**Attila Katona**

**Assignment 3 PROG2120**

**Oct. 19, 2018**

**Searched Value = “Attila”**

**Searched Key = 50001**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Structure** | **Searching** | **Result** | **# of Searches** | **Elapsed Time** | **# of Values** |
| HashTable | Value | True | 1000 | 00:00:00.0334463 | 50002 |
| HashTable | Key | True | 1000 | 00:00:00.0000415 | 50002 |
| Dictionary<int,string> | Value | True | 1000 | 00:00:00.6393884 | 50002 |
| Dictionary<int,string> | Key | True | 1000 | 00:00:00.0000521 | 50002 |
| List<string> | Value | True | 1000 | 00:00:00.4737173 | 50002 |
| ArrayList | Value | True | 1000 | 00:00:00.3627985 | 50002 |

Above I have created a table documenting the test results. The table clearly shows that searching for a key in a hash table is the quickest method. These results vary every time the test is run but in general the fastest mode of search is for the key using either a hash table or a dictionary. I would say the Dictionary is faster than the hash table 3/5 times, but they are almost toe to toe most of the time. Below I am testing the process with 500 002 entries.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Structure** | **Searching** | **Result** | **# of Searches** | **Elapsed Time** | **# of Values** |
| HashTable | Value | True | 1000 | 00:00:01.2412943 | 500002 |
| HashTable | Key | True | 1000 | 00:00:00.0000514 | 500002 |
| Dictionary<int,string> | Value | True | 1000 | 00:00:04.9540953 | 500002 |
| Dictionary<int,string> | Key | True | 1000 | 00:00:00.0000393 | 500002 |
| List<string> | Value | True | 1000 | 00:00:04.1670145 | 500002 |
| ArrayList | Value | True | 1000 | 00:00:02.8889185 | 500002 |

As you can see the dictionary is now a little faster searching for the key than the hash table, but both are relatively the same speed as before with ten times less values. At high values like this the List is almost twice as slow as the Array List but with smaller values, they very similar.

I believe the reason Dictionary is the fastest out all of them especially the hash table is because the dictionary is a generic type which allows any data type to be used. This means that there is no boxing and unboxing. Also, since deep down it is essentially a hash table that has been optimized. The hash table uses objects for its type which means it needs to box and unbox. I also believe the dictionary is dynamically sorted on insert while hash tables are not.

The ArrayList according to my results is faster than the List<string>. Arraylists are dynamically allocated memory and stores the data in an array, so I believe this is why its faster to search but will have slower performance speeds in general. But the boxing and unboxing plays a role here as well when adding to a structure, as ArrayList does this and the List doesn’t so this might be why its slower for everything except searching. Since the ArrayList uses an array it can find things pretty fast using its index’s which may act like a key since it is just finding the value but not retrieving and using it. No casting needed.

References:

References:

<https://docs.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2008/4yh14awz(v=vs.90)>

<https://docs.microsoft.com/en-us/dotnet/api/system.collections.arraylist.contains?redirectedfrom=MSDN&view=netframework-4.7.2#System_Collections_ArrayList_Contains_System_Object_>

<https://stackoverflow.com/questions/1089132/net-hashtable-vs-dictionary-can-the-dictionary-be-as-fast>

<https://www.codeproject.com/Questions/394625/Which-is-better-array-ArrayList-or-List-in-terms-o>