The UNIVERSITY of OKLAHOMA

DSA/CS 5005 Computing Structures – Programming Project 3

Summer 2020

Due August 13th, 2020, 11:59 PM

Discussion

Sorting algorithms are arranging the given data in order. Several sorting techniques are used in almost every application to efficiently handle requests of many kinds. In this project, you will be implementing the ***Bubble Sort and Quick sort techniques***. You will also be measuring the time taken for each of them and plotting them as a line graph to visually compare.

Implementation of the class

The sortData class defined below in this specification will be used. The two fields we have here, which are private, are the number of elements(N) and the elements themselves(elements) which is an array of integers.

Generation of the random numbers for sorting

You are required to generate random numbers that are numElements number long. The numElements are given to you in the input file.
The way you would generate the random numbers is with the seed(srand()) that is given to you in the input file. This seed would make sure the random numbers that you generate are pseudo random. This means that with the same seed, every time you generate the numbers, it will be the same set of random numbers.
The random numbers generated also need to be within the range(between lower and upper) given in the input file.

Input file

The first line in the input file is the number of elements(numElements). The line following that has the seed followed by the lower range and the upper range for random number generation.

Submission

There will be 2 parts to the submission component in this project. To write the algorithms and sort the pseudo randomly generated numbers so that you pass the GradeScope autograding test cases(upload code saved as **project3.cpp**). The second would be to experiment and write a report on the sorting algorithms. More details follow.

Autograder submission:

You will be given a sample output file and you would need to write the algorithms and display the sorted numbers for the given input file.

Report submission:
You will have to plot the time taken for sorting(does not include random generation nor printing – use clock_t) 5000, 10000, 50000, 100000 number of elements for both the algorithms(a graph(time vs numElements)).
You also need to explain the results you get by answering the question of why.

Bonus credit

As a bonus, you may implement the Shell Sort and Adaptive Sort (can be found on page 650 on our Data Structures Book) algorithms as well. You'd have to use the same class along with the new methods for each of these two new algorithms. You'd also have to discuss them in your report along with the Bubble and Quick sort algorithms.
Each of the extra credit sorting algorithms will be worth 3% of the project grade each. So, if both are implemented you can get up to 6% extra credit.

Final submission requirement

You would need to submit 3 files at the end.

1. project3.cpp – for autograding
2. Report with plot.

Constraints

1. You are allowed to use only the libraries given in the sample code below.
2. You need to work individually in this project.
3. Any use of internet's resources need to be cited in your code.

sortElements class definition:

```
#include<iostream>
#include<cstdlib> // used for random number generation
#include<math.h> // used for ceil and floor functions in adaptive sort
#include<ctime> // used for measuring time

using namespace std;

class sortElements{
protected:
    int numElements; //Number of elements of the array
    int* elements; //dynamic array with the elements of type int

public:
    sortElements(); // default constructor
    sortElements(int n); // non-default constructor
    ~sortElements(); // destructor

    void generateRandom(int seed, int lower, int upper); // will generate
                                                         numElement set of elements
                                                         randomly with the seed and range
                                                         from lower to upper
    void displayElements(int* arr); // display the given set
    int* getElementsArray(); // return the entire array of elements
    int getnumElements(); // return the number of elements

    // Necessary methods for each of the sorting algorithm
```

```
    // - Bubble and Quick sort methods
    // - Extra credit – Shell and Adaptive sort methods
};
```