

Home	Data Structure	C	C++	C#	Java	SQL	HTML	CSS	JavaScript	Ajax
------	----------------	---	-----	----	------	-----	------	-----	------------	------

Selection Sort Algorithm

In this article, we will discuss the Selection sort Algorithm. The working procedure of selection sort is also simple. This article will be very helpful and interesting to students as they might face selection sort as a question in their examinations. So, it is important to discuss the topic.

In selection sort, the smallest value among the unsorted elements of the array is selected in every pass and inserted to its appropriate position into the array. It is also the simplest algorithm. It is an in-place comparison sorting algorithm. In this algorithm, the array is divided into two parts, first is sorted part, and another one is the unsorted part. Initially, the sorted part of the array is empty, and unsorted part is the given array. Sorted part is placed at the left, while the unsorted part is placed at the right.

In selection sort, the first smallest element is selected from the unsorted array and placed at the first position. After that second smallest element is selected and placed in the second position. The process continues until the array is entirely sorted.

The average and worst-case complexity of selection sort is $O(n^2)$, where n is the number of items. Due to this, it is not suitable for large data sets.

Selection sort is generally used when -

- A small array is to be sorted
- Swapping cost doesn't matter
- It is compulsory to check all elements

Now, let's see the algorithm of selection sort.

Algorithm

```
SELECTION SORT(arr, n)
```

```
Step 1: Repeat Steps 2 and 3 for i = 0 to n-1
```

```
Step 2: CALL SMALLEST(arr, i, n, pos)
```

```
Step 3: SWAP arr[i] with arr[pos]
```

```
[END OF LOOP]
```

```
Step 4: EXIT
```

```
SMALLEST (arr, i, n, pos)
```

```
Step 1: [INITIALIZE] SET SMALL = arr[i]
Step 2: [INITIALIZE] SET pos = i
Step 3: Repeat for j = i+1 to n
if (SMALL > arr[j])
    SET SMALL = arr[j]
SET pos = j
[END OF if]
[END OF LOOP]
Step 4: RETURN pos
```

Working of Selection sort Algorithm

Now, let's see the working of the Selection sort Algorithm.

To understand the working of the Selection sort algorithm, let's take an unsorted array. It will be easier to understand the Selection sort via an example.

Let the elements of array are -

12	29	25	8	32	17	40
----	----	----	---	----	----	----

Now, for the first position in the sorted array, the entire array is to be scanned sequentially.

At present, **12** is stored at the first position, after searching the entire array, it is found that **8** is the smallest value.

12	29	25	8	32	17	40
----	----	----	---	----	----	----

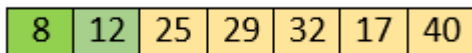
So, swap 12 with 8. After the first iteration, 8 will appear at the first position in the sorted array.

8	29	25	12	32	17	40
---	----	----	----	----	----	----

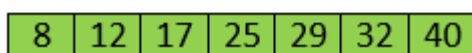
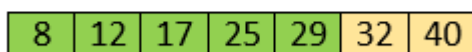
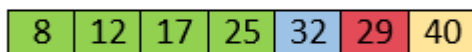
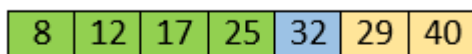
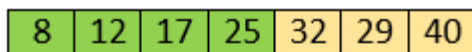
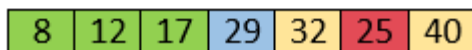
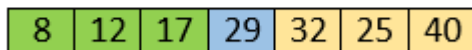
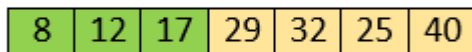
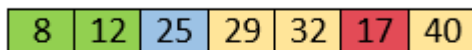
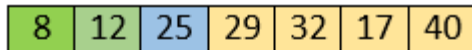
For the second position, where 29 is stored presently, we again sequentially scan the rest of the items of unsorted array. After scanning, we find that 12 is the second lowest element in the array that should be appeared at second position.

