Bubble sort algorithm

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

1. What is the Bubble Sort algorithm?

Bubble sort is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares adjacent elements, and swaps them if they are in the wrong order. The pass of the list is repeated until the list is sorted.

This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

2. *Code example.*

```
#include <iostream>
#include <algorithm>

using namespace std;

int main()
{
  int a[5] = {5, 4, 3, 2, 1};
  int n = sizeof(a)/sizeof(a[0]);
  bool swapped;
  do
  {
    swapped = false;
    for (int i = 0; i < n - 1; i++)
    {
      if (a[i] > a[i + 1])
      {
        swap(a[i], a[i + 1]);
        swapped = true;
      }
    }
  } while (swapped);
  for (int i = 0; i < n; i++)
  {
    cout << a[i] << " ";
  }
  return 0;
}
```

This program will output the sorted array: 1 2 3 4 5.

3. Code explanation.

The first thing the program does is define an array of integers called a, and set its values to **`{5, 4, 3, 2, 1}.`** Then it calculates the length of the array and stores it in the variable **`n`**.

Next, the program declares a **`boolean`** variable called swapped, which will be used to track whether any swaps were made during the sorting process.

Then, the program enters a **`do-while`** loop. This loop will continue to run as long as the swapped variable is true. The first time the loop runs, swapped is initialized to false, so the loop will only run once if no swaps are made.

Inside the loop, the program runs a **`for loop`** that iterates through all the elements of the array, except for the last element. For each iteration, it compares the current element with the element following it. If the current element is greater than the element following it, the two elements are swapped using the **`swap()`** function from the algorithm library. The swapped variable is also set to true, indicating that a swap was made.

After the inner for loop finishes, the **`do-while`** loop checks the value of the swap. If it is true, the loop will run again. If it is false, the loop will terminate and the program will move on to the next block of code.

The final block of code simply prints out the sorted array to the console.

4. Time complexity.

The time complexity of the bubble sort algorithm in the example above is **`O(n^2).`**

Bubble sort has a time complexity of **`O(n^2)`** because it requires two nested loops to sort an array of n elements. The inner loop iterates through the array and compares adjacent elements, while the outer loop repeats this process until the array is sorted.

In each pass through the array, the bubble sort algorithm moves the largest element to the end of the array, so the inner loop needs to make one fewer comparison on each pass. This means that the time complexity of bubble sort is not quite **`O(n^2)`** in the worst case, but it is still generally considered to be **`O(n^2)`** because the difference in the number of comparisons is relatively small compared to the size of the input.

.

I wish you happy learning,

Your Rosen Rusev