

Insertion Sort

Insertion Sort

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 9 | 2 | 7 | 0 | 4 | 6 |

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

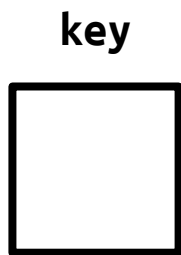
Insertion Sort

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 9 | 2 | 7 | 0 | 4 | 6 |

size = 6

```
void insertionSort(int a[], int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

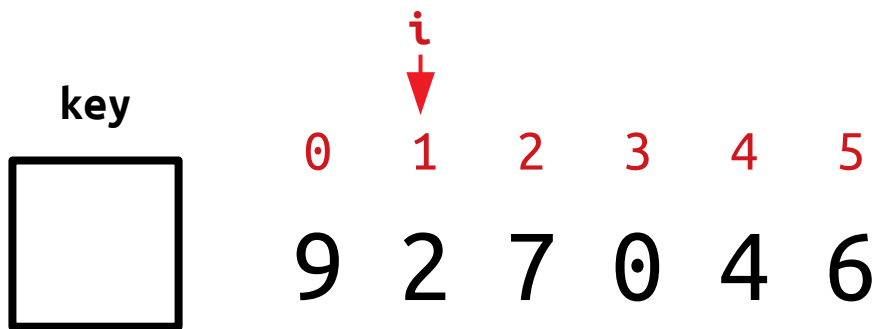


| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 9 | 2 | 7 | 0 | 4 | 6 |

size = 6
i =
j =

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

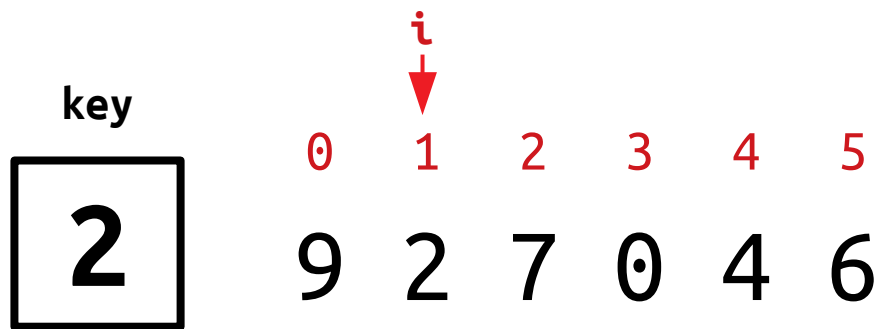
Insertion Sort



size = 6
i = 1
j =

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort



size = 6
i = 1
j =

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

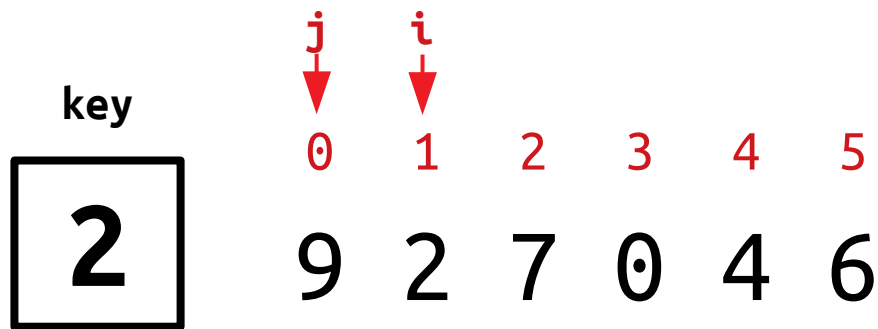
key
2

| | | | | | | |
|--------|--------|---|---|---|---|--|
| j ↓ | i ↓ | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | |
| 9 | 2 | 7 | 0 | 4 | 6 | |

size = 6
i = 1
j = 0

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort



size = 6
i = 1
j = 0

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```


Insertion Sort

