Selection Sort

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.

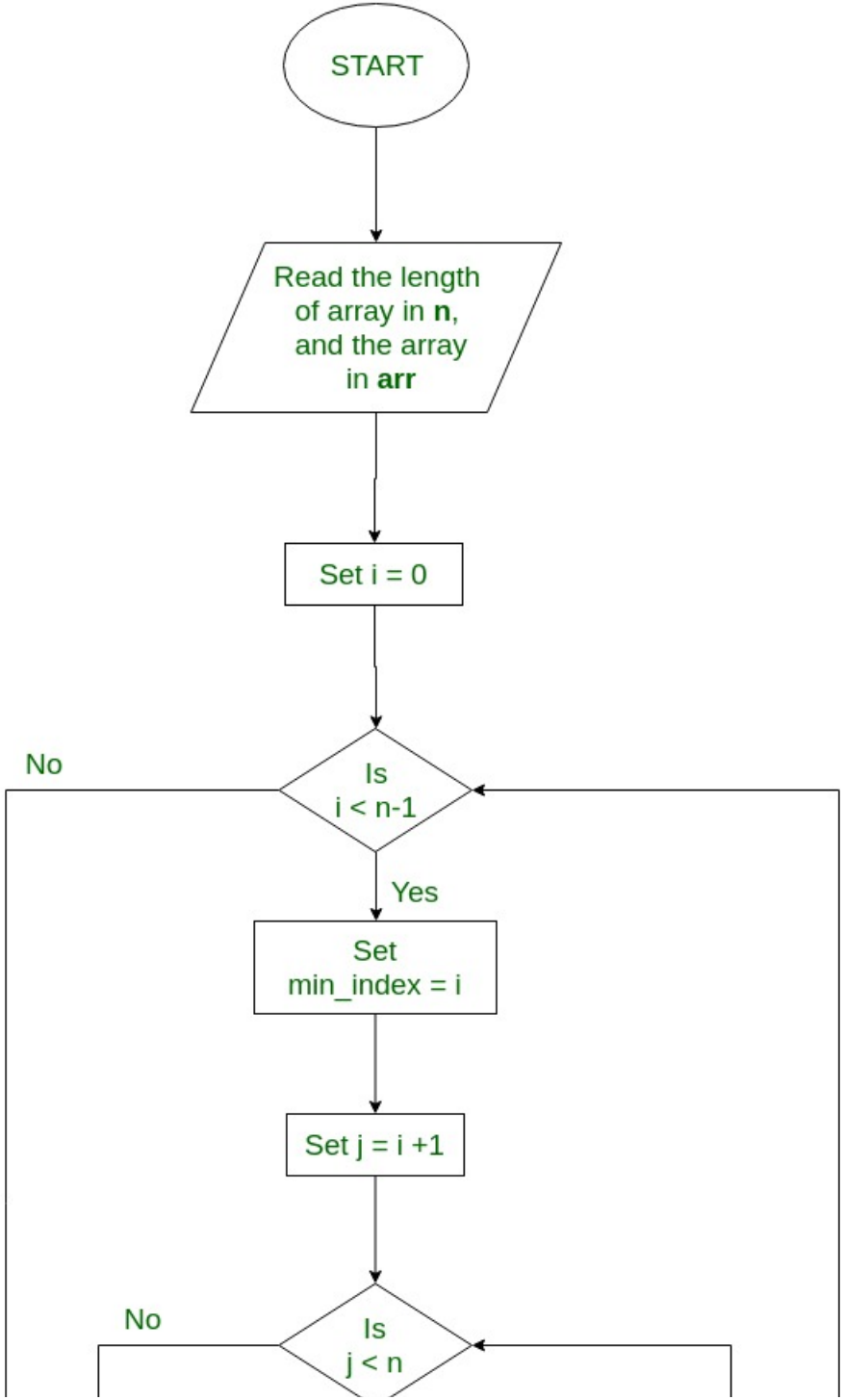
1) The subarray which is already sorted.
2) Remaining subarray which is unsorted.

