How do you sort a dictionary by value?

Ask Question

740

★

143

I often have to sort a dictionary, consisting of keys & values, by value. For example, I have a hash of words and respective frequencies, that I want to order by frequency.

There is a `SortedList` which is good for a single value (say frequency), that I want to map it back to the word.

SortedDictionary orders by key, not value. Some resort to a custom class, but is there a cleaner way?

c# .net sorting dictionary

edited Oct 16 '18 at 16:55
Tiago Martins Peres
2,255 7 20 36

asked Aug 2 '08 at 0:40
Kalid
11.1k 12 38 44

Aside from just sorting the dictionary (as in the accepted answer), you could also just create an `IComparer` that does the trick (true that it accepts a key to compare, but with a key,

497

Use:

```
using System.Linq.Enumerable;
...
List<KeyValuePair<string, string>> myList = aDictionary.To

myList.Sort(
    delegate(KeyValuePair<string, string> pair1,
    KeyValuePair<string, string> pair2)
    {
        return pair1.Value.CompareTo(pair2.Value);
    }
);
```

Since you're targeting .NET 2.0 or above, you can simplify this into lambda syntax -- it's equivalent, but shorter. If you're targeting .NET 2.0 you can only use this syntax if you're using the compiler from Visual Studio 2008 (or above).

```
var myList = aDictionary.ToList();

myList.Sort((pair1,pair2) => pair1.Value.CompareTo(pair2.V
```

edited Aug 31 '17 at 23:01
Peter Mortensen
14.1k 19 88 114

answered Aug 2 '08 at 1:15
Leon Bambrick
17.9k 8 43 71

23 I used this solution (Thanks!) but was confused for a minute until I read Michael Stum's post (and his code snippet from

Arnis Lapsa Sep 26 '10 at 16:40

20 To sort descending switch the x and the y on the comparison: myList.Sort((x,y)=>y.Value.CompareTo(x.Value)); – Arturo Oct 16 '12 at 22:43

5 I think it's worth noting that this requires Linq for the ToList extension method. – Ben Oct 15 '14 at 23:41

6 You guys are waaaay over complicating this -- a dictionary already implements `IEnumerable`, so you can get a sorted list like this: `var mySortedList = myDictionary.OrderBy(d => d.Value).ToList();` – BrainSlugs83 Mar 19 '18 at 21:12

499

Use LINQ:

```
Dictionary<string, int> myDict = new Dictionary<string, in
myDict.Add("one", 1);
myDict.Add("four", 4);
myDict.Add("two", 2);
myDict.Add("three", 3);

var sortedDict = from entry in myDict orderby entry.Value
```

This would also allow for great flexibility in that you can select the top 10, 20 10%, etc. Or if you are using your word frequency index for `type-ahead`, you could also include `StartsWith` clause as well.

edited Aug 31 '17 at 23:02

Peter Mortensen

13 How can I change sortedDict back into a Dictionary<string, int>? Posted new SO question here: stackoverflow.com/questions/3066182/… – Kache Jun 17 '10 at 22:47

1 Sadly this does not work on VS2005 because of .net framework 2.0 there (no LINQ). It is good to have also the Bambrick's answer. – Smalcat Nov 30 '10 at 11:23

18 I'm not sure if it always works because iterating over dictionary doesn't guarantee that KeyValuePairs are "pulled" in the same order they have been inserted. Ergo, it doesn't matter if you use orderby in LINQ because Dictionary can change order of inserted elements. It usually works as expected but there is NO GUARANTEE, especially for large dictionaries. – Bozydar Sobczak Jan 27 '12 at 8:23

15 Return type should be `IEnumerable<KeyValuePair<TKey, TValue>>` or an `OrderedDictionary<TKey, TValue>`. Or one should use a `SortedDictionary` from the start. For a plain `Dictionary` the MSDN clearly states "The order in which the items are returned is undefined.". It seems that @rythos42 's latest edit is to blame. :) – Boris B. Feb 7 '12 at 20:05

14 Please disregard all suggestions of `.ToDictionary` - standard dictionaries do not guarantee a sort order – AlexFoxGill Mar 15 '13 at 16:57

210

```
var ordered = dict.OrderBy(x => x.Value);
```

answered Nov 11 '10 at 17:16

sean
5,807 7 40 52

12 @theJerm: not true – AlexFoxGill Mar 15 '13 at 16:56

2 @theJerm by putting the sorted items back to a dictionary is the order guaranteed then? It might work today, but it's not guaranteed. – nawfal Oct 31 '13 at 7:41

