Tutorial Link https://codequotient.com/tutorials/Selection Sort/5a12e94346765b2b63e34754

TUTORIAL

Selection Sort

Chapter

Selection Sort is an in-place comparison based sorting algorithm.

Idea:

In an array that is sorted in ascending order, the first element is the minimum of all the n array elements, similarly the second element is the minimum of remaining n-1 elements, the third element is the minimum of remaining n-2 elements, and so on.

Algorithm - Considering ascending order

Selection Sort algorithm works on the above idea.

- Firstly, find the minimum element in the given array(*arr[0...n-1]*) and swap it with the element at the 0th index.
- Then, in the remaining n-1 elements(*arr[0...n-2]*) again find the minimum element and swap it with the 1st index element.
- After the **i**th iteration all the array elements from index 0 to i will be sorted, therefore we will continue the above process, to fill the remaining array positions with the correct element.

Let's visualise the algorithm with the following example:

```
arr[ ] = {6, 3, 8, 9, 5}
```

// Find the minimum element in arr[0...4] and swap it with the 0th index element

{**3**, 6, 8, 9, 5}

// Find the minimum element in arr[1...4] and swap it with the 1st index element

{3, **5**, 8, 9, 6}

// Find the minimum element in arr[2...4] and swap it with the 2nd index element

{3, 5, **6**, 9, 8}

// Find the minimum element in arr[3...4] and swap it with the 3rd index element

{3, 5, 6, **8**, 9}

We can notice that after the algorithm ends, the given array is sorted in ascending order.

Pseudo Code

```
for i := 0 to n-2
   select the smallest among A[i], . . . , A[n-1]
   swap it with A[i];
end
```

Javascript

```javascript
function selectionSort(array) {
    let n = array.length;

    for(let i = 0; i < n; i++) {
        let min_index = i;
        for(let j = i+1; j < n; j++){
            if(array[j] < array[min_index]) {
                min_index=j;
            }
        }
        let tmp = array[i];
        array[i] = array[min_index];
```

```
        array[min_index] = tmp;
    }
}


function main(){
    let arr=[6, 3, 8, 9, 5]

    console.log('Given Array:',arr.join(' '))
    selectionSort(arr)
    console.log('Sorted Array:',arr.join(' '))
}

main()
```

C

```
#include <stdio.h>

void swap(int *a, int *b)
{
   int temp = *a;
   *a = *b;
   *b = temp;
}

void selectionSort(int arr[], int n)
{
   int min_index;

   for (int i = 0; i < n-1; i++)
   {
       min_index = i;
       for (int j = i+1; j < n; j++)
       {
           if (arr[j] < arr[min_index])
               min_index = j;
       }
       swap(&arr[min_index], &arr[i]);
   }
}

```

```
void printArray(int arr[], int n)
{
   for (int i = 0; i < n; i++)
       printf("%d ", arr[i]);
   printf("\n");
}

int main()
{
   int arr[] = {6, 3, 8, 9, 5};
   int n = sizeof(arr)/sizeof(arr[0]);

   printf("Given Array: ");
   printArray(arr, n);

   selectionSort(arr, n);

   printf("Sorted Array: ");
   printArray(arr, n);

   return 0;
}

```

Java

```
public class Main
{
   static void selectionSort(int arr[], int n)
   {
       int min_index;

       for (int i = 0; i < n-1; i++)
       {
           min_index = i;
           for (int j = i+1; j < n; j++)
           {
               if (arr[j] < arr[min_index])
                   min_index = j;
           }

           int temp = arr[min_index];
```

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

    static void printArray(int arr[], int n)
    {
        for (int i = 0; i < n; i++)
            System.out.print(arr[i] + " ");
        System.out.println();
    }

    public static void main(String args[])
    {
        int arr[] = {6, 3, 8, 9, 5};
        int n = arr.length;

        System.out.print("Given Array: ");
        printArray(arr, n);

        selectionSort(arr, n);

        System.out.print("Sorted Array: ");
        printArray(arr, n);
    }
}

```

Python 3

```
def selectionSort(array):
    for i in range(len(array)):
        min_index = i
        for j in range(i+1, len(array)):
            if array[min_index] > array[j]:
                min_index = j
        array[i], array[min_index] = array[min_index], array[i]

if __name__=='__main__':
    arr=[6, 3, 8, 9, 5]
    print('Given Array',' '.join( str(x) for x in arr ))
    selectionSort(arr);
```

```
    print('Sorted Array',' '.join( str(x) for x in arr
))
```

C++

```cpp
#include <iostream>
using namespace std;

void swap(int *a, int *b)
{
   int temp = *a;
   *a = *b;
   *b = temp;
}

void selectionSort(int arr[], int n)
{
   int min_index;

   for (int i = 0; i < n-1; i++)
   {
       min_index = i;
       for (int j = i+1; j < n; j++)
       {
           if (arr[j] < arr[min_index])
               min_index = j;
       }
       swap(&arr[min_index], &arr[i]);
   }
}

void printArray(int arr[], int n)
{
   for (int i = 0; i < n; i++)
       cout << arr[i] << " ";
   cout << "\n";
}

int main()
{
   int arr[] = {6, 3, 8, 9, 5};
   int n = sizeof(arr)/sizeof(arr[0]);

```

```
39      cout << "Given Array: ";
40      printArray(arr, n);
41
42      selectionSort(arr, n);
43
44      cout << "Sorted Array: ";
45      printArray(arr, n);
46
47      return 0;
48  }
49
```

Properties of Selection Sort

Time Complexity: O(n^2) *; There is no best or worst case time complexity for selection sort, because in any case it will search the minimum element linearly for all n-1 traversals.*

Space Complexity: O(1)

In-Place Sorting Algorithm: Yes

Stable Sorting Algorithm: Depends on implementation

Advantages

- The total number of swaps in the selection sort algorithm will never exceed O(n). Therefore it will be useful in the problems where we want to minimize the number of swaps while sorting.
- It is an In-Place sorting algorithm that does not require any extra space.

Video Solution

<iframe width="560" height="315" src="https://www.youtube.com/embed/Xb6Y-oP6nPs" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>