key

2

| | | | | | | |
|--------|--------|---|---|---|---|--|
| j ↓ | i ↓ | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | |
| 9 | 2 | 7 | 0 | 4 | 6 | |

size = 6

i = 1

j = 0

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

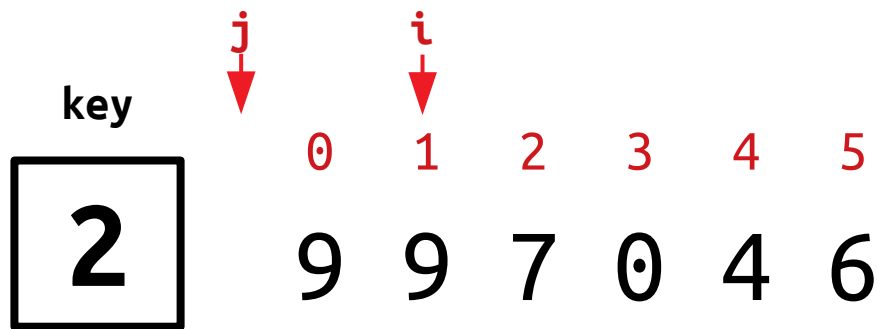
key
2

| j ↓ | i ↓ | | | | | |
|-----|-----|---|---|---|---|--|
| 0 | 1 | 2 | 3 | 4 | 5 | |
| 9 | 9 | 7 | 0 | 4 | 6 | |

size = 6
i = 1
j = 0

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

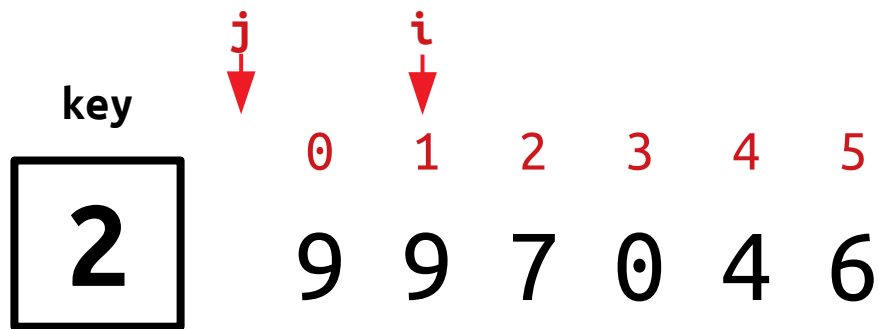
Insertion Sort



size = 6
i = 1
j = -1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

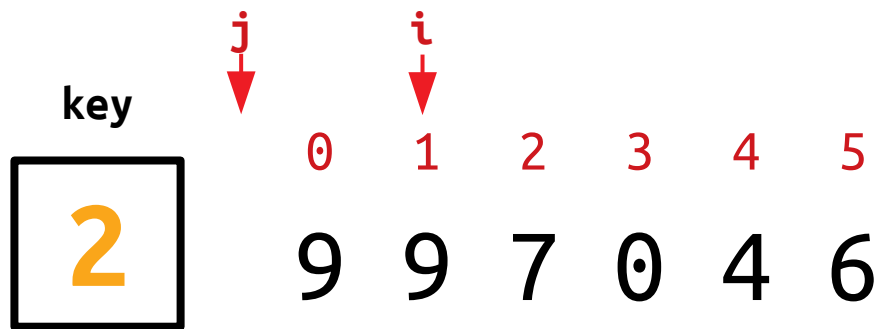
Insertion Sort



size = 6
i = 1
j = -1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

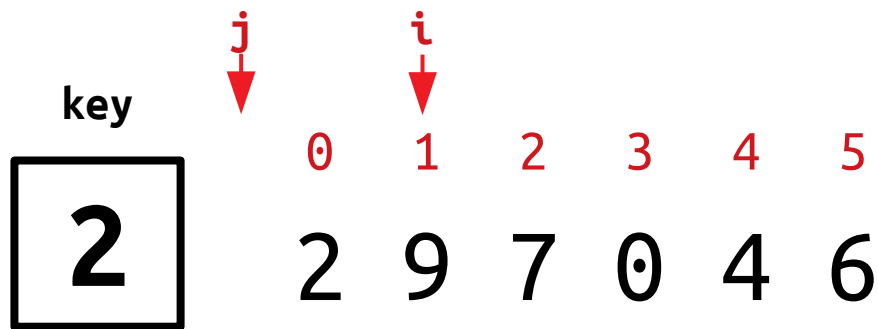
Insertion Sort



size = 6
i = 1
j = -1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

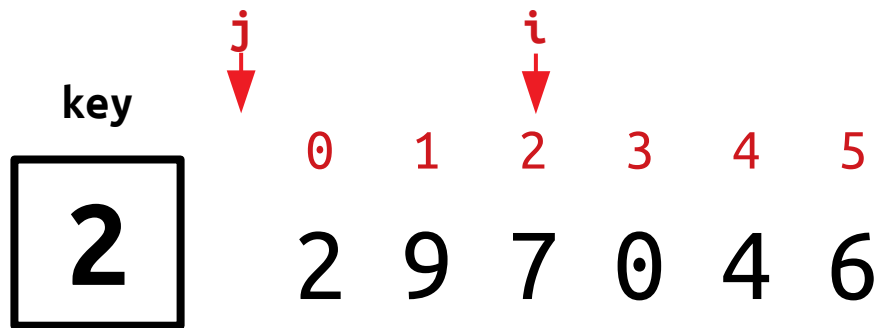
Insertion Sort



size = 6
i = 1
j = -1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

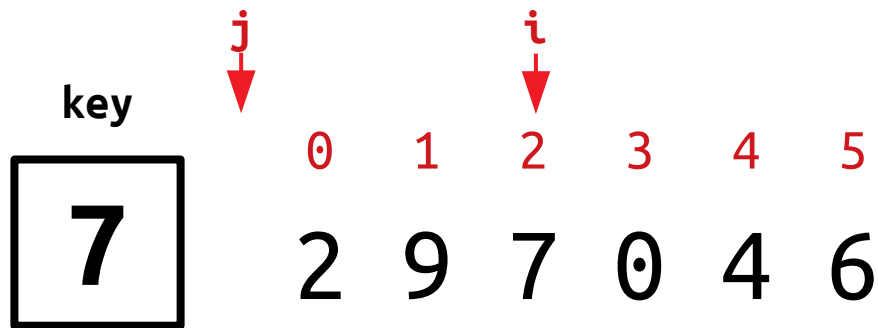
Insertion Sort



size = 6
i = 2
j = -1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort



size = 6
i = 2
j = -1

