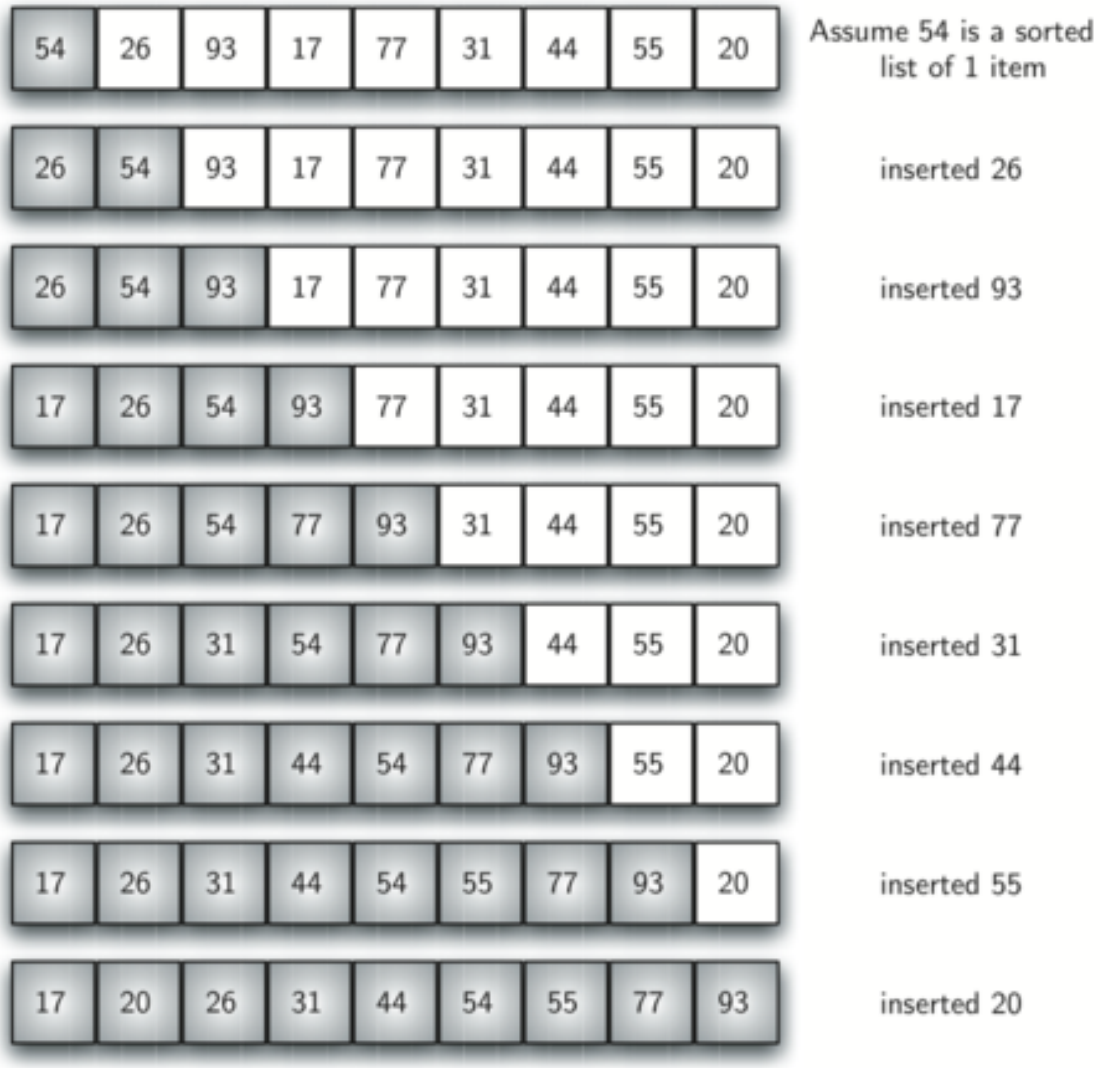
bubleSort(A, n){
  for $i = 1$ to n {
    for $j = 2$ to $n - i + 1$ {
      if ($A[j] < A[j-1]$) {
        $temp = A[j]$
        $A[j] = A[j-1]$
        $A[j-1] = temp$
      }
    }
  }
}

