## Bubble Sort

|  |  |
| --- | --- |
| init | 1 4 3 9 5 8 7 |
| 1 | 1 4 3 5 8 7 **9** |
| 2 | 1 4 3 5 7 **8** 9 |
| 3 | 1 3 4 5 **7** 8 9 |
| 4 | 1 3 4 **5** 7 8 9 |
| 5 | 1 3 **4** 5 7 8 9 |
| 6 | 1 **3** 4 5 7 8 9 |

### Algorithm

Each time bubble the largest element to the right side.

For array A[n]:

The first round bubble A[0 .. n-1], result in the max element bubbled to A[n-1]

The second round bubble A[0 .. n-2], result in the max element bubbled to A[n-2]

…

The last (n-1) round bubble A[0 .. 1], result in the max element bubbled to A[1]

### Pseudocode

Bubble Sort A[n]

For i = n-1 to 1 (total rounds: n-1)

For j = 0 to i

If (A[j] < A[j + 1])

Swap(A[j], A[j+1])

## Insertion Sort

|  |  |
| --- | --- |
| init | 1 4 3 9 5 8 7 |
| 1 | 1 **4** 3 9 5 8 7 |
| 2 | 1 **3** 4 9 5 8 7 |
| 3 | 1 3 4 **9** 5 8 7 |
| 4 | 1 3 4 **5** 9 8 7 |
| 5 | 1 3 4 5 **8** 9 7 |
| 6 | 1 3 4 5 **7** 8 9 |

### Algorithm

Each time put A[n] into sorted sequence A[0 .. n-1]

The first round put A[1] into sorted sequence A[0]

The second round put A[2] into sorted sequence A[0..1]

..

The last round put A[n-1] into sorted sequence A[0..n-2]

### Pseudocode

Insertion Sort A[n]

For i = 1 to n – 1

Key = A[i]

While i > 0 && key < A[i - 1]

A[i - 1] = key

i--

A[i] = key

## Selection Sort

|  |  |
| --- | --- |
| init | 1 4 3 9 5 8 7 |
| 1 | **1** 4 3 9 5 8 7 |
| 2 | 1 **3** 4 9 5 8 7 |
| 3 | 1 3 **4** 9 5 8 7 |
| 4 | 1 3 4 **5** 9 8 7 |
| 5 | 1 3 4 5 **7** 9 8 |
| 6 | 1 3 4 5 7 **8** 9 |

### Algorithm

The first round mark index(0) as minimal, find index(i) that have smallest value in index(0..n-1), swap(0,i)

### Pseudocode

Selection Sort A[n]

for i = 0 to n-2

min = i

for j = i+1 to n-1

if (A[j] < A[min])

min = j

swap(A[i], A[min])

## Quick Sort

key =A[0]

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | i | j | action |
| init | 49 | 38 | 65 | 97 | 76 | 13 | 27 | 0 | 6 | - |
| 1 | **27** | 38 | 65 | 97 | 76 | 13 | **27** | 0 | 6 | A0=A6 |
| 2 | 27 | 38 | **65** | 97 | 76 | 13 | **65** | 2 | 6 | A6=A2 |
| 3 | 27 | 38 | **13** | 97 | 76 | **13** | 65 | 2 | 5 | A2=A5 |
| 4 | 27 | 38 | 13 | **97** | 76 | **97** | 65 | 3 | 5 | A5=A3 |
| 5 | 27 | 38 | 13 | 97 | 76 | 97 | 65 | 3 | 4 | - |
| 6 | 27 | 38 | 13 | 97 | 76 | 97 | 65 | 3 | 3 | - |
| 7 | 27 | 38 | 13 | **49** | 76 | 97 | 65 | 3 | 3 | A3=key |

### Algorithm

For array A[n]  
Set A[0] as pivot, put elements less or equal than pivot to the left, elements greater than pivot to the right.

Then recuse the left and the right.

### Pseudocode

Quick Sort A[n]

quickSort(A[n], low, high){

int partitionIndex = partition(A[n], 0, n-1);

quickSort (A[n], 0, partitionIndex-1);

quickSort (A[n], partitionIndex+1, n-1);

}

partition(A[n], low, high)

While (low < high)

For j = high to low if (A[high] <= key) A[low] = A[high] break

For i = low to high if (A[low] > key) A[high] = A[low] break

A[low] = A[high] = key