```
void insertionSort(int a[], int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```


Insertion Sort

key
7

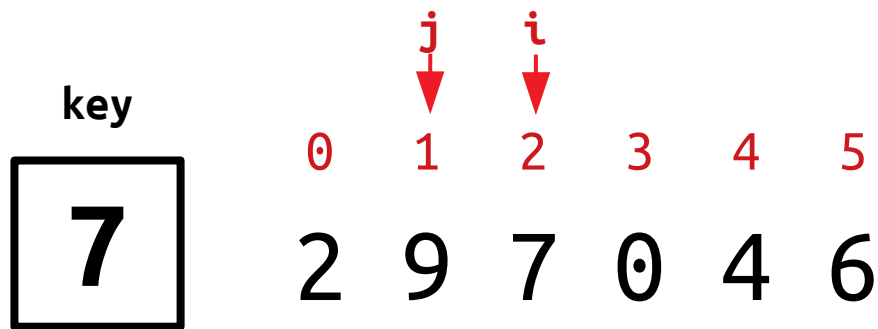
| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 9 | 7 | 0 | 4 | 6 |

Diagram illustrating the Insertion Sort process. The array contains elements [2, 9, 7, 0, 4, 6]. The current element being inserted is 7 (labeled 'key'). The element 9 is at index 1, and 7 is at index 2. Red arrows labeled 'j' and 'i' point to indices 1 and 2 respectively, indicating the current position of the element being inserted.

size = 6
i = 2
j = 1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort



size = 6
i = 2
j = 1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key
7

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 9 | 7 | 0 | 4 | 6 |

Diagram illustrating the Insertion Sort process. The array contains [2, 9, 7, 0, 4, 6]. The element 7 is the current key being inserted. Red arrows labeled 'j' and 'i' point to the positions of 9 and 7 respectively, indicating the shift of elements to the right.

size = 6
i = 2
j = 1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key
7

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 9 | 9 | 0 | 4 | 6 |

Diagram illustrating the Insertion Sort process. The array contains elements [2, 9, 9, 0, 4, 6]. The current element being inserted is 7 (key). The element 9 at index 1 is highlighted in yellow. Red arrows indicate the current position of the element being inserted (j) and the position of the element being compared (i).

size = 6
i = 2
j = 1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

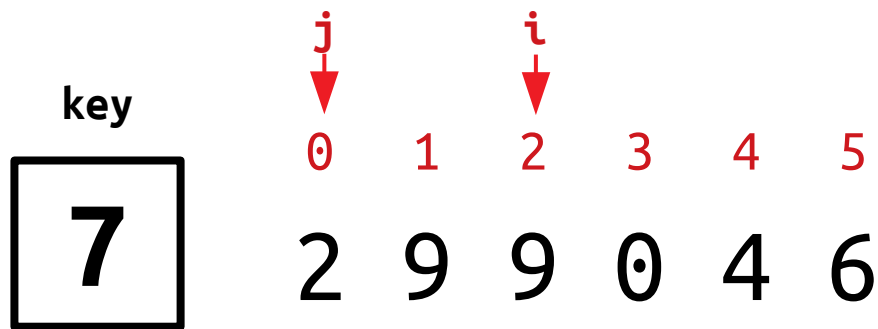
key
7

| | | | | | | |
|--------|---|--------|---|---|---|--|
| j ↓ | | i ↓ | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | |
| 2 | 9 | 9 | 0 | 4 | 6 | |

size = 6
i = 2
j = 0

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort



size = 6
i = 2
j = 0

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

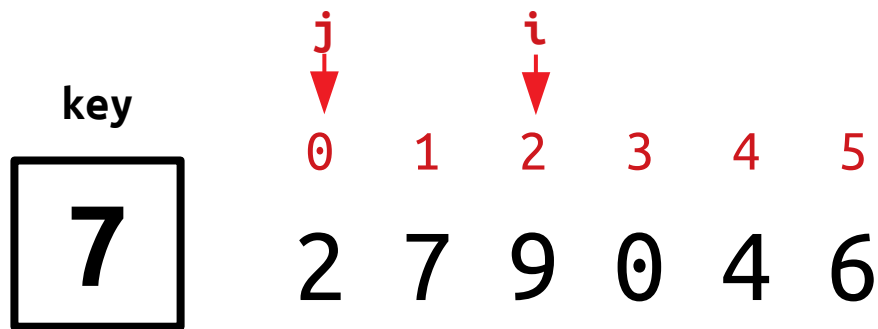
7

| | | | | | | |
|--------|---|---|--------|---|---|--|
| j ↓ | | | i ↓ | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | |
| 2 | 9 | 9 | 0 | 4 | 6 | |