9 There should not be a cast back to a dictionary because dictionaries are not ordered. There's no guarantee the KeyValuePairs will stay in the order you want. – David DeMar Nov 19 '14 at 14:54

156

Looking around, and using some C# 3.0 features we can do this:

```
foreach (KeyValuePair<string,int> item in keywordCounts.Or
{
    // do something with item.Key and item.Value
}
```

This is the cleanest way I've seen and is similar to the Ruby way of handling hashes.

edited Jan 29 '13 at 17:11
Alexander
2,225 2 19 32

answered Aug 2 '08 at 0:43
Kalid
11.1k 12 38 44

`keywordCounts.OrderBy(key => key.Value) select item).ToDictionary(t => t.Key, t => t.Value)` - just a small addition to your answer :) Thanks, btw :) – Andrius Naruševičius Sep 21 '12 at 7:17

5 @AndriusNaruševičius: If you add the resulting items back into a dictionary, you will destroy the order, as dictionaries are not guaranteed to be ordered in any particular fashion. – O. R. Mapper Jan 24 '15 at 10:25

This was handy. How can it be inverted to go the other way? – Dan Hastings Jan 17 '17 at 10:33

150

You can sort a Dictionary by value and save it back to itself (so that when you foreach over it the values come out in order):

```
dict = dict.OrderBy(x => x.Value).ToDictionary(x => x.Key,
```

Sure, it may not be correct, but it works.

edited Apr 30 '15 at 9:29

answered Jun 22 '11 at 10:26

Matt Frear
36.7k 9 52 71

8 You can also use OrderByDescending if you want to sort into a descending list. – Mendokusai Aug 17 '11 at 2:16

2 The Dictionary output is NOT guaranteed to have any particular sort order. – Roger Willcocks Mar 24 '15 at 2:12

3 I would be quite concerned to see this in production code. It is not guaranteed and could change at any time. Not that I shy away from pragmatic solutions, it just shows a lack of understanding of the data structure imo. – jamespconnor Apr 28 '15 at 13:24

58

On a high level, you have no other choice then to walk through the whole Dictionary and look at each value.

Maybe this helps: http://bytes.com/forum/thread563638.html Copy/Pasting from John Timney:

```
Dictionary<string, string> s = new Dictionary<string, stri
s.Add("1", "a Item");
s.Add("2", "c Item");
s.Add("3", "b Item");

List<KeyValuePair<string, string>> myList = new List<KeyVa
myList.Sort(
    delegate(KeyValuePair<string, string> firstPair,
    KeyValuePair<string, string> nextPair)
    {
        return firstPair.Value.CompareTo(nextPair.Value);
    }
);
```

edited Jan 5 '12 at 11:50

Community ♦
1 1

3 stringnextPair -> string> nextPair stringfirstPair -> string> firstPair – Art Feb 25 '10 at 23:38

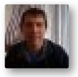
1 Perfect non-Linq solution. It never ceases to amaze me how people feel the need to use Linq even when it's absolutely not required to solve the problem. With C# 3, I believe you can also simplify the Sort to just use a lambda: myList.Sort((x, y) => x.Value.CompareTo(y.Value)); – RobinHood70 Jul 6 '16 at 1:28

23

You'd never be able to sort a dictionary anyway. They are not actually ordered. The guarantees for a dictionary are that the key and value collections are iterable, and values can be retrieved by index or key, but there is no guarantee of any particular order. Hence you would need to get the name value pair into a list.

edited May 15 at 2:00

answered Dec 19 '08 at 22:47


Roger Willcocks
1,243 9 23

1 A sorted dictionary could yield a list of key-value pairs though. – recursive Dec 20 '08 at 5:19

1 @recursive Any dictionary should yield that. Interesting to note that my answer, which is correct, but incomplete (could have done what the better examples did) is voted below an invalid answer that would result in exceptions on duplicate values in

16

You do not sort entries in the Dictionary. Dictionary class in .NET is implemented as a hashtable - this data structure is not sortable by definition.

If you need to be able to iterate over your collection (by key) - you need to use SortedDictionary, which is implemented as a Binary Search Tree.

