Algorithm Assignment Report

Author: Divya Jagannatha

# Sorting Algorithm performance:

The purpose of Sorting Algorithms is to arrange the elements of array in order. The table1 below provides the result of sorting algorithms based on sorting n number of elements. Lambda search is faster than other sorting algorithms as per the result shown below. When the number of elements is 100 or less than 1000 sorting algorithms any one of them can be used. Since the time taken is insignificant. When the number of elements increases Bubble sort performs significantly poorer compared to other sorting algorithms. Insertion and Selection sort although good for few thousand elements, it drops in performance when we reach 100k plus elements. Merge sort is the best of basic sorts, taking only 2sec to sort 100k elements, but Lambda and quick sort are far ahead in performance, and an obvious choice as the number of elements increases or the number of elements is unknown. Lambda and Quicksort are also simple take less lines of code.

*Table 1: Sorting Algorithm Result*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sorting Algorithm Results. Time format (HH:mm:ss:milliseconds) | | | | |
| **Sorting Algorithms** | **n=100** | **n=5000** | **n=100,000** | **n=1000,0000** |
| **Insertion Sort** | 00:00:00.000000 | 00:00:00.000039 | 00:00:12.000688 | Approx 38hrs |
| **Selection Sort** | 00:00:00.000000 | 00:00:00.000055 | 00:00:20.000679 | Approx 60hrs |
| **Bubble Sort** | 00:00:00.000000 | 00:00:00.000230 | 00:01:35.000578 | Approx 11 days |
| **Merge Sort** | 00:00:00.000000 | 00:00:00.000015 | 00:00:02.000826 | Approx 6hrs |
| **Quick Sort** | 00:00:00.000000 | 00:00:00.000001 | 00:00:00.000036 | 00:00:04.000336 |
| **Using Lambda** | 00:00:00.000000 | 00:00:00.000004 | 00:00:00.000007 | 00:00:06.000174 |

## Benefits of avoiding mutation and using the delegate

When writing a program it is important to keep mutations to a minimum. The more an object can change, the more it can change unexpectedly. In some cases like simple arrays, this could mean wasted memory. In other data structures results could vary affecting the entirety of the program.

Delegates are references to a method. It is useful when we are trying to pass methods as parameters. It is used in my program, where in the switch case, we assign the user desired method to the delegate and pass it to the Display Runtime method. Without a delegate, we would have had to call the method in a more complicated way, requiring more code and logic.

# Search Algorithm Report:

The search algorithm's performance is approximately same. But as per results shown in the Table 2, The performance of Binary search is better than both Linear and search using Lambda just differing in few milliseconds.

## Binary Search:

The searching algorithm is faster than Linear Search algorithm. The algorithm uses sorted collection of data and uses divide and conquer approach. Binary search looks for a particular item by comparing the middle most item of the collection. If the match occurs, then the index of item is returned. If the middle item is greater than the item, it is searched in the sub-array to the left. Otherwise, the item is searched for in the sub-array to the right of the middle item. This process continues the sub-array until the subarray's size reduces to zero.

Table 2: Search Algorithms Results

|  |  |  |  |
| --- | --- | --- | --- |
| Search Algorithm Results. Time format (HH:mm:ss:milliseconds) | | | |
| **Search Results for 1000,000 element Array** | **Best case(target is the first item in the array)** | **Average case(target is the middle item in the array)** | **Worst case(target is the last item in Array)** |
| **Linear Search** | 00:00:00.000000 | 00:00:00.000002 | 00:00:00.000005 |
| **Binary Search** | 00:00:00.000000 | 00:00:00.000001 | 00:00:00.000001 |
| **Using Lambda** | 00:00:00.000001 | 00:00:00.000004 | 00:00:00.000010 |

# Performance Compare of different Data Structure:

Basic array is not very good for addition and deletion operations because the size of the array is constant and if we don’t know the exact incoming of elements, then one we will be wasting the memory or we may have to resize the array. But if we know the exact size of elements to be added in array then performance of array is better. Search works very fast because of indexing.

Stack and Queue are fastest in every category compared to other data structures based on the results shown in Table 3. They are dynamic and come in handy while using unordered elements.

Table 3: Performance Compare of different Data Structures

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Compare different data types for various functions. 1000,000 elements were added to each type to start  For Addition and deletion 100,000 elements were added/deleted  For Search and search by Index: A random number was chosen to search from | | | | |
|  | **Addition** | **Search** | **Deletion** | **Access by index** |
| **Array (T[])** | 00:00:00.000043 | 00:00:00.000001 | 00:00:00.000035 | 00:00:00.000000 |
| **Dynamic array (List)** | 00:00:00.000002 | 00:00:00.000000 | 00:00:50.000500 | 00:00:00.000000 |
| **Stack** | 00:00:00.000002 | 00:00:00.000001 | 00:00:00.000001 | 00:00:00.000000 |
| **Queue** | 00:00:00.000002 | 00:00:00.000001 | 00:00:00.000001 | 00:00:00.000000 |
| **Dictionary** | 00:00:00.000018 | 00:00:00.000006 | 00:00:00.000013 | 00:00:00.000001 |
| **SortedDictionary** | 00:00:00.000207 | 00:00:00.000126 | 00:00:00.000134 | 00:00:00.000001 |
| **HashSet** | 00:00:00.000018 | 00:00:00.000000 | 00:00:00.000014 | 00:00:00.000000 |

HashSet requires unique elements so it is very useful while working with sets, but they are not the fastest when compared to Addition and Deletion operations. It was surprising to see their performance operations. I expected it to be slower, because it cannot be searched based on index and since it is not sorted only Linear Search can be used.

Dictionary and SortedDictionary performance is slow but it is expected since it carries the double the memory because of key and value. And the key has to be unique which makes adding, deleting and searching harder. SortedDictionary is slower since it continuously sorts itself as we Add or Remove elements.

Benefits of Dictionary data structure from the functional programming perspective.

Dictionary data structure is versatile and is very commonly used in object oriented and functional programming. One of the most talked about features of functional programs is Pure functions, although dictionary itself is mutable it can also be carefully used to hold large nested datasets of immutable sets and lists.

References:

Data structures algorithms [2022-11-18]. Tutorials point [Electronic]. Available: <https://www.tutorialspoint.com/data_structures_algorithms/binary_search_algorithm.htm>

Microsoft [2022-11-18]. How to read from a text file. [Electronic]. Available: <https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/file-system/how-to-read-from-a-text-file>

Microsoft [2022-11-18]. Filtering data. [Electronic]. Available: <https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/filtering-data>

Stackoverflow [2022-11-17]. Delete an element from array. [Electronic]. Available: <https://stackoverflow.com/questions/496896/how-to-delete-an-element-from-an-array-in-c-sharp>

Geeksforgeeks [2022-11-20] Filter function. [Electronic]. Available: <https://www.geeksforgeeks.org/c-sharp-program-to-print-the-employees-whose-name-started-with-s-and-age-is-greater-than-23-using-linq/>

Submain [2022-11-20] Map and Reduce function. [Electronic]. Available: <https://blog.submain.com/csharp-functional-programming/>

Sharpcorner [2022-11-19] Async function c#. [Electronic]. Available: <https://www.c-sharpcorner.com/article/async-and-await-in-c-sharp/>

Geeksforgeeks [2022-11-18] Quicksort. [Electronic]. Available: <https://www.geeksforgeeks.org/quick-sort/>

Abdulgafour [2022-11-15] Sorting Algorithms. [Electronic]. Available: <https://hv.instructure.com/courses/6387/pages/lecture-and-resources?module_item_id=244313>