8	29	25	12	32	17	40
---	----	----	----	----	----	----

Now, swap 29 with 12. After the second iteration, 12 will appear at the second position in the sorted array. So, after two iterations, the two smallest values are placed at the beginning in a sorted way.



The same process is applied to the rest of the array elements. Now, we are showing a pictorial representation of the entire sorting process.



Now, the array is completely sorted.

Selection sort complexity

Now, let's see the time complexity of selection sort in best case, average case, and in worst case. We will also see the space complexity of the selection sort.

1. Time Complexity

Case Time Complexity

Best Case

$$O(n^2)$$

Average Case

$O(n^2)$

Worst Case

$O(n^2)$

- **Best Case Complexity** - It occurs when there is no sorting required, i.e. the array is already sorted. The best-case time complexity of selection sort is **$O(n^2)$** .
- **Average Case Complexity** - It occurs when the array elements are in jumbled order that is not properly ascending and not properly descending. The average case time complexity of selection sort is **$O(n^2)$** .
- **Worst Case Complexity** - It occurs when the array elements are required to be sorted in reverse order. That means suppose you have to sort the array elements in ascending order, but its elements are in descending order. The worst-case time complexity of selection sort is **$O(n^2)$** .

2. Space Complexity

Space Complexity

$O(1)$

Stable

YES

- The space complexity of selection sort is $O(1)$. It is because, in selection sort, an extra variable is required for swapping.

Implementation of selection sort

Now, let's see the programs of selection sort in different programming languages.

Program: Write a program to implement selection sort in C language.

```
#include <stdio.h>

void selection(int arr[], int n)
{
    int i, j, small;

    for (i = 0; i < n-1; i++) // One by one move boundary of unsorted subarray
```

```
{
    small = i; //minimum element in unsorted array

    for (j = i+1; j < n; j++)
        if (arr[j] < arr[small])
            small = j;
    // Swap the minimum element with the first element
    int temp = arr[small];
    arr[small] = arr[i];
    arr[i] = temp;
}

void printArr(int a[], int n) /* function to print the array */
{
    int i;
    for (i = 0; i < n; i++)
        printf("%d ", a[i]);
}

int main()
{
    int a[] = { 12, 31, 25, 8, 32, 17 };
    int n = sizeof(a) / sizeof(a[0]);
    printf("Before sorting array elements are - \n");
    printArr(a, n);
    selection(a, n);
    printf("\nAfter sorting array elements are - \n");
    printArr(a, n);
    return 0;
}
```

Output:

After the execution of above code, the output will be -

```
Before sorting array elements are -  
12 31 25 8 32 17  
After sorting array elements are -  
8 12 17 25 31 32
```

Program: Write a program to implement selection sort in C++ language.

```
#include <iostream>

using namespace std;

void selection(int arr[], int n)
{
    int i, j, small;

    for (i = 0; i < n-1; i++) // One by one move boundary of unsorted subarray
    {
        small = i; //minimum element in unsorted array

        for (j = i+1; j < n; j++)
            if (arr[j] < arr[small])
                small = j;

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

void printArr(int a[], int n) /* function to print the array */
{
    int i;
    for (i = 0; i < n; i++)
        cout << a[i] << " ";
}

int main()
```

```
{  
    int a[] = { 80, 10, 29, 11, 8, 30, 15 };  
    int n = sizeof(a) / sizeof(a[0]);  
    cout<< "Before sorting array elements are - "<<endl;  
    printArr(a, n);  
    selection(a, n);  
    cout<< "\nAfter sorting array elements are - "<<endl;  
    printArr(a, n);  
  
    return 0;  
}
```

Output:

After the execution of above code, the output will be -

```
Before sorting array elements are -  
80 10 29 11 8 30 15  
After sorting array elements are -  
8 10 11 15 29 30 80
```