size = 6
i = 2
j = 0

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

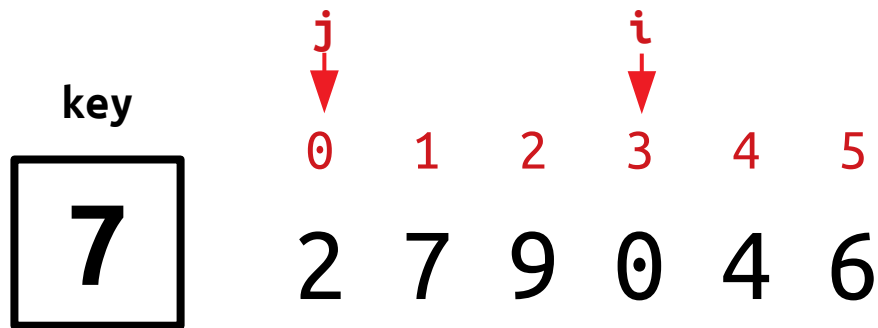
Insertion Sort



size = 6
i = 2
j = 0

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```


Insertion Sort



size = 6

i = 3

j = 0

```
void insertionSort(int a[], int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

0

| | | | | | | |
|--------|---|---|---|--------|---|--|
| j ↓ | | | | i ↓ | | |
| 0 | 1 | 2 | 3 | 4 | 5 | |
| 2 | 7 | 9 | 0 | 4 | 6 | |

size = 6
i = 3
j = 0

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key
0

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 7 | 9 | 0 | 4 | 6 |

Diagram illustrating the Insertion Sort process. The array contains elements [2, 7, 9, 0, 4, 6]. The current element being inserted is 0 (key). The element 0 is being compared with the element at index 2 (9) and is being shifted to its correct position at index 3.

size = 6
i = 3
j = 2

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

0

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 7 | 9 | 0 | 4 | 6 |

Diagram illustrating the Insertion Sort process. The array contains elements [2, 7, 9, 0, 4, 6]. The current element being inserted is 0 (labeled 'key'). The element 0 is at index 3, and the element 9 is at index 2. Red arrows indicate the shift of elements 9 and 0 to the right, and the insertion of the key (0) at index 2.

size = 6
i = 3
j = 2

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

0

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 7 | 9 | 0 | 4 | 6 |

Diagram illustrating the Insertion Sort process. The array elements are 2, 7, 9, 0, 4, 6. The current element being inserted is 0 (highlighted in yellow). The element 9 is highlighted in orange. Red arrows indicate the shift of elements from index 2 to 3 (labeled 'j') and from index 3 to 4 (labeled 'i').

size = 6
i = 3
j = 2

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

0

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 7 | 9 | 9 | 4 | 6 |

(Note: In the original image, the value 9 at index 2 is highlighted in orange, and red arrows labeled 'j' and 'i' point to indices 2 and 3 respectively.)

size = 6
i = 3
j = 2

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

0

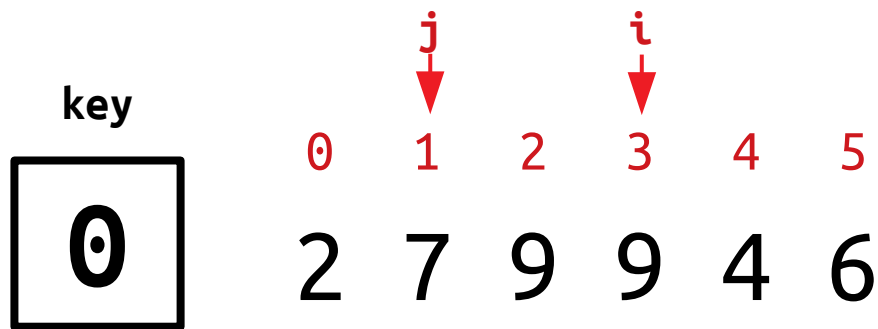
| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 7 | 9 | 9 | 4 | 6 |

Diagram illustrating the Insertion Sort process. The array contains elements [2, 7, 9, 9, 4, 6]. The current element being inserted is 0 (labeled 'key'). The array indices are shown above the elements. Red arrows indicate the current position of the element being inserted (j) and the current element being compared (i).

size = 6
i = 3
j = 1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort



size = 6
i = 3
