Data Structure and Algorithms

Lecture 4 & 5 : Sort & Search Algorithms on Array

Things to cover

- Interpolation Search
- Quick Sort

Quick Sort

```
// Function to partition the array around a pivot
int partition(int arr[], int low, int high) {
    int pivot = arr[high]; // Choosing the last element as pivot
    int i = (low - 1); // Index of smaller element

    for (int j = low; j <= high - 1; j++) {
        // If current element is smaller than or equal to pivot
        if (arr[j] <= pivot) {
            i++; // Increment index of smaller element
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}
```

```
// Main Quick Sort function
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        // pi is partitioning index, arr[pi] is now at right place
        int pi = partition(arr, low, high);

        // Separately sort elements before partition and after partition
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
```

Explanation

Arr = {50, 30, 10, 20, 60, 40}

low= 0, high = 5

For j = 0 to 4

```
if (arr[j] <= pivot) {
            i++; // Increment index of smaller element
            swap(&arr[i], &arr[j]);
        }
```

pivot = 40,
i = -1

j = 0, arr[0] > pivot → DO NOTHING

j = 1, arr[1] < pivot → i = 0, swap arr[0] and arr[1] — Arr = {30, 50, 10, 20, 60, 40}

j = 2, arr[2] < pivot → i = 1, swap arr[2] and arr[1] — Arr = {30, 10, 50, 20, 60, 40}

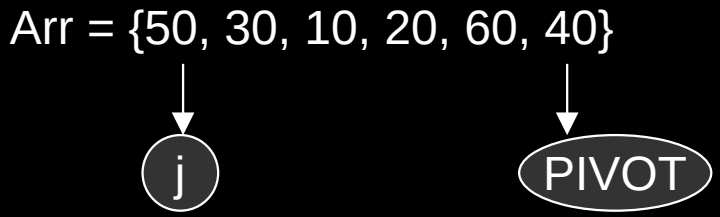
j = 3, arr[3] < pivot → i = 2, swap arr[3] and arr[2] — Arr = {30, 10, 20, 50, 60, 40}

