Selection sort algorithm

Source: ***https://github.com/rusevrosen/codehub.github.io/tree/main/Algorithms/Sorting_Algorithms***

1. *What is Selection Sort algorithm?*

Selection sort is an in-place comparison sorting algorithm. This sorting algorithm is known for its simplicity and memory efficiency — it doesn't take up any extra space. The selection sort method repeatedly searches "remaining items" to find the least one and moves it to its final location.

This algorithm divides the input array into two subparts — the sorted part and the unsorted part. Initially, the sorted part of the array is empty, and the unsorted part is the input array.

The algorithm works on the principle of finding the lowest number from the unsorted part of the array and then swapping it with the first element of the unsorted part. This is done over and over until the entire array becomes sorted (in ascending order).

2. *Code example.*

```cpp
#include <iostream>

void selectionSort(int arr[], int n) {
  for (int i = 0; i < n - 1; i++) {
    int minIndex = i;
    for (int j = i + 1; j < n; j++) {
      if (arr[j] < arr[minIndex]) {
        minIndex = j;
      }
    }
    std::swap(arr[i], arr[minIndex]);
  }
}

int main() {
  int arr[] = {3, 5, 1, 4, 2};
  int n = sizeof(arr) / sizeof(arr[0]);

  std::cout << "Original array: ";
  for (int i = 0; i < n; i++) {
    std::cout << arr[i] << " ";
  }
  std::cout << std::endl;

  selectionSort(arr, n);

  std::cout << "Sorted array: ";
  for (int i = 0; i < n; i++) {
    std::cout << arr[i] << " ";
  }
  std::cout << std::endl;

  return 0;
}
```

The output of the example is:

```
Original array: 3 5 1 4 2
Sorted array: 1 2 3 4 5
```

3. *Code explanation.*

First, we define the **`selectionSort`** function, which takes an array **`arr`** and its size n as arguments. The function uses a nested loop to find the minimum element in the unsorted part of the array and places it at the beginning of the unsorted part by swapping it with the element at the current index.

Then, in main, we create an array **`arr`** of integers and assign it the values **`{3, 5, 1, 4, 2}`**. We also calculate the size of the array and store it in the variable **`n`**.

Next, we print out the original array using a loop.

Then, we call the **`selectionSort`** function, passing it the array and its size as arguments. This sorts the array in ascending order.

Finally, we print out the sorted array using another loop.

The program outputs the original array, then the sorted array.

4. *Time complexity.*

The time complexity of the selection sort algorithm implemented in the example above is O(n^2).

This is because the algorithm uses two nested loops. The outer loop iterates over all the elements in the array, and the inner loop iterates over the remaining elements to find the minimum element.

The time complexity of an algorithm describes how the running time of the algorithm grows as the input size grows. In this case, the input size is the number of elements in the array.

The time complexity of the selection sort is O(n^2) in the worst case, average case, and best case. This means that, as the input size increases, the running time of the algorithm grows at a rate of n^2.

For example, if the input size is 10, it takes 100 steps to complete the algorithm. If the input size is 100, it takes 10000 steps to complete the algorithm.

I wish you happy learning,

Your Rosen Rusev