j = 1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```


Insertion Sort

key

0

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 7 | 9 | 9 | 4 | 6 |

(Note: In the original image, red arrows point from 'j' to index 1 and from 'i' to index 3. The value 7 at index 1 is highlighted in orange.)

size = 6
i = 3
j = 1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

0

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 7 | 7 | 9 | 4 | 6 |

(Note: In the original image, the number 7 at index 1 is highlighted in yellow, and red arrows point from 'j' at index 1 and 'i' at index 3 down to the array elements.)

size = 6
i = 3
j = 1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

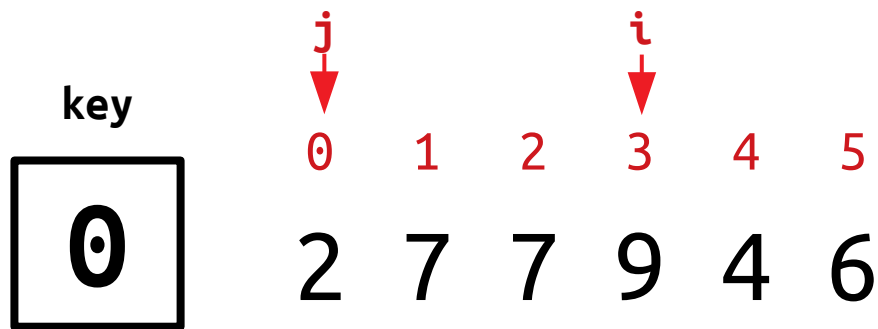
0

| | | | | | |
|--------|---|---|--------|---|---|
| j ↓ | | | i ↓ | | |
| 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 7 | 7 | 9 | 4 | 6 |

size = 6
i = 3
j = 0

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort



size = 6
i = 3
j = 0

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

0

| | | | | | |
|--------|---|---|--------|---|---|
| j ↓ | | | i ↓ | | |
| 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 7 | 7 | 9 | 4 | 6 |

size = 6
i = 3
j = 0

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

0

| | | | | | |
|--------|---|---|--------|---|---|
| j ↓ | | | i ↓ | | |
| 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 2 | 7 | 9 | 4 | 6 |

size = 6
i = 3
j = 0

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

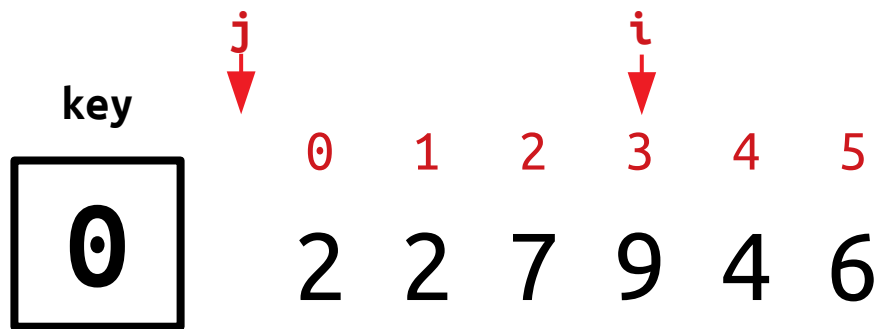
Insertion Sort



size = 6
i = 3
j = -1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort



size = 6
i = 3
j = -1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

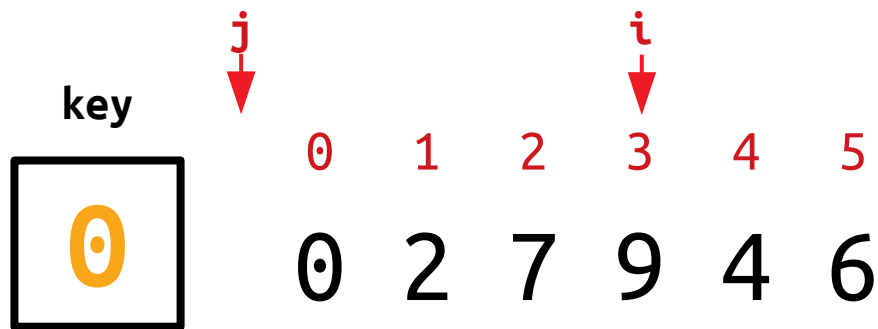

Insertion Sort



size = 6
i = 3
j = -1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

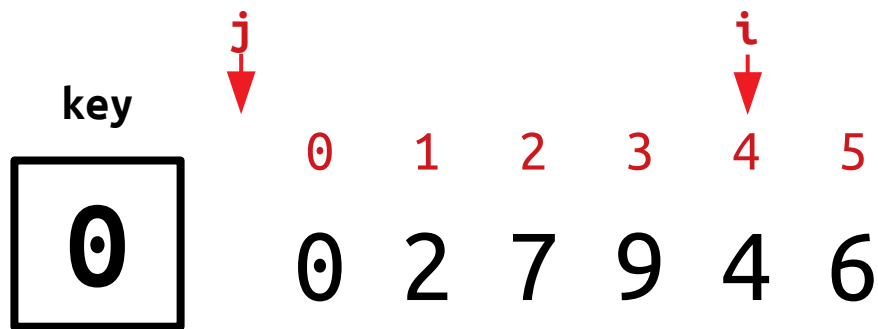
Insertion Sort



size = 6
i = 3
j = -1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

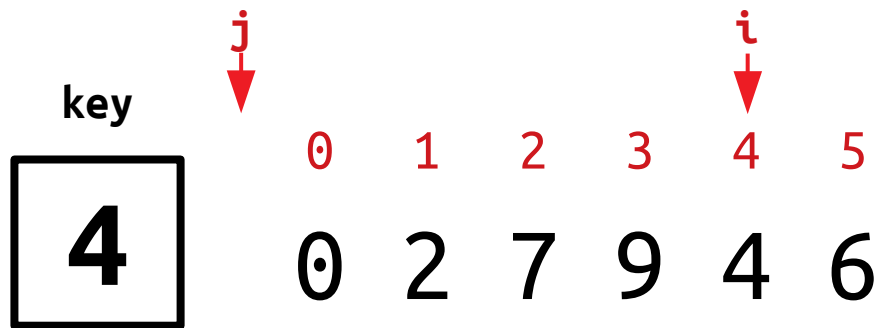
Insertion Sort



size = 6
i = 4
j = -1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort



size = 6
i = 4
j = -1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key
4

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 7 | 9 | 4 | 6 |

Diagram illustrating the Insertion Sort process. The array contains elements [0, 2, 7, 9, 4, 6]. The current element being inserted is 4 (labeled 'key'). The element 9 is at index 3, and 4 is at index 4. Red arrows labeled 'j' and 'i' point to indices 3 and 4 respectively, indicating the current position of the element being inserted.

size = 6
i = 4
j = 3

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key
4

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 7 | 9 | 4 | 6 |