In your case, however the source structure is irrelevant, because it is sorted by a different field. You would still need to sort it by frequency and put it in a new collection sorted by the relevant field (frequency). So in this collection the frequencies are keys and words are values. Since many words can have the same frequency (and you are going to use it as a key) you cannot use neither Dictionary nor SortedDictionary (they require unique keys). This leaves you with a SortedList.

I don't understand why you insist on maintaining a link to the original item in your main/first dictionary.

If the objects in your collection had a more complex structure (more fields) and you needed to be able to efficiently access/sort them using several different fields as keys - You would probably need a custom data structure that would consist of the main storage that supports O(1) insertion and removal (LinkedList) and several indexing structures - Dictionaries/SortedDictionaries/SortedLists. These indexes would use one of the fields from your complex class as a key and a pointer/reference to the LinkedListNode in the LinkedList as a value.

All of the above is only justified if you are going to do some look-up heavy processing. If you only need to output them once sorted by frequency then you could just produce a list of (anonymous) tuples:

```
var dict = new SortedDictionary<string, int>();
// ToDo: populate dict

var output = dict.OrderBy(e => e.Value).Select(e => new {fr
e.Key}).ToList();

foreach (var entry in output)
{
    Console.WriteLine("frequency:{0}, word: {1}",entry.fre
}
```

edited Oct 31 '13 at 7:43

nawfal
44.6k 36 260 307

answered Dec 13 '12 at 6:19

Zar Shardan
4,263 1 27 32

14

```
Dictionary<string, string> dic= new Dictionary<string, str
var ordered = dic.OrderBy(x => x.Value);
return ordered.ToDictionary(t => t.Key, t => t.Value);
```

answered Jul 20 '15 at 11:01

mrfazolka

10

Or for fun you could use some LINQ extension goodness:

```
var dictionary = new Dictionary<string, int> { { "c", 3 },
dictionary.OrderBy(x => x.Value)
  .ForEach(x => Console.WriteLine("{0}={1}", x.Key,x.Value
```

answered Jun 30 '10 at 11:12

mythz
121k 14 198 341

10

Sort values

This show how to sort the values in a Dictionary. We see a console program you can compile in Visual Studio and run. It adds keys to a Dictionary and then sorts them by their values. Remember that Dictionary instances are not initially sorted in any way. We use the LINQ orderby keyword in a query statement.

OrderBy Clause Program that sorts Dictionary [C#]

```
using System;
using System.Collections.Generic;
using System.Linq;

class Program
{
    static void Main()
```

```
            // Order by values.
            // ... Use LINQ to specify sorting by value.
            var items = from pair in dictionary
                    orderby pair.Value ascending
                    select pair;

            // Display results.
            foreach (KeyValuePair<string, int> pair in items)
            {
                Console.WriteLine("{0}: {1}", pair.Key, pair.Va
            }

            // Reverse sort.
            // ... Can be looped over in the same way as above
            items = from pair in dictionary
            orderby pair.Value descending
            select pair;
        }
    }
```

Output

```
dog: 0
cat: 1
programmer: 2
eel: 3
mouse: 5
```

edited Oct 24 '12 at 14:32

CAMOBAP
3,187 8 44 78

answered Jul 20 '12 at 9:49

lasitha edirisooriya
2,751 2 21 18

9

```
Dim MyDictionary As SortedDictionary(Of String, MyDictiona

MyDictionaryListView.ItemsSource = MyDictionary.Values.Ord
entry.MyValue)

Public Class MyDictionaryEntry ' Need Property for GridVie
    Public Property MyString As String
    Public Property MyValue As Integer
End Class
```

XAML:

```
<ListView Name="MyDictionaryListView">
    <ListView.View>
        <GridView>
            <GridViewColumn DisplayMemberBinding="{Binding
Header="MyStringColumnName"></GridViewColumn>
            <GridViewColumn DisplayMemberBinding="{Binding
Header="MyValueColumnName"></GridViewColumn>
         </GridView>
    </ListView.View>
</ListView>
```

edited Aug 31 '17 at 23:04
Peter Mortensen
14.1k 19 88 114

answered Apr 23 '10 at 9:36
BSalita
4,261 3 39 51

The easiest way to get a sorted Dictionary is to use the

```
{
    sortedSections = new SortedDictionary<int, string>(sec
}
```

