Project 1: Search algorithms complexity

dr Szymon Murawski

In this project you will implement linear and binary search, as well as measure time complexity of each implementation.

Task 1: *Utils*

We will be performing search operations on arrays of numbers, so an array generator will be very useful for this and future assignments. We would also like to easily paste the data we obtained from algorithm to excel spreadsheet, so a function that can save our measurements into file would be very nice.

1. Please create static class `ArrayGenerator` with methods of the following signatures:
   (a) `public static int[] GenerateRandomArray(int size)` - creates an array of size `size` of random integers
   (b) `public static int[] GenerateSortedAscendingArray(int size)` - creates an array of size `size` of random integers sorted in ascending order
2. For measuring time we will be using System.Diagnostics.Stopwatch class. Please get familiar with it.
3. Do this only after you are done with next two tasks! We would probably like to save data obtained in our analysis to file. Please create class `DataLogger` with methods of the following signatures:
   (a) `public static void LogData(string fileName, string[] columnLabels, int[,] data)` - saves into file `filename` a row with all the labels in `columnLabels` parameters, and then row by row writes to file all data in `data` parameter.

Task 2: *Linear search*

In lectures pseudocodes for three different implementations of linear search were given: simple, improved and improved with sentinel. In this task we will implement them, analyze complexity by measuring running time and operations performed in regards to input array size, and compare this three algorithms

1. Please create static class `LinearSearch` with functions of the following signatures:
   (a) `static bool SimpleLinearSearch(int[] array, int searchedValue)`
   (b) `static bool ImprovedLinearSearch(int[] array, int searchedValue)`
   (c) `static bool ImprovedLinearSearchWithSentinel(int[] array, int searchedValue)`
2. Each of those functions should be implement according to the pseudocode given during lecture

3. Measure the time it takes to perform the search operation using each of those methods for varying size of input array (generated using function previous task). For each array size perform multiple measurements (at least 3).
4. Gather the data in spreadsheet (excel or other)
5. Make a graph time T vs size n for each of the algorithms, you should obtain three graphs. Add a linear trend line on each of those graphs
6. Make one graph with only the trend lines from those three data sources.
7. Repeat steps 3 to 6, but this time measure dominant operations performed vs input array size. As dominant operation take comparison between an array element and searched value.

Task 3: *Binary search*

In this task we will analyze binary search. Please do the following:

1. Create static class `BinarySearch` with function of the following singature:
   (a) `static bool BinarySearch(int[] array, int searchValue)`
2. Implement this function according to the pseudocode given during lecture
3. Measure the time to perform search operation for varying size of input array. Do remember, that for binary search to work properly you need to feed sorted array into the function. For each array size perform multiple measurements
4. Make a graph time T vs size n, add a logarythmic trend line
5. Repeat steps 3 and 4, but this time measure dominant operations performed vs input array size. As dominant operation take comparison between an array element and searched value.

Task 4: *Analysis*

Please write conclusions on the topic: how each of those algorithms scale, what can you say about scalability of linear and binary search, how do the data you obtained correspond to complexity given during lectures?