CSC 323 Algorithm Design and Analysis
Fall 2019
Instructor: Dr. Natarajan Meghanathan

Project 4: Design and Implementation of Algorithms to Generate a Unimodal Array of Unique Elements and Implementation of the Binary Search Algorithm to Determine the Maximum Element in a Unimodal Array

Due by: Oct. 15th, 11.59 PM (in Canvas)

In this project, you will go through a sequence of four tasks to eventually find the maximum element in an unimodal array of unique elements using the binary search algorithm.

Task 1: Given an 'arraySize' and 'maxVal' (wherein maxVal > arraySize), design and implement an algorithm to generate a random array (say, it is pointed by arrayPtr) of size 'arraySize' such that the element values are in the range [1...maxVal] and *the array has unique elements*.

Task 2: Implement an in-place sorting algorithm of your choice (say, Bubble Sort, Insertion Sort or Selection Sort) to sort the array generated in Task 1. At the end of Task 2, the pointer arrayPtr would point to the sorted array.

Task 3: Design and implement a simple algorithm to generate a unimodal array of the sorted array obtained in Task 2. The elements of the unimodal array are to be stored in a different block of memory and hence should be pointed by a different pointer (say, unimodalArrayPtr).

Task 4: Implement the binary search algorithm discussed in class to determine the maximum element in the unimodal array generated in Task 3.

Run your code for three different values of arraySize (10, 20, 30) and capture the screenshots for each case (note that maxVal is 100 for all the three cases).

The screenshot captured should include the outputs for each of the above four tasks, i.e.:

Output for Task 1: Print the randomly generated array of unique elements;
Output for Task 2: Print the sorted version of the array generated in Task 1;
Output for Task 3: Print the unimodal version of the sorted array of Task 2;
Output for Task 4: Print the maximum value and the index of the maximum value as a result of running the binary search algorithm on the unimodal array of Task 3)

Submission:
(1) A code file that features your implementations for all the four tasks
(2) A PDF file that includes the following (the points included below also covers the corresponding code):

(i) Task 1 (20 pts): Provide the pseudo code, time complexity and space complexity of your algorithm to generate a random array of unique elements for an arraySize and maxVal passed as inputs. Also, address any space-time tradeoff that you encountered as part of your different design choices that you considered for the algorithm.
(ii) Task 2 (20 pts): Provide the pseudo code and time complexity analysis of the sorting algorithm used. Also, justify that its space complexity is $\Theta(1)$.
(iii) Task 3 (20 pts): Provide the pseudo code and time complexity analysis of the algorithm used to generate a unimodal array of the sorted array of Task 2.

(iv) Task 4 (40 pts) Implement/run the binary search algorithm on the unimodal array of Task 3 and capture the screenshots (featuring the task outputs as mentioned above) for the three input values of arraySize (10, 20, 30) and maxVal = 100 for all the three cases.