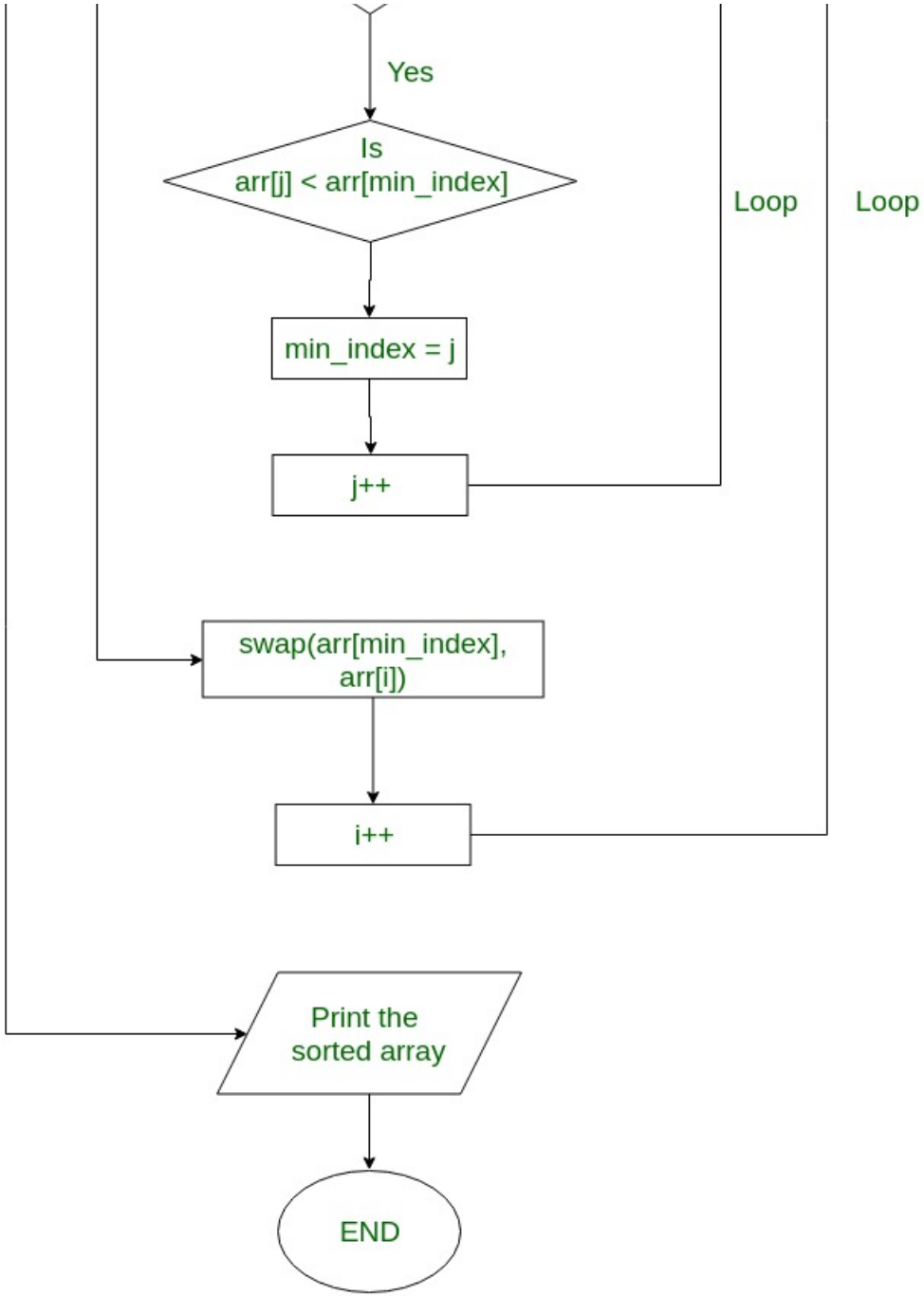
In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.

Following example explains the above steps:

```
arr[] = 64 25 12 22 11

// Find the minimum element in arr[0...4]
// and place it at beginning
11 25 12 22 64

// Find the minimum element in arr[1...4]
// and place it at beginning of arr[1...4]
11 12 25 22 64

// Find the minimum element in arr[2...4]
// and place it at beginning of arr[2...4]
11 12 22 25 64

// Find the minimum element in arr[3...4]
// and place it at beginning of arr[3...4]
11 12 22 25 64
```

Flowchart of the Selection Sort:


START
Read the length of array in n, and the array in arr
Set i = 0
No
Is i < n-1
Yes
Set min_index = i
Set j = i +1
No
Is j < n



Flowchart for Selection Sort

Flowchart for Selection Sort

Recommended: Please solve it on "*PRACTICE*" first, before moving on to the solution.