Ejercicio:
Cree un fuente sorting.cpp, que reciba como argumento un entero n y cree un arreglo $A[0..n-1]$ de enteros aleatorios en el rango $[-L, L]$ (utilice: #define L 30). Luego:
Implemente el método **bubleSort(int*** A**, int** n**);**.
Implemente el método **bubleSortDesc(int*** A**, int** n**)**, que implementa la burbuja ordenando descendentemente.
Testeelos y pruebe su correctitud !!

Insertion Sort

- Los n elementos de $A[1..n]$ se ordenan en $n-1$ pasos.
- Se considera al primer elemento, $A[1]$, como un subarreglo ordenado de largo 1, y se inserta $x = A[2]$ en su posición correcta. Queda entonces el subarreglo $A[1..2]$ ordenado, faltando por ordenar $A[3..n]$.
- En el segundo paso, se inseta $x = A[3]$ en su posición correcta, haciendo los *corrimientos* entre elementos consecutivos partiendo desde $A[2]$ y bajando (de derecha a izquierda) hasta la posición correcta de x. Queda entonces el subarreglo $A[1..3]$ ordenado, faltando por ordenar $A[4..n]$.
- En el k-ésimo paso, insertamos $x = A[k]$ en su posición correcta, digamos j, $1 \leq j \leq k$, haciendo los corrimientos entre elementos consecutivos partiendo desde $A[k]$ y bajando hasta j. Queda entonces el subarreglo $A[1..k]$ ordenado, faltando por ordenar $A[k+1..n]$.
- El proceso se repite $n-1$ pasos hasta que la lista queda totalmente ordenada.

Insertion Sort - Ejemplo



Insertion Sort - Pseudocódigo

Input: un arreglo de n elementos $A[1..n]$
Output: A ordenado ascendentemente

insertionSort(A, n){
  for $i = 2$ to n do
    $key = A[i]$
    $j = i - 1$
    while($j > 0$ AND $A[j] > key$) do
      $A[j + 1] = A[j]$
      $j = j - 1$
    $A[j + 1] = key$
}

Insertion Sort - Pseudocódigo

Input: un arreglo de n elementos $A[1..n]$
Output: A ordenado ascendentemente

insertionSort(A, n){
  for $i = 2$ to n do
    $key = A[i]$
    $j = i - 1$
    while($j > 0$ AND $A[j] > key$) do
      $A[j + 1] = A[j]$
      $j = j - 1$
    $A[j + 1] = key$
}

Ejercicios:

- ¿Cuál de los dos algoritmos es más eficiente?. Explique porqué.
- Implemente, en C++, el método **insertionSort(int*** A**, int** n**);**.
- Implemente el método **insertionSortDesc(int*** A**, int** n**)**, que implementa insertion Sort ordenando descendentemente.

Ejercicios Propuestos

1. Re-implemente el algoritmo de la burbuja, haciendo ahora que los números *burbujeen* de derecha a izquierda y no de izquierda a derecha.
2. Cree una función **mixSort(**A, B, n_a, n_b**)**, que reciba dos arreglos ordenadas ascendentemente $A[0..n_A - 1]$ y $B[0..n_B - 1]$, y retorne un arreglo ordenado X, de largo $n_A + n_B$, mezclando los elementos de ambos arreglos.
   Para esto, utilice 3 contadores para recorrer cada arreglo: i, j y k, e inicialicelos con 0, luego use dos variables $u = A[i]$ y $v = B[j]$ y en cada paso guarde el menor entre u y v en $X[c]$, incrementando c en 1. Si el menor es u incremente i en 1 y actualice u con el valor siguiente de A, sino incremente j en 1 actualice v con el valor siguiente de B. Continue iterando de esta forma hasta alamacenar todos los elementos en X.

   Para testear, genere arreglos con números aleatorios y ordenelos con la burbuja o insertionSort, luego llame a *mixSort()* con estos arreglos ya ordenados.