There is different ways on how you can get data back from a dictionary, which one is the fastest? which is very interesting, because in the modern society every millisecond counts for something. We can conclude that using a for loop for a small amount of data is faster then using the key, by up to 0,000012 sec. up to 100 elements.

Which is fastest to search for a element Dictionary Vs List

As developers we most often need to save data *(in the form of: databases, dictionaries, lists, hasmps etc.)* so we can access it later on and we are interested in doing that in the most efficient way possible, so a potential customer/user are not left hanging. With that comes the importance of fast response time and less need of CPU performance.

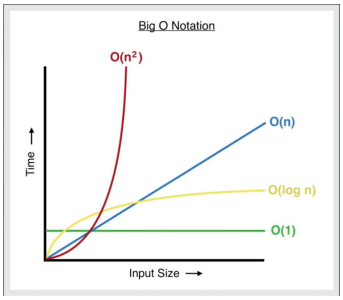
Dictionary – is an object which has the structure of key-value pairs, so all values giving within the dictionary are paired up with a unique key *(identifier)*. The advantage of having the key available is that giving the key makes is pretty fast to find the corresponding value. The time.

We all know the dictionary with its mathematical algorithm always deliver the same response time no matter the size of it or the position it needs to find. You could also run through all the element within the dictionary and manually compare the value to what you want to find.

Compare is to storing the key-value pairs in a list (where the key is the index). To then find a specific key-value pair, you would have to search the list from the beginning until a matching key is found. which means that, The longer the list is the more time it would take to find the key-value pair.

Big O notation

- Dictionary: O(1).
- List: O(N).



Problem statement:

- When is it faster to use a foreach rather then a dictionaries key to search for chosen element.

To answer the given problem statement we have giving our self, we have to do some experiments so we can collect some valuable and very useful data. But firstly we'll have to create a plan of execution, where we state the different experiments we want to do and what we expect the result turn out to be. by doing this will make it very easy to make a well-documented and credible conclusion.

Plan of execution

Experiment 1 - The purpose of experiment one is to declare that there even exists a difference between the response time of finding a item in a dictionary with either a key or through a foreach loop. To do this we simply do the following steps:

1) First we setup a dictionary with a decent element amount of elements

```
Stopwatch stopwatch = new Stopwatch();

//Console.WriteLine("Dictionary");
Dictionary<int, string> dictionary = new Dictionary<int, string>() {
{ 1,  "One" }, { 2,  "Two" }, { 3,  "Two" }, { 4,  "Two" }, { 5,  "Two" }, { 6,  "Two" }, { 7,  "Two" }, { 8,  "Two" },{ 9,  "Two" },  { 10, "Two" },
{ 11, "Two" }, { 12, "Two" }, { 13, "Two" }, { 14, "Two" }, { 15, "Two" }, { 16, "Two" }, { 17, "Two" }, { 18, "Two" },{ 19, "Two" },  { 20, "two" },
{ 21, "One" }, { 22, "Two" }, { 23, "Two" }, { 24, "Two" }, { 25, "Two" }, { 26, "Two" }, { 27, "Two" }, { 28, "Two" },{ 29, "Two" },  { 30, "Two" },
{ 31, "One" }, { 32, "Two" }, { 33, "Two" }, { 34, "Two" }, { 35, "Two" }, { 36, "Two" }, { 37, "Two" }, { 38, "Two" },{ 39, "Two" },  { 40, "Two" },
{ 41, "One" }, { 42, "Two" }, { 43, "Two" }, { 44, "Two" }, { 45, "Two" }, { 46, "Two" }, { 47, "Two" }, { 48, "Two" },{ 49, "Two" },  { 50, "Two" },
{ 51, "One" }, { 52, "Two" }, { 53, "Two" }, { 54, "Two" }, { 55, "Two" }, { 56, "Two" }, { 57, "Two" }, { 58, "Two" },{ 59, "Two" },  { 60, "Two" },
{ 61, "One" }, { 62, "Two" }, { 63, "Two" }, { 64, "Two" }, { 65, "Two" }, { 66, "Two" }, { 67, "Two" }, { 68, "Two" },{ 69, "Two" },  { 70, "Two" },
{ 71, "One" }, { 72, "Two" }, { 73, "Two" }, { 74, "Two" }, { 75, "Two" }, { 76, "Two" }, { 77, "Two" }, { 78, "Two" },{ 79, "Two" },  { 80, "Two" },
{ 81, "One" }, { 82, "Two" }, { 83, "Two" }, { 84, "Two" }, { 85, "Two" }, { 86, "Two" }, { 87, "Two" }, { 88, "Two" },{ 89, "Two" },  { 90, "Two" },
{ 91, "One" }, { 92, "Two" }, { 93, "Two" }, { 94, "Two" }, { 95, "Two" }, { 96, "Two" }, { 97, "Two" }, { 98, "Two" },{ 99, "Two" },  { 100, "Two"},
{ 101, "One"}, { 102, "Two"}, { 103, "Two"}, { 104, "Two"}, { 105, "Two"}, { 106, "Two"}, { 107, "Two"}, { 108, "Two"},{ 109, "Two"},  { 110, "Two"},};
stopwatch.Start();
var searchDic = dictionary[1];
stopwatch.Stop();
Console.WriteLine($"Time elapsed: {stopwatch.Elapsed}");
```

2) Then we create a timer which starts and ends at the exact time the dictionary does what we tell it to do - finding a specific element.
3) Then we want to test out two scenarios, how long time does it take to find:
    a) The first element by using the key.
    b) And find the first element through a foreach loop.
4) We Run each method 20 times and write the times down in a excel-spreadsheet.
5) Read the data and conclude if there is a difference or not between the response times.