C++

```
// C++ program for implementation of selection sort
#include <bits/stdc++.h>
using namespace std;

void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

void selectionSort(int arr[], int n)
{
    int i, j, min_idx;

    // One by one move boundary of unsorted subarray
    for (i = 0; i < n-1; i++)
    {
        // Find the minimum element in unsorted array
        min_idx = i;
        for (j = i+1; j < n; j++)
        if (arr[j] < arr[min_idx])
            min_idx = j;

        // Swap the found minimum element with the first element
        swap(&arr[min_idx], &arr[i]);
    }
}

/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;
}

// Driver program to test above functions
int main()
{
```

```
    int arr[] = {64, 25, 12, 22, 11};
    int n = sizeof(arr)/sizeof(arr[0]);
    selectionSort(arr, n);
    cout << "Sorted array: \n";
    printArray(arr, n);
    return 0;
}

// This is code is contributed by rathbhupendra
```

C

```
// C program for implementation of selection sort
#include <stdio.h>

void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

void selectionSort(int arr[], int n)
{
    int i, j, min_idx;

    // One by one move boundary of unsorted subarray
    for (i = 0; i < n-1; i++)
    {
        // Find the minimum element in unsorted array
        min_idx = i;
        for (j = i+1; j < n; j++)
          if (arr[j] < arr[min_idx])
            min_idx = j;

        // Swap the found minimum element with the first element
        swap(&arr[min_idx], &arr[i]);
    }
}

/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
```

```
// Driver program to test above functions
int main()
{
    int arr[] = {64, 25, 12, 22, 11};
    int n = sizeof(arr)/sizeof(arr[0]);
    selectionSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}
```

Python

```
# Python program for implementation of Selection
# Sort
import sys
A = [64, 25, 12, 22, 11]

# Traverse through all array elements
for i in range(len(A)):

    # Find the minimum element in remaining
    # unsorted array
    min_idx = i
    for j in range(i+1, len(A)):
        if A[min_idx] > A[j]:
            min_idx = j

    # Swap the found minimum element with
    # the first element
    A[i], A[min_idx] = A[min_idx], A[i]

# Driver code to test above
print ("Sorted array")
for i in range(len(A)):
    print("%d" %A[i]),
```

Java

```
// Java program for implementation of Selection Sort
class SelectionSort
{
    void sort(int arr[])
    {
        int n = arr.length;
```

```
        // One by one move boundary of unsorted subarray
        for (int i = 0; i < n-1; i++)
        {
            // Find the minimum element in unsorted array
            int min_idx = i;
            for (int j = i+1; j < n; j++)
                if (arr[j] < arr[min_idx])
                    min_idx = j;

            // Swap the found minimum element with the first
            // element
            int temp = arr[min_idx];
            arr[min_idx] = arr[i];
            arr[i] = temp;
        }
    }

    // Prints the array
    void printArray(int arr[])
    {
        int n = arr.length;
        for (int i=0; i<n; ++i)
            System.out.print(arr[i]+" ");
        System.out.println();
    }

    // Driver code to test above
    public static void main(String args[])
    {
        SelectionSort ob = new SelectionSort();
        int arr[] = {64,25,12,22,11};
        ob.sort(arr);
        System.out.println("Sorted array");
        ob.printArray(arr);
    }
}
/* This code is contributed by Rajat Mishra*/
```

C#

```
// C# program for implementation
// of Selection Sort
using System;

class GFG
{
    static void sort(int []arr)
```

```
    {
        int n = arr.Length;

        // One by one move boundary of unsorted subarray
        for (int i = 0; i < n - 1; i++)
        {
            // Find the minimum element in unsorted array
            int min_idx = i;
            for (int j = i + 1; j < n; j++)
                if (arr[j] < arr[min_idx])
                    min_idx = j;

            // Swap the found minimum element with the first
            // element
            int temp = arr[min_idx];
            arr[min_idx] = arr[i];
            arr[i] = temp;
        }
    }

    // Prints the array
    static void printArray(int []arr)
    {
        int n = arr.Length;
        for (int i=0; i<n; ++i)
            Console.Write(arr[i]+" ");
        Console.WriteLine();
    }

    // Driver code
    public static void Main()
    {
        int []arr = {64,25,12,22,11};
        sort(arr);
        Console.WriteLine("Sorted array");
        printArray(arr);
    }

}
// This code is contributed by Sam007
```

PHP

```
<?php
// PHP program for implementation
// of selection sort
function selection_sort(&$arr, $n)
{
```

```
    for($i = 0; $i < $n ; $i++)
    {
        $low = $i;
        for($j = $i + 1; $j < $n ; $j++)
        {
            if ($arr[$j] < $arr[$low])
            {
                $low = $j;
            }
        }

        // swap the minimum value to $ith node
        if ($arr[$i] > $arr[$low])
        {
            $tmp = $arr[$i];
            $arr[$i] = $arr[$low];
            $arr[$low] = $tmp;
        }
    }
}

// Driver Code
$arr = array(64, 25, 12, 22, 11);
$len = count($arr);
selection_sort($arr, $len);
echo "Sorted array : \n";

for ($i = 0; $i < $len; $i++)
    echo $arr[$i] . " ";

// This code is contributed
// by Deepika Gupta.
?>
```

Output:

```
Sorted array:
11 12 22 25 64
```

Time Complexity: O(n^2) as there are two nested loops.