Diagram illustrating the Insertion Sort process. The array contains elements [0, 2, 7, 9, 4, 6]. The current element being inserted is 4 (labeled 'key'). The element 9 is at index 3, and 4 is at index 4. Red arrows labeled 'j' and 'i' point to indices 3 and 4 respectively, indicating the current position of the element being inserted.

size = 6
i = 4
j = 3

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key
4

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 7 | 9 | 4 | 6 |

Diagram illustrating the Insertion Sort process. The array contains elements [0, 2, 7, 9, 4, 6]. The current element being inserted is 4 (labeled 'key'). The element 9 is at index 3, and 4 is at index 4. Red arrows labeled 'j' and 'i' point to indices 3 and 4 respectively, indicating the current position of the element being inserted.

size = 6
i = 4
j = 3

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key
4

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 7 | 9 | 9 | 6 |

Diagram illustrating the Insertion Sort process. The array contains elements [0, 2, 7, 9, 9, 6]. The current element being inserted is 4 (labeled 'key'). The element 9 at index 3 is highlighted in yellow. Red arrows labeled 'j' and 'i' point to index 3 and index 4 respectively, indicating the current position of the element being inserted.

size = 6
i = 4
j = 3

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```


Insertion Sort

key
4

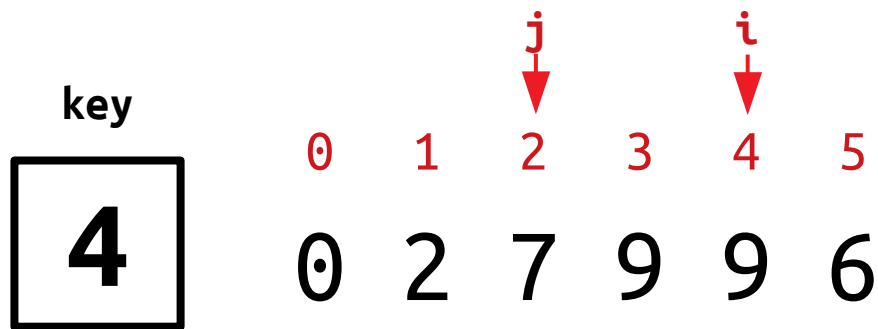
| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 7 | 9 | 9 | 6 |

Diagram illustrating the Insertion Sort process. The array contains elements [0, 2, 7, 9, 9, 6]. The current element being inserted is 4 (labeled 'key'). The element 4 is being compared with the element at index 2 (7) and index 4 (9). Red arrows indicate the comparison points. The element 4 is being shifted to its correct position.

size = 6
i = 4
j = 2

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort



size = 6
i = 4
j = 2

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key
4

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 7 | 9 | 9 | 6 |

Diagram illustrating the Insertion Sort process. The array contains elements [0, 2, 7, 9, 9, 6]. The current element being inserted is 4 (labeled 'key'). The element 7 is highlighted in yellow. Red arrows indicate the current position 'j' at index 2 and the position 'i' at index 4.

size = 6
i = 4
j = 2

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key
4

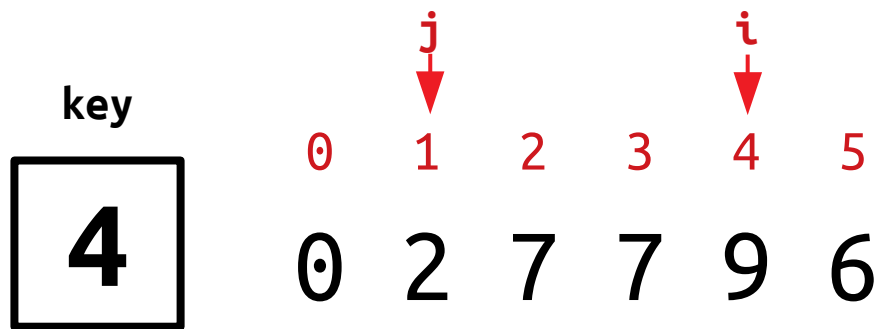
| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 7 | 7 | 9 | 6 |

Diagram illustrating the Insertion Sort process. The array is [0, 2, 7, 7, 9, 6]. The element 4 is the key being inserted. The current element being compared is 7 at index 2 (j), and the element being shifted is 9 at index 4 (i).

size = 6
i = 4