Program: Write a program to implement selection sort in C# language.

```
using System;  
class Selection {  
    static void selection(int[] arr)  
    {  
        int i, j, small;  
        int n = arr.Length;  
        for (i = 0; i < n-1; i++) // One by one move boundary of unsorted subarray  
        {  
            small = i; //minimum element in unsorted array  
  
            for (j = i+1; j < n; j++)  
                if (arr[j] < arr[small])  
                    small = j;  
  
            // Swap the minimum element with the first element  
            int temp = arr[small];  
            arr[small] = arr[i];  
            arr[i] = temp;  
        }  
    }  
}
```



```
arr[i] = temp;
}
}

static void printArr(int[] a) /* function to print the array */
{
    int i;
    int n = a.Length;
    for (i = 0; i < n; i++)
        Console.Write(a[i] + " ");
}

static void Main() {
    int[] a = { 85, 50, 29, 18, 7, 30, 3};
    Console.WriteLine("Before sorting array elements are - ");
    printArr(a);
    selection(a);
    Console.WriteLine("\nAfter sorting array elements are - ");
    printArr(a);
}
}
```

Output:

```
Before sorting array elements are - 85 50 29 18 7 30 3
After sorting array elements are - 3 7 18 29 30 50 85
```

Program: Write a program to implement selection sort in python.

```
def selection(a): # Function to implement selection sort
    for i in range(len(a)): # Traverse through all array elements
        small = i # minimum element in unsorted array
        for j in range(i+1, len(a)):
            if a[small] > a[j]:
                small = j
        # Swap the found minimum element with
        # the first element
```

```
a[i], a[small] = a[small], a[i]

def printArr(a): # function to print the array

    for i in range(len(a)):
        print (a[i], end = " ")

a = [69, 14, 1, 50, 59]
print("Before sorting array elements are - ")
printArr(a)
selection(a)
print("\nAfter sorting array elements are - ")
selection(a)
printArr(a)
```

Output:

```
Before sorting array elements are -
69 14 1 50 59
After sorting array elements are -
1 14 50 59 69
```

Program: Write a program to implement selection sort in Java.

```
public class Selection
{
    void selection(int a[]) /* function to sort an array with selection sort */
    {
        int i, j, small;
        int n = a.length;
        for (i = 0; i < n-1; i++)
        {
            small = i; //minimum element in unsorted array

            for (j = i+1; j < n; j++)
                if (a[j] < a[small])
```

```

        small = j;
// Swap the minimum element with the first element
    int temp = a[small];
    a[small] = a[i];
    a[i] = temp;
}

}

void printArr(int a[]) /* function to print the array */
{
    int i;
    int n = a.length;
    for (i = 0; i < n; i++)
        System.out.print(a[i] + " ");
}

public static void main(String[] args) {
    int a[] = { 91, 49, 4, 19, 10, 21 };
    Selection i1 = new Selection();
    System.out.println("\nBefore sorting array elements are - ");
    i1.printArr(a);
    i1.selection(a);
    System.out.println("\nAfter sorting array elements are - ");
    i1.printArr(a);
    System.out.println();
}
}

```

Output:

```

D:\JTP>javac Selection.java
D:\JTP>java Selection
Before sorting array elements are -
91 49 4 19 10 21
After sorting array elements are -
4 10 19 21 49 91

```

Program: Write a program to implement selection sort in PHP.

