Jacob Ciurca

Dr. Lu

Lab 1

Cosc 320

9/2/2020

**PreLab Tasks:** Reviewed the time complexity, design, and implementation of Bubble sort, Selection sort, Insertion sort, Merge sort, Quicksort, and Radix sort

**Lab Task:**

**deSelsort.h:**

#ifndef DESELSORT\_H\_INCLUDED

#define DESELSORT\_H\_INCLUDED

template <class T>

void double\_ended\_selection(T a[], int n){

T min;

T max;

int pass = 1;

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

min = a[i];

max = a[i];

int min\_i = i, max\_i = i;

for (int k = i; k <= j; k++) {

if(a[k] > max) {

max = a[k];

max\_i = k;

} if(a[k] < min) {

min = a[k];

min\_i = k;

}

}

T temp = a[i];

a[i] = a[min\_i];

a[min\_i] = temp;

if(a[i] != a[min\_i]){

std::cout << "Swap " << a[i] << " and " << a[min\_i] << std::endl;

}

if (a[min\_i] == max){

T temp = a[j];

a[j] = a[min\_i];

a[min\_i] = temp;

if(a[i] != a[min\_i]){

std::cout << "Swap " << a[j] << " and " << a[min\_i] << std::endl;

}

}

else{

T temp = a[j];

a[j] = a[max\_i];

a[max\_i] = temp;

if(a[i] != a[max\_i]){

std::cout << "Swap " << a[j] << " and " << a[max\_i] << std::endl;

}

}

std::cout << "Pass " << pass << ": ";

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

std::cout << a[i] << ", ";

}

std::cout << "\n";

pass++;

}

}

#endif // DESELSORT\_H\_INCLUDED

**Lab01.cpp:**

#include <iostream>

#include "deSelsort.h"

using namespace std;

int main()

{

cout << "Please enter the size you would like the array to be.\n";

int s;

cin >> s;

cout << "Please enter the array.\n";

int arr[s];

for(int i=0; i<s; i++){

cin >> arr[i];

}

cout << "Starting Array: ";

for(int i=0; i<s; i++){

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

}

cout << endl;

double\_ended\_selection(arr, s);

cout << "Sorted Array: ";

for(int i=0; i<s; i++){

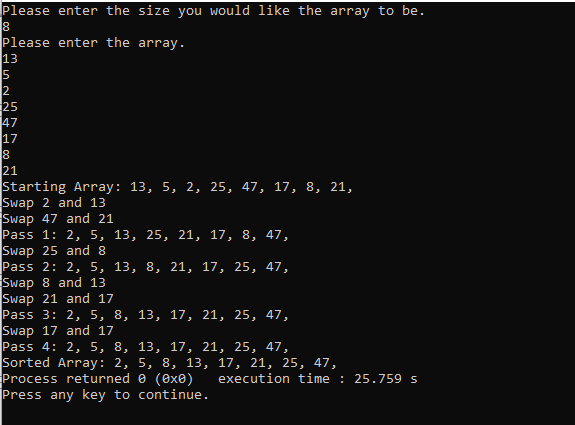
cout << arr[i] << ", ";

}

return 0;

}

**Sample Output:**



**Questions:**

1. If there is an even amount of elements to be sorted then once all the elements have been passed through in the array then the array has been sorted but if there are an odd number of odd elements once there is one element left unsorted this is when min = max and the array has been sorted.
2. The time complexity of double ended selection sort is O(n^2) which is the same as selection sort. Double ended selection sort is still slightly better than regular selection sort because after each pass, the number of elements to be viewed decreases by two instead of one making it ever so slightly faster but in big O notation O(n^2 – n) is still equal to O(n^2) because as n becomes larger n^2 comes to dominate, neglecting the term -n.

**PostLab:** This lab was good at refreshing my memory on the implementation and complexity of sorting algorithms that we previously covered in 220. This lab took me around 1 hour 45 minutes to complete.