Selection Sort Program in C

codewithharry.com/videos/data-structures-and-algorithms-in-hindi-55

In the last lecture, we learned what selection sort is and how it is used to sort an array of elements by selecting the smallest of all from the unsorted part and replacing it with its final position. Finally, we enlisted our conclusions made after analyzing the algorithm using our defined criteria. Before we move on to the programming part, let's review these characteristics of the selection sort algorithm.

1. The time complexity of the selection sort algorithm is $O(n^2)$ in all its cases.
2. It is not a stable algorithm since it fails to preserve the original order of equal elements. We saw one example the other day.
3. It is not a recursive algorithm.
4. Selection sort is not an adaptive algorithm. It anyways makes comparisons regardless of whether the array given is sorted or not.

That being all that was known about the selection sort algorithm, let us move on to the programming part. I have attached the source code below. Follow it as we proceed.

Understanding the code snippet below:

1. First few steps remain the same as we did for the previous two algorithms. The first step is to define an array of elements. We define an array of integers.

2. Define an integer variable for storing the size/length of the array.

3. Before we proceed to write the function for Selection Sort, we would first make a function for displaying the array's content.

4. Since we did that already in previous lectures, we would just copy the function *printArray* from the previous programming lecture.

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

Code Snippet 1: Creating the *printArray* function

5. Create a void function *selectionSort* and pass the array's address and the array's length as its parameters. Create two integer variables, one for maintaining the *min* index, called the *indexOfMin,* and another for swapping purposes called the Now;

create a *for* loop that tracks the number of passes. If you recall, to sort an array of length 5 using the selection sort algorithm, we made a total of 4 passes. So, for an array of length n, we would make (n-1) passes. And the loop starts from the 0^{th} index and ends at $(n-1)^{th}$.

And if you remember, at each pass, we first initialize the *indexOfMin* to be the first index of the unsorted part. So, inside this loop, initialize the *indexOfMin* to be *i,* which is always the first index of the unsorted part of the array.

Create another loop to iterate over the rest of the elements in the unsorted part to find if there is any lesser element than the one at *indexOfMin*. Make this loop run from *i+1* to the last. And compare the elements at every index.If you find an element at index *j,* which is less than the element at *indexOfMin,* then update *indexOfMin* to *j*.

And finally, when you finish iterating through the second loop, just swap the elements at indices *i & indexOfMin*. Swap using the *temp* variable. Follow the same steps at each pass.

And at the end, when you finish iterating through both the *i* and thej loops, you would receive a sorted array. All we had to do was this.

```
void selectionSort(int *A, int n){
    int indexOfMin, temp;
    printf("Running Selection sort...\n");
    for (int i = 0; i < n-1; i++)
    {
        indexOfMin = i;
        for (int j = i+1; j < n; j++)
        {
            if(A[j] < A[indexOfMin]){
                indexOfMin = j;
            }
        }
        // Swap A[i] and A[indexOfMin]
        temp = A[i];
        A[i] = A[indexOfMin];
        A[indexOfMin] = temp;
    }
}
```

Code Snippet 2: Creating the *selectionSort* function

Here is the whole source code:

```c
#include<stdio.h>

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

void selectionSort(int *A, int n){
    int indexOfMin, temp;
    printf("Running Selection sort...\n");
    for (int i = 0; i < n-1; i++)
    {
        indexOfMin = i;
        for (int j = i+1; j < n; j++)
        {
            if(A[j] < A[indexOfMin]){
                indexOfMin = j;
            }
        }
        // Swap A[i] and A[indexOfMin]
        temp = A[i];
        A[i] = A[indexOfMin];
        A[indexOfMin] = temp;
    }
}

int main(){
    // Input Array (There will be total n-1 passes. 5-1 = 4 in this case!)
    //  00  01  02  03  04
    // |03, 05, 02, 13, 12

    // After first pass
    //  00  01  02  03  04
    //  02,|05, 03, 13, 12

    // After second pass
    // 00  01  02  03  04
    // 02, 03,|05, 13, 12

    // After third pass
    // 00  01  02  03  04
    // 02, 03, 05,|13, 12

    // After fourth pass
    // 00  01  02  03  04
    // 02, 03, 05, 12,|13


    int A[] = {3, 5, 2, 13, 12};
    int n = 5;
    printArray(A, n);
    selectionSort(A, n);
    printArray(A, n);

    return 0;
}
```

Code Snippet 3: Program to implement the Selection Sort Algorithm

Let us now check if our functions work well. Consider an array A of length 5.

```
int A[] = {3, 5, 2, 13, 12};
int n = 5;
printArray(A, n);
selectionSort(A, n);
printArray(A, n);
```

Code Snippet 4: Using the *selectionSort* function

And the output we received was:

```
3 5 2 13 12
Running Selection sort...
2 3 5 12 13
PS D:\MyData\Business\code playground\Ds & Algo with Notes\Code>
```

Figure 1: Output of the above program

Visualize each pass individually using the dry run method. This would give you a lot more confidence. The dry run of the above example is there in the source code itself.

And that was all to be done to implement the selection sort algorithm. I hope you practice these algorithms taught here yourself. Let me know if you do. And if you don't, it's never too late. Gear yourselves up. And start today.

I appreciate your support throughout. I hope you enjoyed the tutorial. If you genuinely appreciate my work, please let your friends know about this course too. If you haven't checked out the whole playlist yet, move on to codewithharry.com or my YouTube channel to access it. See you all in the next tutorial, where we'll learn another sorting algorithm called the **Quicksort Algorithm**. Till then, keep coding.