CSE331: Data Structures and Algorithms

Insertion Sort Lab Report

Name: Anthony Amgad Fayek

Program: CESS

ID: 19P9880

The Full Project is in a GitHub Repository Below

This project is to test the insertion sort and to compare it with its average running time value that is calculated mathematically using c++.

Here are the used libraries and definitions:

```
#include <iostream>
#include <fstream>
#include <ctime>
#include <cstdlib>
#define LENGTH 10000
using namespace std;
```

We first create the insertion sort function:

```
int insertionSort(int arr[], int len) {
	int step = 0;
	for (int i = 1; i < len; i++) {
		int key = arr[i];
		step++;
		int j = i - 1;
		step++;
		while (j >= 0 && arr[j] > key) {
			arr[j + 1] = arr[j];
			step++;
			j--;
			step++;
		}
		arr[j + 1] = key;
		step++;
	}
	return step;
}
```

The lines that use the variable step are to calculate the actual running time of the insertion sort. We can deduce that since the code consists of 2 nested loops that the average run time will be $\Theta(n^2)$.

We then create a function that generates random numbers into a file:

```
void createRandFile() {
	ofstream mfile("unsortedFile.txt");
	srand(time(0));

	for (int i = 0; i < LENGTH; i++) {
		mfile << ((rand() % LENGTH) + 1) << endl;
	}
}
```

The samples created can be found in the GitHub repository linked below

<u>Afterwards we need a function that reads the generated values:</u>

```
void readFile(int arr[], int l) {
	ifstream mfile("unsortedFile.txt");
	for (int i = 0; i < l; i++) {
		mfile >> arr[i];
	}
}
```

<u>We can also add additional functions to store the sorted values in a different file and test whether the sort was successful or not:</u>

```
void createSortedFile(int arr[]) {
	ofstream mfile("sortedFile.txt");

	for (int i = 0; i < LENGTH; i++) {
		mfile << arr[i] << endl;
	}
}

bool testSortedFile() {
	int arr[LENGTH];
	ifstream mfile("sortedFile.txt");
	for (int i = 0; i < LENGTH; i++) {
		mfile >> arr[i];
	}
	for (int i = 0; i < LENGTH - 1; i++) {
		if (arr[i] > arr[i + 1])
			return false;
	}
	return true;
}
```

The sorted generated file of the samples above can be found in the GitHub repository linked below.

We now want to test the insertion sort and get back the "step" value at different array sizes. To do so we are going to create an additional file in main that has both the number of elements sorted and the sort's "step" value that can be imported in excel to create a comparison.

Thus we can create the main function:

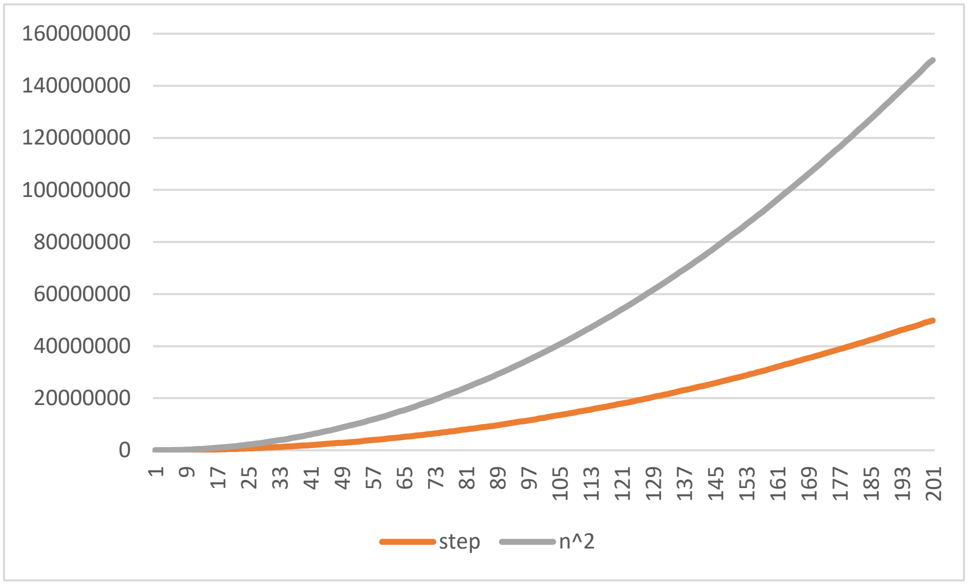
```
int main() {
	int arr[LENGTH];
	createRandFile();
	readFile(arr, LENGTH);
	ofstream sFile("stepFile.txt");
	int x[LENGTH];

	for (int i = 10; i < 10000; i += 50) {
		for (int j = 0; j < i; j++) {
			x[j] = arr[j];
		}
		sFile << i << ',' << insertionSort(x, i) << endl;
	}

	sFile << 10000 << ',' << insertionSort(arr, 10000) << endl;

	createSortedFile(arr);
	if(testSortedFile())
		cout << "Test for sort is succesful" << endl;
	else
		cout << "Test for sort is not succesful" << endl;

	system("pause");
	return 0;
}
```

Here is a sample of the result produced in excel:

n	step	n^2
10	51	100
60	1971	3600
110	6879	12100
160	14389	25600
210	22247	44100
260	34269	67600
310	47711	96100
360	63457	129600
410	80219	168100
460	97435	211600

The full excel file can be found in the GitHub repository linked below.

Graph created using all the samples collected:



GitHub Repository:
https://github.com/Anthony-Amgad/CSE331InsertionSort19P9880