Auxiliary Space: O(1)

The good thing about selection sort is it never makes more than O(n) swaps and can be useful when memory write is a costly operation.

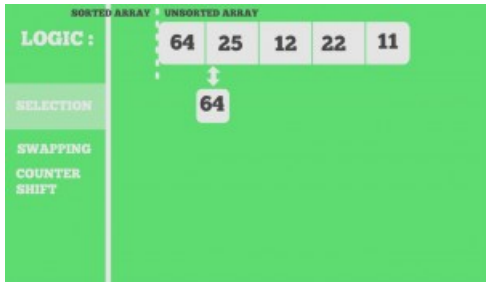
Exercise :

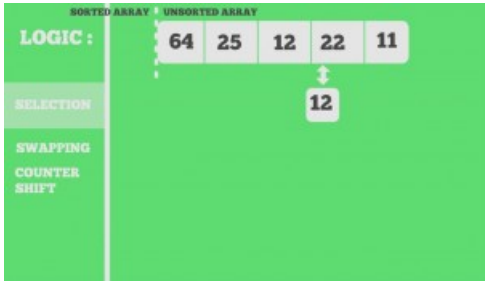
Sort an array of strings using Selection Sort

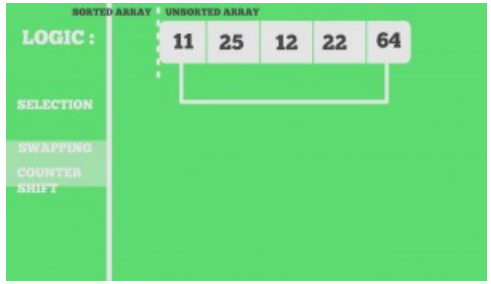
Stability : The default implementation is not stable. However it can be made stable. Please see stable selection sort for details.

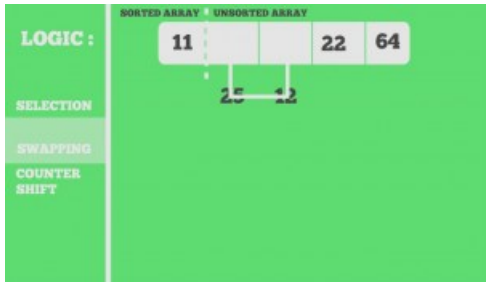
In Place : Yes, it does not require extra space.

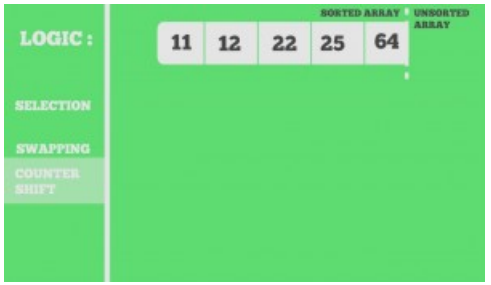
Snapshots:











Quiz on Selection Sort

Other Sorting Algorithms on GeeksforGeeks/GeeksQuiz:

- Bubble Sort
- Insertion Sort
- Merge Sort
- Heap Sort
- QuickSort
- Radix Sort
- Counting Sort
- Bucket Sort
- ShellSort

Coding practice for sorting.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Recommended Posts:

Comparison among Bubble Sort, Selection Sort and Insertion Sort

Program to sort an array of strings using Selection Sort

Recursive Selection Sort

Stable Selection Sort

8086 program for selection sort

Iterative selection sort for linked list

C++ program for Sorting Dates using Selection Sort

A sorting algorithm that slightly improves on selection sort

Recursive selection sort for singly linked list | Swapping node links

Job Selection Problem - Loss Minimization Strategy | Set 2

Why Quick Sort preferred for Arrays and Merge Sort for Linked Lists?

Bucket Sort To Sort an Array with Negative Numbers

Insertion sort to sort even and odd positioned elements in different orders

Odd Even Transposition Sort / Brick Sort using pthreads

Java Program for Odd-Even Sort / Brick Sort

Improved By : DeepikaPathak, RishiAdvani, rathbhupendra

Article Tags : Sorting Medlife

Practice Tags : Medlife Sorting

55

1.5

To-do Done

Based on **148** vote(s)

Feedback/ Suggest Improvement Add Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY

About Us
Careers
Privacy Policy
Contact Us

LEARN

Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE

Courses
Company-wise
Topic-wise
How to begin?

CONTRIBUTE

Write an Article
Write Interview Experience
Internships
Videos