j = 2

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

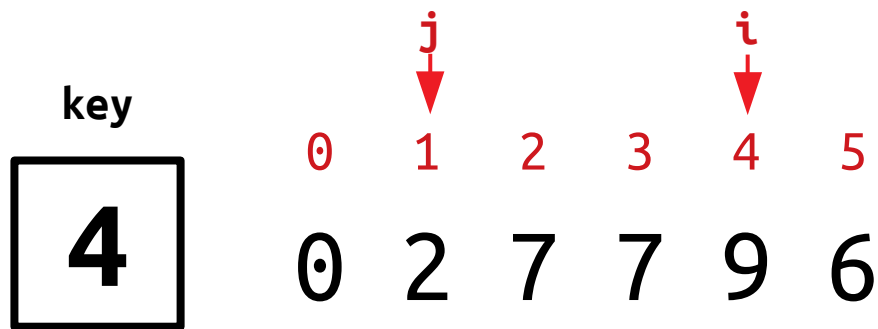
Insertion Sort



size = 6
i = 4
j = 1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

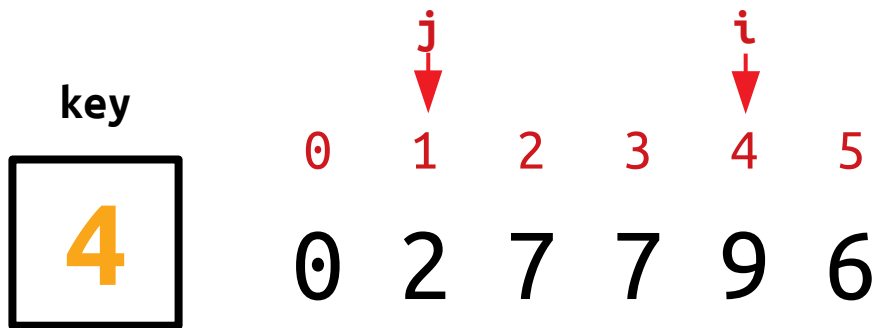
Insertion Sort



size = 6
i = 4
j = 1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

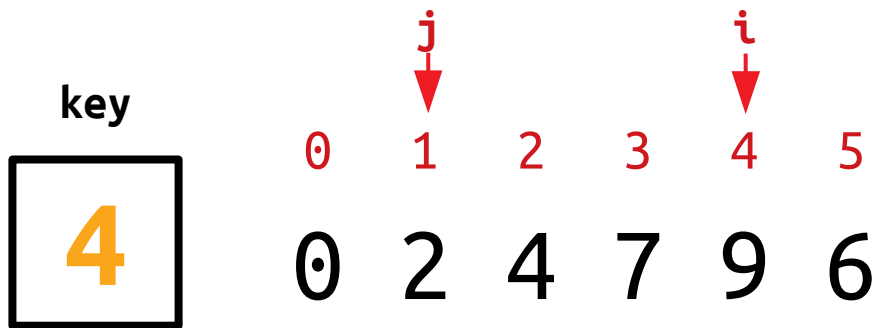
Insertion Sort



size = 6
i = 4
j = 1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

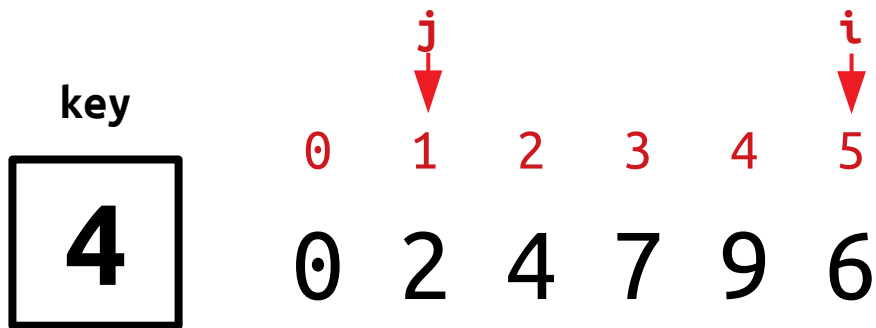
Insertion Sort



size = 6
i = 4
j = 1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

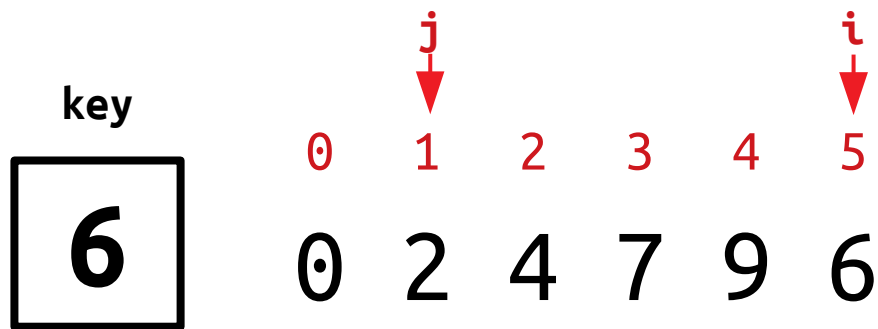

Insertion Sort



size = 6
i = 5
j = 1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort



size = 6
i = 5
j = 1

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key
6

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 4 | 7 | 9 | 6 |

Diagram illustrating the Insertion Sort process. The array contains elements [0, 2, 4, 7, 9, 6]. The current element being inserted is 6 (labeled 'key'). The element 9 is at index 4, and 6 is at index 5. Red arrows indicate the shift of elements from index 4 to index 5, and the insertion of 6 at index 4.

size = 6
i = 5
j = 4

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

| |
|---|
| 6 |
|---|

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 4 | 7 | 9 | 6 |

Diagram illustrating the Insertion Sort process. The array contains elements [0, 2, 4, 7, 9, 6]. The current element being inserted is 6 (labeled 'key'). The element 9 is at index 4, and 6 is at index 5. Red arrows indicate the shift of elements from index 4 to index 5, and the insertion of the key at index 4.

size = 6
i = 5
j = 4

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

6

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 4 | 7 | 9 | 6 |

(Note: In the original image, red arrows point from index 4 to index 5 and from index 5 to index 4, indicating a swap.)

size = 6
i = 5
j = 4

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

6

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 4 | 7 | 9 | 9 |

(Note: Red arrows point from index 4 to 3 and index 5 to 4. The value 9 at index 4 is highlighted in orange.)

size = 6
i = 5
j = 4

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

6

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 4 | 7 | 9 | 9 |

Diagram illustrating the Insertion Sort process. The array contains elements [0, 2, 4, 7, 9, 9] at indices 0 to 5. The current element being inserted (key) is 6. The element at index 3 (7) is the current element being compared with the key. The element at index 4 (9) is the element being shifted to the right. The element at index 5 (9) is the element being shifted to the right.

size = 6
i = 5
j = 3

