


Algoritmos / Bubble Sort

Recorrer y comparar los elementos de la lista. A pesar de su ineficiencia, Bubble Sort sigue siendo útil para fines educativos o para ordenar pequeñas cantidades de datos donde la simplicidad es más importante que la velocidad.

js


 Copiar código

```
// Algoritmos / Bubble Sort
function bubbleSort(nums){
  for(let i = 0; i < nums.length; i++){
    if(nums[i] > nums[i + 1]){
      let j = nums[i + 1];
      nums[i + 1] = nums[i];
      nums[i] = j;
      bubbleSort(nums);
    }
  }
  return nums;
}
```

```
let arr = [12, 2, 3, 41, 5, 6, 7, 83, 9, 10, 11, 18, 13, 22, 34, 15, 56,35,66,8,87,98];
console.log(arr);
```

```
let res = bubbleSort(arr);
console.log(res);
```

js

 Copiar código

```
// Ejercicio 2
function Burbuja(ndatos){
  let n = ndatos.length;

  for(let i = 0; i < n; i++){
    for(let j = 0, l = n - 1; j < l; j++){
      if(ndatos[j] > ndatos[j+1]){
        [ndatos[j], ndatos[j+1]] = [ndatos[j+1], ndatos[j]];
      }
    }
  }
}
```

```

    }
    return ndatos;
  }

  let arr2 = [12, 2, 3, 41, 5, 6, 7, 83, 9, 10, 11, 18, 13, 22, 34, 15, 56, 35, 66, 8, 87, 98];
  console.log(arr2);


  let result = Burbuja(arr2);
  console.log(result);

```

Algoritmos / Select Sort

Selection Sort es un algoritmo de ordenamiento simple que divide el arreglo en dos partes: la parte ordenada y la parte no ordenada. En cada iteración, busca el elemento más pequeño en la parte no ordenada y lo intercambia con el primer elemento de la parte no ordenada. Esto significa que el elemento más pequeño siempre se coloca al principio de la parte ordenada. Este proceso se repite hasta que todos los elementos estén en su lugar, lo que resulta en un arreglo ordenado. Aunque no es eficiente para conjuntos de datos grandes debido a su complejidad cuadrática, es fácil de entender e implementar para conjuntos de datos pequeños o parcialmente ordenados.

js

 Copiar código

```

// Algoritmo Selection Sort
function SelectionSort (ndatos){
  let aux = [...ndatos];

  for(let i = 0; i < ndatos.length; i++){

    let min = aux.slice( i + 1)

    .reduce((a,e,j) => (e < aux [a] ? j + i +1: a), i);

    if(min !=i){
      [aux[i], aux[min]] = [aux[min], aux[i]];
    }
  }
  return aux;
}


let arr3 = [12, 2, 3, 41, 5, 6, 7, 83, 9, 10, 11, 18, 13, 22, 34, 15, 56, 35, 66, 8, 87, 98];
console.log(arr3);

```

```
let result = SelectionSort(arr3);
console.log(result);
```

11.04.2024 MOD-2

js

 Copiar código

```
// Algoritmo Selection Sort ejemplo 2
function SelectionSort (arr){
  // si solo tiene un elemento, no hay que ordenar nada
  if(arr.length < 2) return arr;
  let min;
  let index;

  for (let i = 0; i < arr.length; i++){
    min = arr[i]

    for (let j = i + 1; j < arr.length; j++){
      // compara el valor menor que tenemos con el próximo en la lista
      if(min > arr[j]){
        min = arr[j];
        index = j;
      }
    }
    // intercambia los valores, si encuentra uno mayor
    if(index){
      [arr[i], arr[index]] = [arr[index], arr[i]]
      index = undefined;
    }
  }
  return arr;
}

let arr3 = [12, 2, 3, 41, 5, 6, 7, 83, 9, 10, 11, 18, 13, 22, 34, 15, 56,35,66,8,87,98];
console.log(arr3);


let result = SelectionSort(arr3);
console.log(result)
```

Algoritmos / Insertion Sort

Insertion Sort es un algoritmo simple de ordenamiento que recorre el arreglo de izquierda a derecha, insertando cada elemento en su posición correcta en la parte ordenada del arreglo. Comienza con un solo elemento (considerado ya ordenado) y luego compara cada elemento

subsiguiente con los elementos en la parte ordenada, moviéndolos hacia la derecha hasta encontrar su posición adecuada. Este proceso continúa hasta que todos los elementos están en su lugar, lo que resulta en un arreglo ordenado. Aunque no es tan eficiente como otros algoritmos para grandes conjuntos de datos, es útil para conjuntos de datos pequeños o parcialmente ordenados.


js

 Copiar código

```
// Algoritmo insertionSort
function insertionSort(arr){
  let valorToInsert = arr[1];
  let j;
  for(let i = 1; i < arr.length; i++){
    valToInsert = arr[i];
    for(j = i - 1; j >= 0 && arr[j] > valToInsert; j-- ){
      arr[j + 1] = arr[j]
    }
    arr[j + 1] = valToInsert;
  }
  return arr;
}

let arreglo = [12, 2, 3, 41, 5, 6, 7, 83, 9, 10, 11, 18, 13, 22, 34, 15, 56,35,66,8,87,98];
let result = insertionSort(arreglo);
console.log(arreglo);
```

js

 Copiar código

```
// Algoritmo insertionSort ejercicio 2
function InSort(val){
  for(let i = 1; i < val.length; i++){
    let j = i - 1
    let aux = val[i]
    while(j >= 0 && val[j] > aux){
      val[j + 1] = val[j];
      --j;
    }
    val[j+1] = aux;
  }
  return val
}

let arreglo = [12, 2, 3, 41, 5, 6, 7, 83, 9, 10, 11, 18, 13, 22, 34, 15, 56,35,66,8,87,98];
let result = InSort(arreglo);
console.log(result);
```

Bryan Hernández | Telento Tech DWFSV2-42 | 2024