<?php

```
function selection(&$a, $n) /* function to sort an array with selection sort */
{
    for ($i = 0; $i < $n; $i++)
    {
        $small = $i; //minimum element in unsorted array

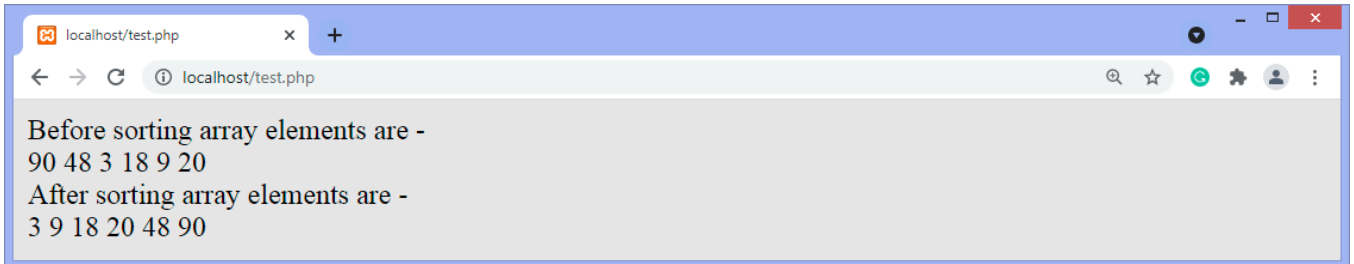
        for ($j = $i+1; $j < $n; $j++)
            if ($a[$j] < $a[$small])
                $small = $j;
        // Swap the minimum element with the first element
        $temp = $a[$small];
        $a[$small] = $a[$i];
        $a[$i] = $temp;
    }
}

function printArray($a, $n)
{
    for($i = 0; $i < $n; $i++)
    {
        print_r($a[$i]);
        echo " ";
    }
}

$a = array( 90, 48, 3, 18, 9, 20 );
$n = count($a);
echo "Before sorting array elements are - <br>";
printArray($a, $n);
selection($a, $n);
echo "<br> After sorting array elements are - <br>";
printArray($a, $n);
?>
```

Output:

After the execution of above code, the output will be -



```
Before sorting array elements are -  
90 48 3 18 9 20  
After sorting array elements are -  
3 9 18 20 48 90
```

So, that's all about the article. Hope the article will be helpful and informative to you.

This article was not only limited to the algorithm. We have also discussed the Selection sort complexity, working, and implementation in different programming languages.

[< Prev](#)[Next >](#)

 [For Videos Join Our Youtube Channel: Join Now](#)


Feedback


- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share





Learn Latest Tutorials


 [Splunk tutorial](#)
Splunk


 [SPSS tutorial](#)
SPSS


 [Swagger tutorial](#)
Swagger








 [T-SQL tutorial](#)
Transact-SQL

 [Tumblr tutorial](#)
Tumblr






 [React tutorial](#)
ReactJS

 [Regex tutorial](#)
Regex








 [Reinforcement learning tutorial](#)

			Reinforcement Learning
 R Programming tutorial R Programming	 RxJS tutorial RxJS	 React Native tutorial React Native	 Python Design Patterns Python Design Patterns
 Python Pillow tutorial Python Pillow	 Python Turtle tutorial Python Turtle	 Keras tutorial Keras	

Preparation

 Aptitude Aptitude	 Logical Reasoning Reasoning	 Verbal Ability Verbal Ability	 Interview Questions Interview Questions
 Company Interview Questions Company Questions			

Trending Technologies

 Artificial Intelligence Tutorial Artificial Intelligence	 AWS Tutorial AWS	 Selenium tutorial Selenium	 Cloud Computing tutorial Cloud Computing
 Hadoop tutorial Hadoop	 ReactJS Tutorial ReactJS	 Data Science Tutorial Data Science	 Angular 7 Tutorial Angular 7



Blockchain
Tutorial

Blockchain



Git Tutorial
Git



Machine
Learning Tutorial
Machine Learning



DevOps
Tutorial
DevOps

B.Tech / MCA



DBMS tutorial
DBMS



Data Structures
tutorial
Data Structures



DAA tutorial
DAA



Operating
System tutorial
Operating System



Computer
Network tutorial
Computer Network



Compiler
Design tutorial
Compiler Design



Computer
Organization and
Architecture
Computer
Organization



Discrete
Mathematics
Tutorial
Discrete
Mathematics



Ethical Hacking
Tutorial
Ethical Hacking



Computer
Graphics Tutorial
Computer Graphics



Software
Engineering
Tutorial
Software
Engineering



html tutorial
Web Technology



Cyber Security
tutorial
Cyber Security



Automata
Tutorial
Automata



C Language
tutorial
C Programming



C++ tutorial
C++



Java tutorial
Java



.Net
Framework
tutorial
.Net



Python tutorial
Python



List of
Programs
Programs



Control
Systems tutorial
Control System



Data Mining
Tutorial
Data Mining



Data
Warehouse
Tutorial
Data Warehouse