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

6

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 4 | 7 | 9 | 9 |

(Red arrows point from index 3 to index 4 and from index 5 to index 4)

size = 6
i = 5
j = 3

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```


Insertion Sort

key

6

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 4 | 7 | 9 | 9 |

Diagram illustrating the Insertion Sort process. The array contains elements [0, 2, 4, 7, 9, 9] at indices 0 to 5. The element 6 (key) is being inserted. The current element being compared is 7 at index 3 (labeled 'j' with a red arrow). The element at index 5 (labeled 'i' with a red arrow) is 9.

size = 6
i = 5
j = 3

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

6

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 4 | 7 | 7 | 9 |

(Note: In the original image, red arrows point from index 3 to index 4 and from index 5 to index 4. The value 7 at index 3 is highlighted in yellow.)

size = 6
i = 5
j = 3

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key
6

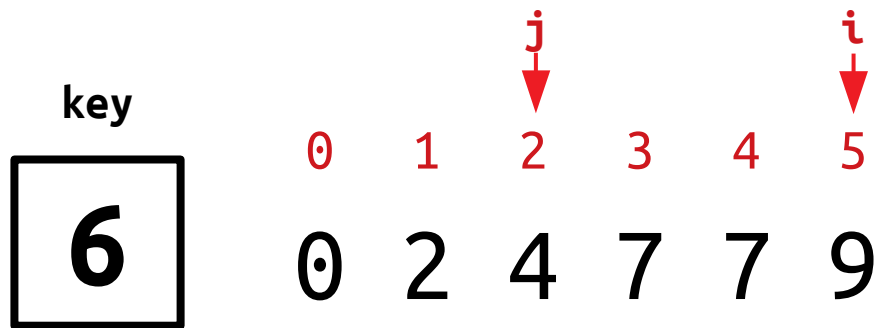
| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 4 | 7 | 7 | 9 |

Diagram illustrating the Insertion Sort process. The array contains elements [0, 2, 4, 7, 7, 9] at indices 0 to 5. The current element being inserted is 6 (key). The element 4 at index 2 is labeled 'j' with a red arrow pointing down. The element 9 at index 5 is labeled 'i' with a red arrow pointing down.

size = 6
i = 5
j = 2

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort



size = 6
i = 5
j = 2

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

6

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 4 | 7 | 7 | 9 |

(Red arrows point from indices 2 and 5 to the values 4 and 9 respectively)

size = 6
i = 5
j = 2

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1 ; i < size ; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

key

6

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 4 | 6 | 7 | 9 |

Diagram illustrating the Insertion Sort process. The array contains elements [0, 2, 4, 6, 7, 9]. The current element being inserted is 6 (labeled 'key'). The element 4 is at index 2, and the element 9 is at index 5. Red arrows indicate the current position of the element being inserted (j) and the current element being compared (i).

size = 6
i = 5
j = 2

```
void insertionSort(int a*, int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```

Insertion Sort

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 2 | 4 | 6 | 7 | 9 |

Ordered!

```
void insertionSort(int a[], int size) {  
    int i, j, key;  
    for (i = 1; i < size; i++) {  
        key = a[i];  
        j = i - 1;  
        while ((j >= 0) && (a[j] > key)) {  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = key;  
    }  
}
```