Part 12: Sorting Algorithms

Dr. Andi Toce
Lecture Notes for MAC 101 (Introduction to Computer Science)

Last updated / viewed: November 26, 14

Table of Contents

1. Sorting Algorithms – The Bubble Sort

Very often we need to arrange (sort) data for searching or other purposes. There is a large variety of sorting algorithms. We will only discuss here the simplest algorithms.

Bubble Sort

BubbleSort.cpp

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

const int arraySize = 10;
int comparisonCount = 0;
int swapCount = 0;
int myArray[arraySize];
void BubbleSort();
void swap(int one, int two);

int main(){

        cout<<"Enter " << arraySize << " integers: "<<endl;

        // Input values to array
        for(int i=0; i<arraySize; i++)
                cin >> myArray[i];
    cout << endl;

    cout<<"Array values before sorting: ";
    for(int j=0; j<arraySize; j++)
       cout << myArray[j] << " ";
    cout<<endl;

    BubbleSort();

    cout<<"Array values after sorting: ";
    for(int j=0; j<arraySize; j++)
        cout << myArray[j] << " ";
    cout << endl << endl;
    cout << "Comparisons used: " << comparisonCount << endl;
    cout << "Swaps used: " << swapCount << endl;

    return 0;
} // end main

void BubbleSort(){

        for(int i=0; i<arraySize; i++)
                for(int j=0; j<arraySize-1; j++){
                        comparisonCount++;
                        if(myArray[j] > myArray[j+1])
                                swap(j, j+1);
                }
} // end BubbleSort

void swap(int one, int two){
        swapCount++;
        int temp = myArray[one];
        myArray[one] = myArray[two];
        myArray[two] = temp;
}// end swap
```

Output

```
Enter 10 integers:
6
5
4
8
9
2
1
4
5
6

Array values before sorting: 6 5 4 8 9 2 1 4 5 6
Array values after sorting: 1 2 4 4 5 5 6 6 8 9

Comparisons used: 90
Swaps used: 23
```

Question: Can we improve the above algorithms to reduce the number of operations performed?

2. Selection Sort

SelectionSort.cpp

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

const int arraySize = 10;
int comparisonCount = 0;
int swapCount = 0;
int myArray[arraySize];
void SelectionSort();
void swap(int one, int two);

int main(){

        cout<<"Enter " << arraySize << " integers: "<<endl;

        // Input values to array
        for(int i=0; i<arraySize; i++)
                cin >> myArray[i];
    cout << endl;

    cout<<"Array values before sorting: ";
    for(int j=0; j<arraySize; j++)
       cout << myArray[j] << " ";
    cout<<endl;

    SelectionSort();

    cout<<"Array values after sorting: ";
    for(int j=0; j<arraySize; j++)
        cout << myArray[j] << " ";
    cout << endl << endl;
    cout << "Comparisons used: " << comparisonCount << endl;
    cout << "Swaps used: " << swapCount << endl;

    return 0;
} // end main

void SelectionSort(){

        int min;
        for(int i=0; i<arraySize-1; i++){
                min=i;
                for(int j=i+1; j<arraySize; j++){
                        comparisonCount++;
                        if(myArray[j] < myArray[min])
                                min=j;
                }

                swap(i, min);
        }
} // end SelectionSort

void swap(int one, int two){
        swapCount++;
        int temp = myArray[one];
        myArray[one] = myArray[two];
        myArray[two] = temp;
}// end swap
```

Output

```
Enter 10 integers:
6
5
4
2
1
65
4
6
1
32

Array values before sorting: 6 5 4 2 1 65 4 6 1 32
Array values after sorting: 1 1 2 4 4 5 6 6 32 65

Comparisons used: 45
Swaps used: 9
```

3. Insertion Sort

InsertionSort.cpp

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

const int arraySize = 10;
int comparisonCount = 0;
int myArray[arraySize];
void InsertionSort();

int main(){

        cout<<"Enter " << arraySize << " integers: "<<endl;

        // Input values to array
        for(int i=0; i<arraySize; i++)
                cin >> myArray[i];
    cout << endl;

    cout<<"Array values before sorting: ";
    for(int j=0; j<arraySize; j++)
       cout << myArray[j] << " ";
    cout<<endl;

    InsertionSort();

    cout<<"Array values after sorting: ";
    for(int j=0; j<arraySize; j++)
        cout << myArray[j] << " ";

    return 0;
} // end main

void InsertionSort(){

        int in, out;

        for(out=1; out<arraySize; out++){
                int temp=myArray[out];
                in = out;
                while (in>0 && myArray[in-1]>=temp){
                        myArray[in] = myArray[in-1];
                        in--;
                }
                myArray[in]=temp;
        } // end for
} // end InsertionSort
```

Output

```
Enter 10 integers:
5
5
1
63
9
7
1
3
1
6

Array values before sorting: 5 5 1 63 9 7 1 3 1 6
Array values after sorting: 1 1 1 3 5 5 6 7 9 63
```

Try now: Add a counter for the number of comparisons and the number of shifts. Print the value of each counter at the end of the output.

Question: Which of the three sorting algorithms performs better and under what circumstances?