j = 4, arr[4] > pivot → DO NOTHING

swap arr[3] and arr[5] — Arr = {30, 10, 20, 40, 60, 50}

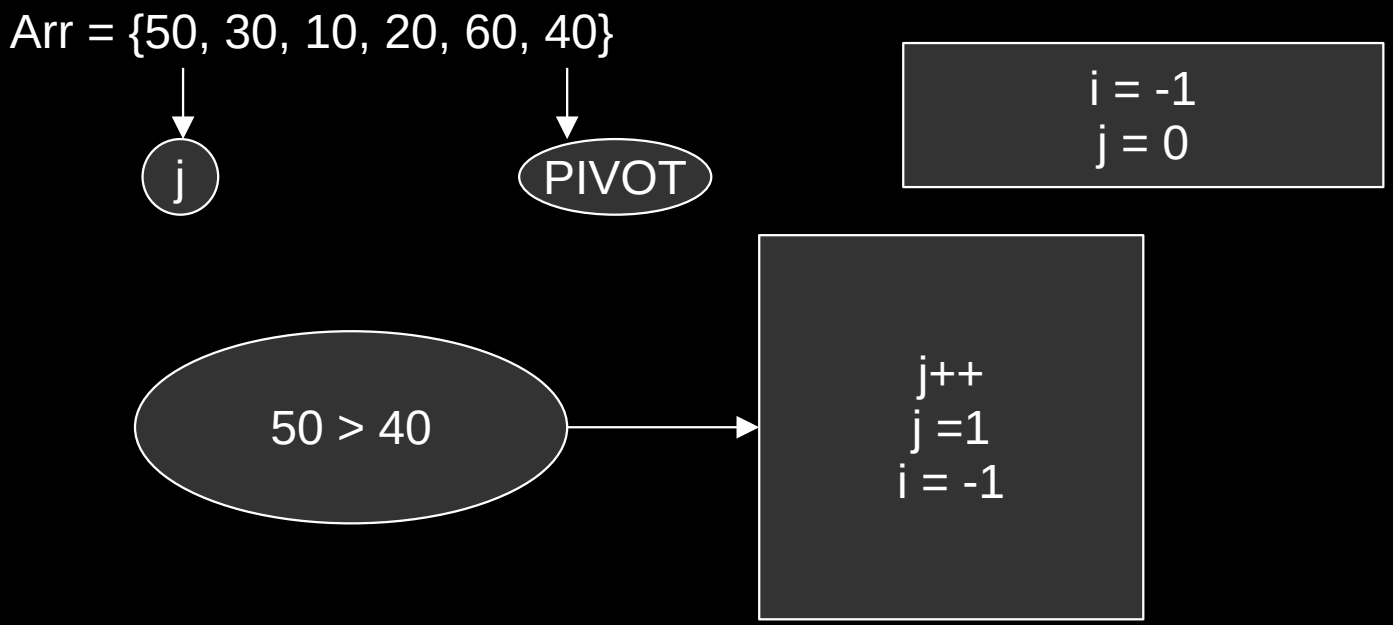
Return 3

Explanation

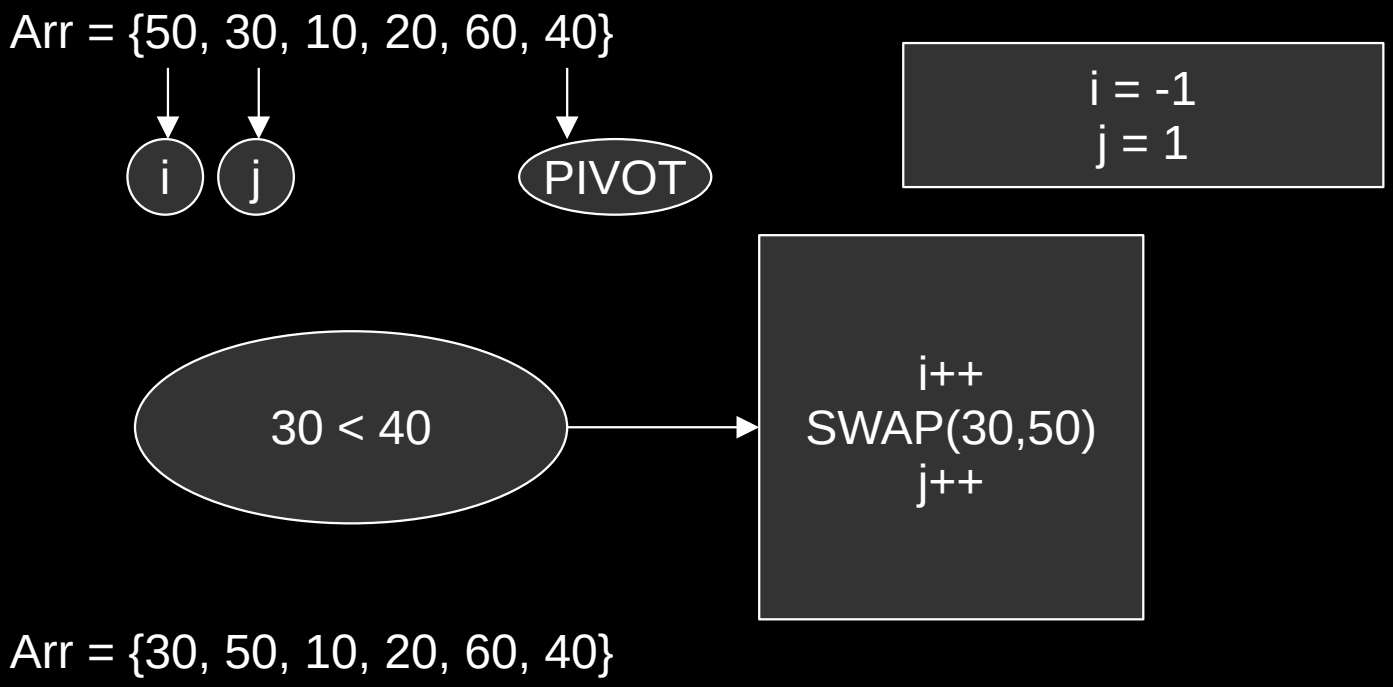


i = -1
j = 0

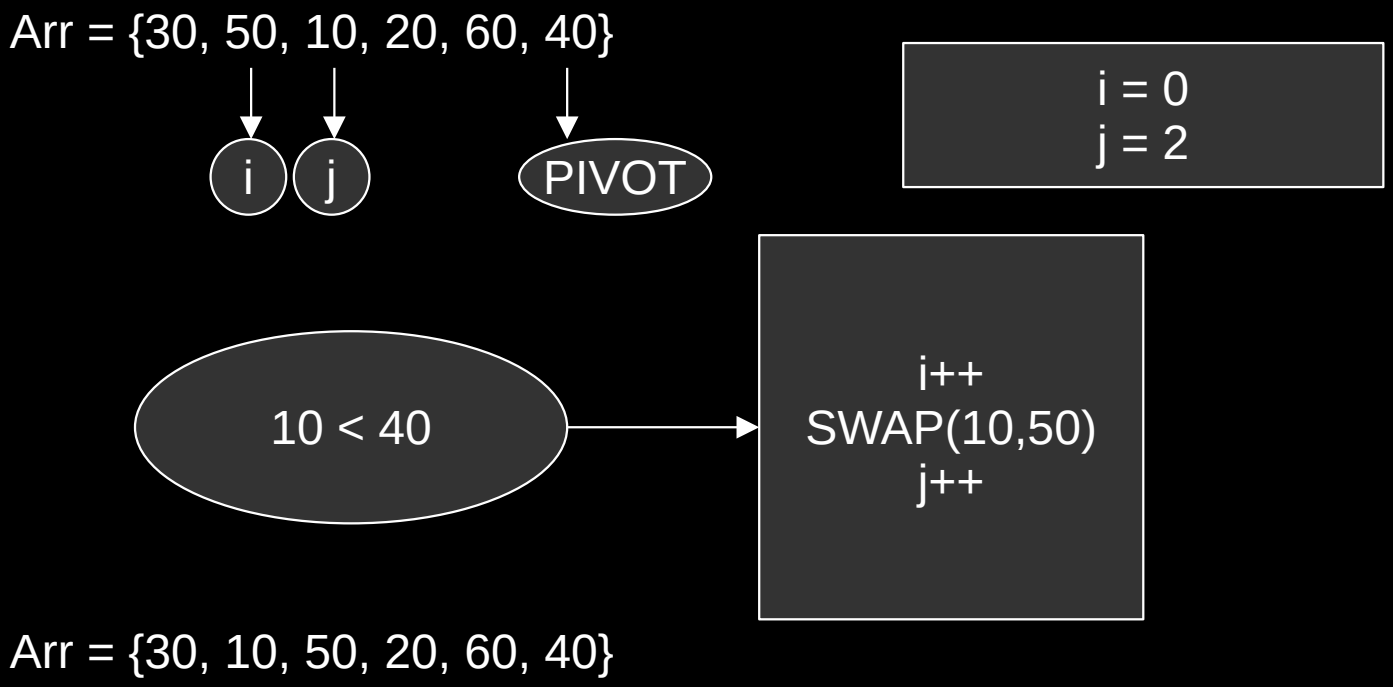
Explanation



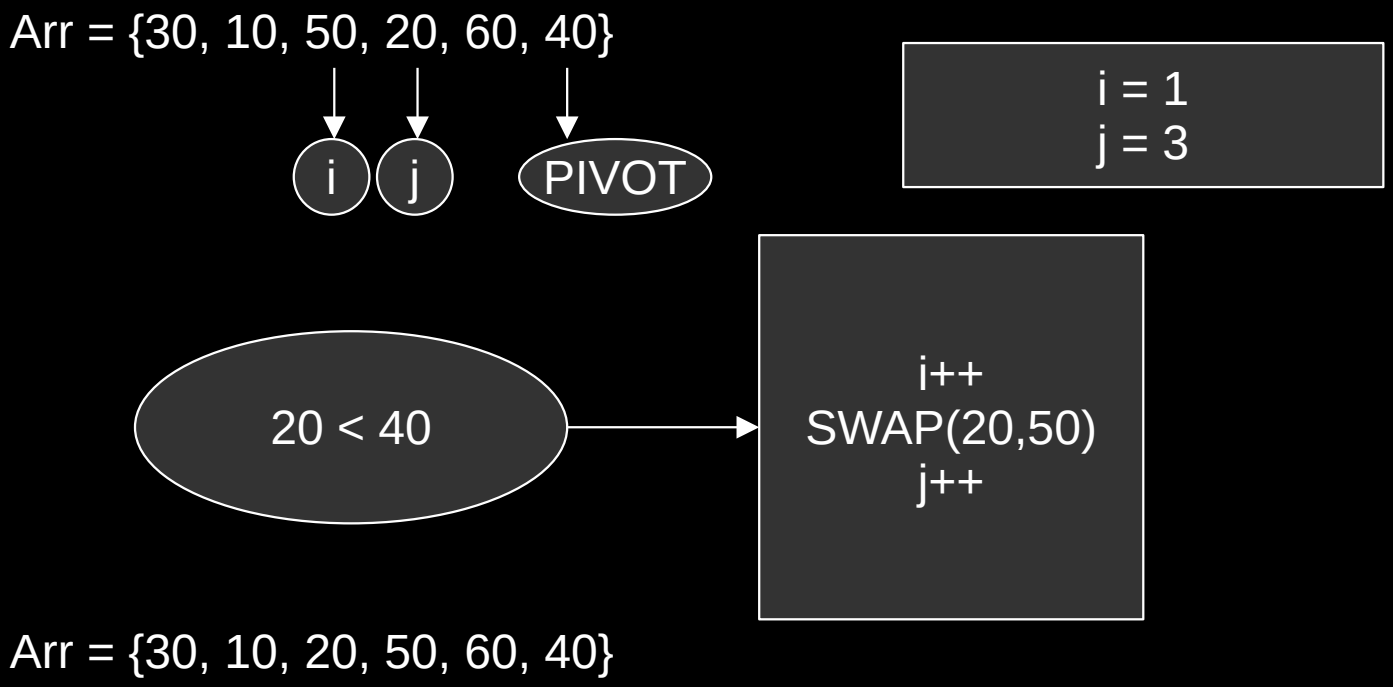
Explanation



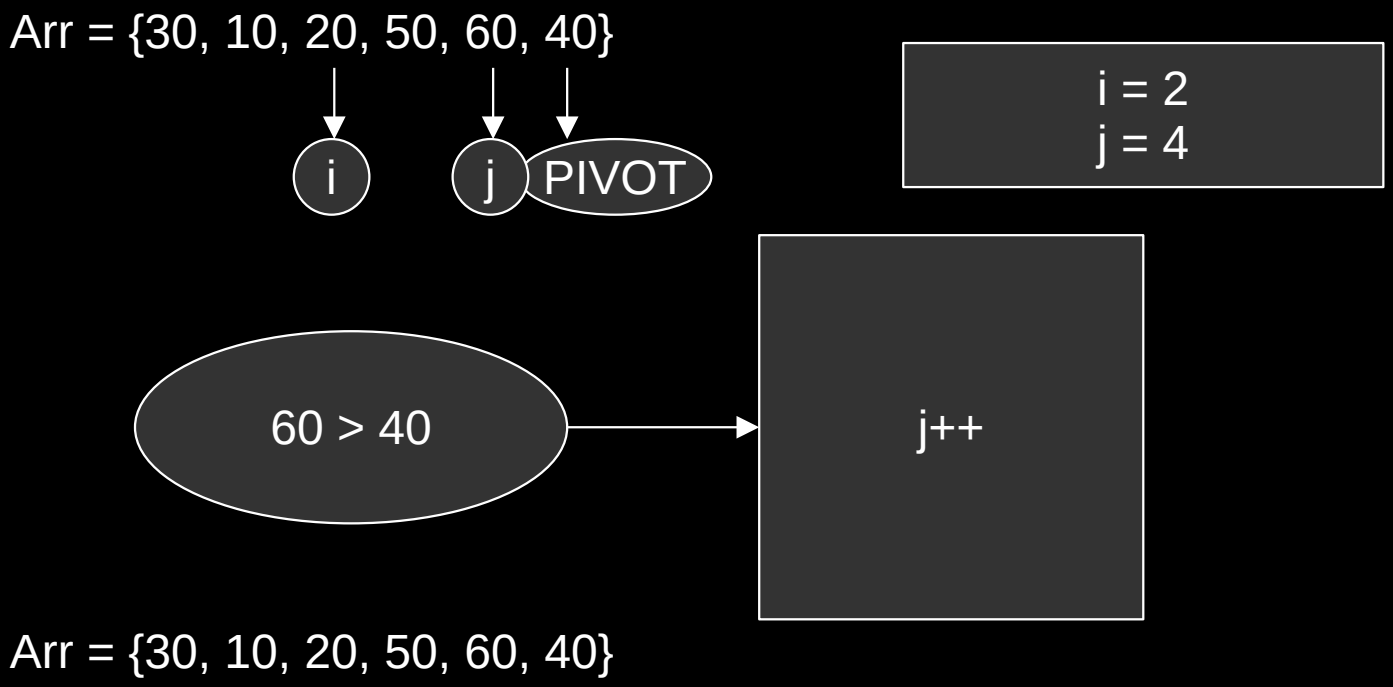