`sortedSections` will contains the sorted version of `sections`

edited Jan 5 '12 at 9:25
Bertrand Marron
15.8k 7 43 82

answered Apr 2 '10 at 22:36
Alex Ruiz
83 1 1

5 As you mention in your comment, `SortedDictionary` sorts by keys. The OP wants to sort by value. `SortedDictionary` doesn't help in this case. – Marty Neal Sep 12 '12 at 15:50

Well... If he/she (you) can, just set the values as the keys. I timed the operations and `sorteddictionary()` always won out by at least 1 microsecond, and it's much easier to manage (as the overhead of converting it back into something easily interacted with and managed similarly to a Dictionary is 0 (it is already a `sorteddictionary`)). – mbrownnyc Oct 25 '13 at 14:17

2 @mbrownnyc - nope, doing that requires the assumption or precondition that the VALUES are unique, which is not guaranteed. – Roger Willcocks Mar 24 '15 at 2:14

The other answers are good, if all you want is to have a "temporary" list sorted by Value. However, if you want to be unsorted, and you want the other one to be sorted, you could create your bijection with code like

```
var dict = new Bijection<Key, Value>(new Dictionary<Key,Va
                                     new SortedDictionary<Value,
```

You can use `dict` like any normal dictionary (it implements `IDictionary<K, V>`), and then call `dict.Inverse` to get the "inverse" dictionary which is sorted by `Value` .

`Bijection<K1, K2>` is part of Loyc.Collections.dll, but if you want, you could simply copy the source code into your own project.

Note: In case there are multiple keys with the same value, you can't use `Bijection` , but you could manually synchronize between an ordinary `Dictionary<Key,Value>` and a `BMultiMap<Value,Key>` .

edited Oct 16 '18 at 15:51

Tilman B. aka Nerdyyy
124 1 10

answered Feb 26 '16 at 7:15

Qwertie
8,263 13 78 121

Similar to http://stackoverflow.com/questions/268321 but can replace each Dictionary with SortedDictionary. Although the answers look not to support duplicate values (assumes 1 to 1). – crokusek Jul 11 '16 at 23:35

3

```
Dictionary<int, int> dict = new Dictionary<int, int>();
dict.Add(21,1041);
dict.Add(213, 1021);
dict.Add(45, 1081);
dict.Add(54, 1091);
dict.Add(3425, 1061);
sict.Add(768, 1011);
```

1) you can use `temporary dictionary to store values as` :

```
Dictionary<int, int> dctTemp = new Dictionary<int,

foreach (KeyValuePair<int, int> pair in dict.OrderI
{
    dctTemp .Add(pair.Key, pair.Value);
}
```

edited Feb 2 '15 at 11:21
RajeshKdev
5,194 5 44 70

answered Feb 2 '15 at 10:46
Akshay Kapoor
107 3

0

Actually in C#, Dictionaries dint have sort() methods, as you are more interested in sort by values, you cant get values until you provide them key, in short, you need to iterate through them, using LINQ's Order By,

```
{
    Console.WriteLine(item);// items are in sorted order
}
```

you can do one trick,

```
var sortedDictByOrder = items.OrderBy(v => v.Value);
```

or

```
var sortedKeys = from pair in dictName
            orderby pair.Value ascending
            select pair;
```

its also depend on what kind of values you are storing,
is it single (like string, int) or multiple (like List, Array, user defined class),
if single you can make list of it then apply sort.
if user defined class, then that class must implement IComparable,
`ClassName: IComparable<ClassName>` and override
`compareTo(ClassName c)` as they are more faster than LINQ, and more object oriented.

edited Feb 26 at 12:45

answered Feb 26 at 12:39

Ashish Kamble
749 1 6 20

```
Dictionary <<string, string>> ShareUserNewCopy =
       ShareUserCopy.OrderBy(x => x.Value).ToDictionary(pa
                                                        pa
```

edited Jul 24 '12 at 12:30
Tisho
6,554 5 35 50

answered Jul 24 '12 at 12:24
pawan Kumar
29 1

8 By putting the sorted items back into a dictionary, they are no longer guaranteed to be sorted when you enumerate the new dictionary. – Marty Neal Sep 12 '12 at 15:48

2 And why are you adding this answer when this is already answered? – nawfal Oct 31 '13 at 7:42

-2

Given you have a dictionary you can sort them directly on values using below one liner:

```
var x = (from c in dict orderby c.Value.Order ascending se
c.Key, c=>c.Value);
```

answered May 31 '14 at 22:30
aggaton
1,560 1 14 27

protected by Mureinik May 3 '15 at 6:40

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?