Lab 8: Implementation of Elementary Sorting Algorithms

Ali Hassan 03-135211-005

**Objectives**

**Understanding of:**

* Implementation of Insertion Sort
* Implementation of Selection Sort
* Implementation of Bubble Sort

**Tools Required**

1. PC with Windows 7 Professional
2. Visual Studio 2010

**Sorting:**

As soon as you create a significant collection of data, you will probably think of reasons to sort it. You need to arrange names in alphabetical order, students by grade, customers by zip code, house sales by price, cities in order of increasing population, countries by GNP, stars by magnitude, and so on.

Sorting data may also be a preliminary step to searching it. As we saw in the previous labs of a binary search, which can be applied only to sorted data, is much faster than a linear search. Because sorting is so important and potentially so time-consuming, it has been the subject of extensive research in computer science, and some very sophisticated methods have been developed. However, in this lab we will look at perhaps the simplest algorithm: the Bubble sort, Selection Sort and Insertion sort.

**Bubble Sort (Pseudocode):**

begin BubbleSort(list)

for all elements of list if list[i] > list[i+1] swap(list[i], list[i+1]) end if end for return list

end BubbleSort

**Selection Sort (Pseudocode):**

begin SelectionSort list : array of items n : size of list for i = 1 to n - 1

/\* set current element as minimum\*/ min = i

/\* check the element to be minimum \*/ for j = i+1 to n if list[j] < list[min] then min = j; end if end for

/\* swap the minimum element with the current element\*/ if indexMin != i then swap list[min] and list[i] end if end for

end SelectionSort

**Insertion Sort:**

The insertion sort is substantially better than the bubble sort and selection sort. It still executes in

O(N2) time, but it’s about twice as fast as the bubble sort. It’s also not too complex, although it’s slightly more involved than the bubble sort. It’s often used as the final stage of more sophisticated sorts, such as quick sort.

**(Pseudocode)**

begin insertionSort( A : array of items ) int holePosition int valueToInsert for i = 1 to length(A) inclusive do: /\* select value to be inserted \*/ valueToInsert = A[i] holePosition = i

/\*locate hole position for the element to be inserted \*/

while holePosition > 0 and A[holePosition-1] > valueToInsert do:

A[holePosition] = A[holePosition-1] holePosition = holePosition -1 end while

/\* insert the number at hole position \*/ A[holePosition] = valueToInsert end for end insertionSort

# TASKS

**Task #.1: Estimated Time: 30 Mins.**

Write a program in C++ to sort an array of ‘n’ elements using Selection Sort Algorithm in descending order.

Code:

#include <iostream>

using namespace std;

void

swap (int \*xp, int \*yp)

{

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

void

selectionSort (int arr[], int n)

{

int i, j, min\_idx;

for (i = 0; i < n - 1; i++)

{

min\_idx = i;

for (j = i + 1; j < n; j++)

if (arr[j] > arr[min\_idx])

min\_idx = j;

swap (&arr[min\_idx], &arr[i]);

}

}

void

printArray (int arr[], int size)

{

int i;

for (i = 0; i < size; i++)

cout << arr[i] << " ";

cout << endl;

}

int

main ()

{

int arr[] = { 34, 25, 182, 22, 41 };

int n = sizeof (arr) / sizeof (arr[0]);

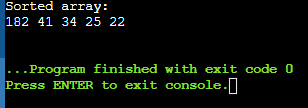
selectionSort (arr, n);

cout << "Sorted array: \n";

printArray (arr, n);

return 0;

}



**Task #.2: Estimated Time: 30 Mins.**

Write a program in C++ to count number of swaps is Selection Sort.

Code:

#include<iostream>

using namespace std;

void SelectionSort (int[], int);

void Swap (int &, int &);

void Display (int[], int);

int

main ()

{

cout << endl;

int array[15];

int size = 15;

cout << "The Array: ";

for (int i = 0; i < size; i++)

array[i] = random () % 15;

Display (array, size);

cout << endl;

cout << "Selection Sort the Array in Descending Array! " << endl;

SelectionSort (array, size);

Display (array, size);

getchar ();

return 0;

}

void

SelectionSort (int arr[], int n)

{

int min = 0;

int count = 0;

for (int i = 0; i < n - 1; i++)

{

min = i;

for (int j = i + 1; j < n; j++)

{

if (arr[j] > arr[min])

min = j;

}

if (min != i)

{

Swap (arr[i], arr[min]);

count++;

}

}

cout << "Number of Swaps: " << count << endl << endl;

}

void

Swap (int &x, int &y)

{

int temp;

temp = x;

x = y;

y = temp;

}

void

Display (int arr[], int n)

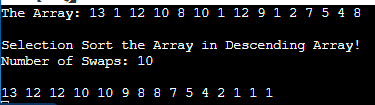
{

for (int i = 0; i < n; i++)

cout << arr[i] << " ";

cout << endl;

}



**Task #.2: Estimated Time: 30 Mins.**

Write a program in C++ to print numbers in descending order using recursion.

Example call: Reverse(5)

Output: 5 4 3 2 1

Code:

#include<iostream>

using namespace std;

void Reverse (int);

int

main ()

{

cout << endl;

int num;

cout << "Enter a Number: ";

cin >> num;

cout << endl;

cout << "Print Reverse Numbers from " << num << endl;

Reverse (num);

cout << endl;

getchar ();

getchar ();

return 0;

}

void

Reverse (int n)

{

if (n > 0)

{

cout << n << " ";

n--;

Reverse (n);

}

}