Explanation



Explanation

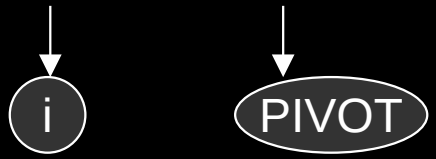


Explanation



Explanation

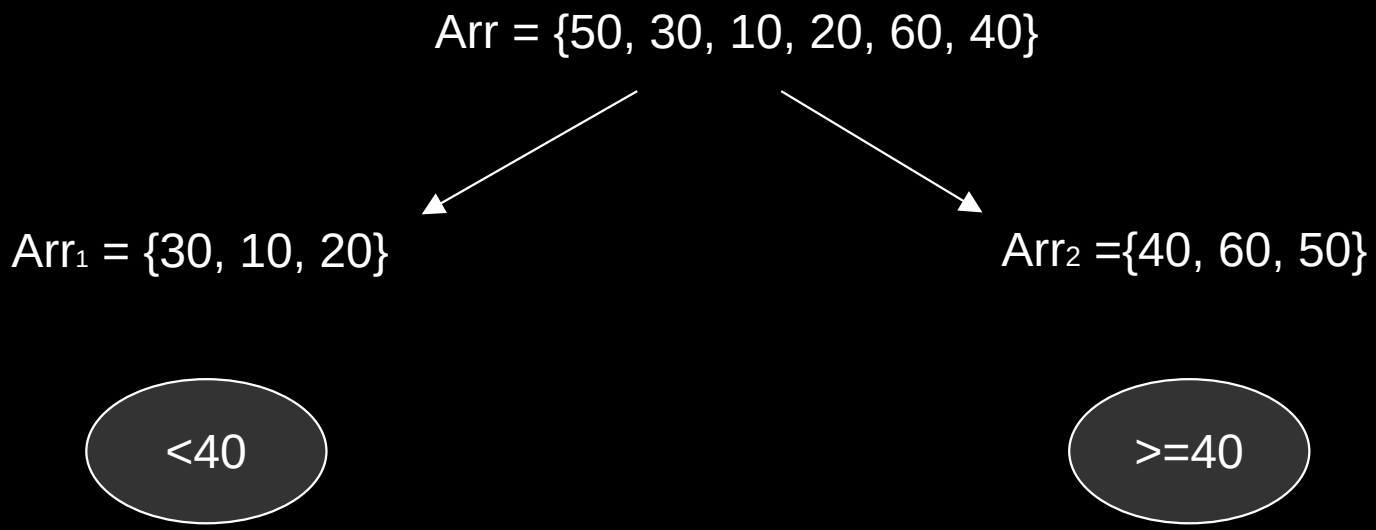
Arr = {30, 10, 20, 50, 60, 40}



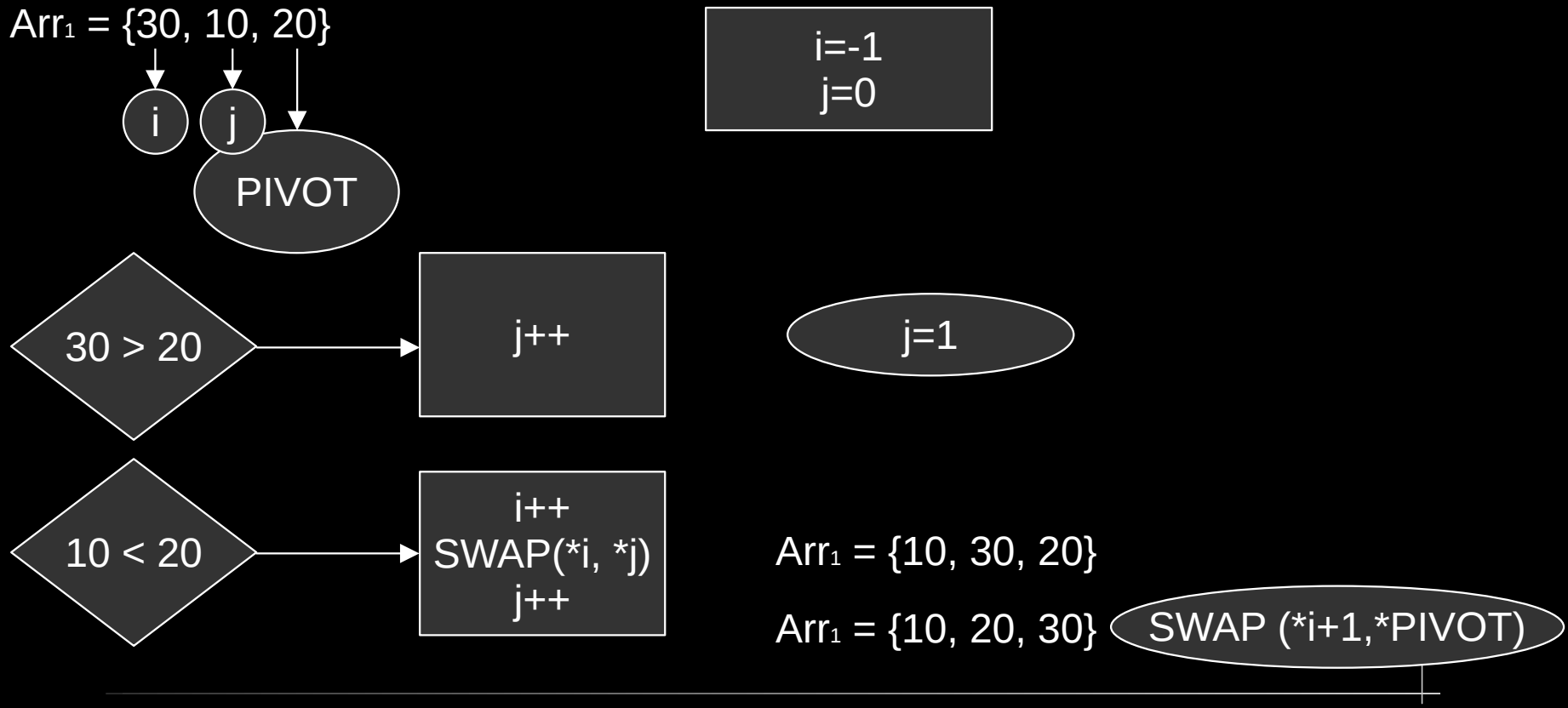
Arr = {30, 10, 20, 40, 60, 50}

SWAP (*i+1,*PIVOT)

Explanation



Explanation

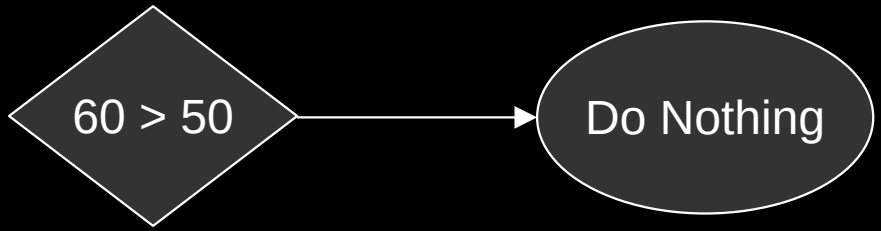


Explanation

Arr = {10, 20, 30, 40, 60, 50}

Arr_1 = {60,50}

i=3
j=4



Arr = {10, 20, 30, 40, 50, 60}

SWAP (*i+1,*PIVOT)

Interpolation Search

```
int interpolationSearch(int arr[], int n, int target) {
    int low = 0;
    int high = n - 1;

    while (low <= high && target >= arr[low] && target <= arr[high]) {
        // Handle division by zero if arr[high] == arr[low]
        if (arr[high] == arr[low]) {
            if (arr[low] == target) {
                return low; // Target found if all elements are same
            } else {
                return -1; // Target not found
            }
        }

        int pos = low + ((target - arr[low]) * (high - low)) / (arr[high] - arr[low]);

        if (arr[pos] == target) {
            return pos;
        } else if (arr[pos] < target) {
            low = pos + 1;
        } else {
            high = pos - 1;
        }
    }
    return -1; // Element not found
}
```

Linked List Introduction

Linear Data Structure

Elements are not stored in Contiguous Memory

Singly Linked List

Doubly Linked List

Circular Linked List