Expectation #1: We expect that the foreach will be the fastest option to go with if the dictionary is that big.

***ATTENTION**: if there was any different, proceed to experiment 2. otherwise conclude.*

Experiment 2 The purpose of experiment two is to find out how big collection can be before it has a slower response time with find the element through a foreach loop than just finding the element by the key. To execute this experiment we did the following:

```
Stopwatch stopwatch = new Stopwatch();

//Console.WriteLine("Foreach over the list");
stopwatch.Start();
foreach (var item in dictionary)
{
    if (item.Key == 10)
    {
        stopwatch.Stop();
        Console.WriteLine($"Time elapsed: {stopwatch.Elapsed}");
    }
}
```

- We use the setup form experiment #1 and the we want to run the program with even spaced intervals around the point were we think we break of the dictionary being faster than the foreach.

Expectation #2: We expect to have enough data to have found the barrier were the foreach will be slower then the *"find by key".*

Experiments

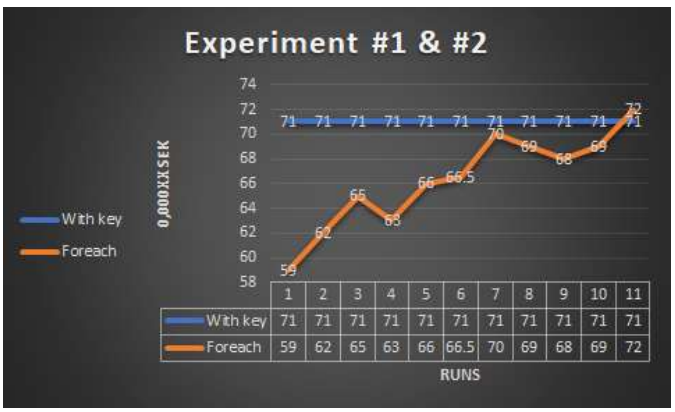
After we've done all the experiments that planned to do, we were left with a lot of raw data. To make it as clear a possible, we choose to make something visuel out of it.

Excel here we come!

The way the processed the data was, by firstly sorting it *(lowest to highest)* and then removing the two lowest and two highest time - reason being, so we potentially eliminated any wrong data that might have snooked in under our experiments phase. Then we took the average of each of runs *(key = 1, 10, 20… 100)*

	With key 1	Foreach 1	Foreach 10	Foreach 20	Foreach 30	Foreach 40	Foreach 50	Foreach 60	Foreach 70	Foreach 80	Foreach 90	Foreach 100
Response time	60	51	52	54	54	55	56	58	58	59	59	60
	62	51	53	55	55	55	56	59	59	60	59	61
	63	52	54	55	55	55	57	60	59	60	61	62
	63	53	54	55	57	57	57	60	61	60	61	62
	63	53	55	56	57	58	59	60	62	61	62	63
	64	54	55	57	58	58	59	61	63	61	62	64
	65	54	56	57	58	59	60	61	63	62	63	64
	65	54	57	57	59	59	61	61	63	63	63	64
	65	54	57	57	59	60	61	62	64	63	63	64
	66	55	57	58	59	61	61	62	64	63	64	65
	66	55	57	61	59	61	61	62	64	63	64	65
	69	55	58	61	60	63	64	63	64	64	64	66
	69	55	58	62	60	64	64	64	65	64	64	67
	69	57	59	66	60	66	64	64	66	64	65	68
	70	57	60	66	61	68	65	66	67	64	66	69
	62	57	61	69	61	71	66	71	69	64	69	69
	73	58	62	69	62	65	67	73	70	68	69	69
	75	62	62	70	62	74	68	80	73	69	73	83
	69	68	68	71	64	62	70	91	73	69	74	86
	81	79	93	74	91	81	73	92	99	70	95	94
	81	92	105	105	123	88	87	100	322	79	227	97
avg	71	59.5625	61.875	65.4375	63.1875	66.3125	66.5	70.0625	69.375	67.625	69.1875	71.875

The most important row to look at, is the bottom one, this is also the data we will use to create our graf from.



We can clearly see that the more elements our foreach has to loop through, the slower it gets. It also seems like we don't have a completely linear-curve, which we're going to discuss in our conclusion.

Conclusion Refleksion

After we have successfully executed the experiments, we are then able to conclude that it is possible to tell when the foreach loop is faster than finding the element by key.
The data we have collected show us that, when a collection reach around or below 100 it is favorable to use the foreach and go through every single element instead of using the dictionary key to find the specific element.

The answer is: 100 elements.

If you in a specific case like in space research or developing of space technology it can be very important to have the fastest response time, in this case we could save up to 0.000012 sec

why is our curve not linear: as we saw, our curve didn't follow a "straight" linear graf. The reason why we think this is happening, is because of the computers CPU usage.
We could properly have avoided this completely, by simply just collecting more data, the program should maybe have run for 50 or even 100 time per run *(instead of our decided 20).*

Refs:

- https://docs.microsoft.com/en-us/previous-versions/ms379571(v=vs.80)?redirectedfrom=MSDN