

Forum:[Hack Forums](#)

Thread: [Implementing a Dictionary-like structure for large sets?](#)

severos - November 1, 2016, 11:50 am

So lets say I need something like a Dictionary (small access time) but with large number of keys (lets say around 100million keys), what would a good approach for this problem to start thinking??

P.S: I've tried the normal Dictionary and it's giving me out of memory exception, so I'll have to deal with that somehow... no idea how tho.

mrkoolaid - November 1, 2016, 12:19 pm

You may need to step into the realm of math for large sets of data. I run into this problem when I'm trying to solve stuff on hackerrank.com. You will need some kind of sorting algorithm. (https://en.wikipedia.org/wiki/External_sorting)

I was also reading that the windows sort command could make short work of a huge file.

severos - November 1, 2016, 12:41 pm

(Nov 1, 2016 12:19 PM)mrkoolaid Wrote: [\[CITE\]](#) You may need to step into the realm of math for large sets of data. I run into this problem when I'm trying to solve stuff on hackerrank.com. You will need some kind of sorting algorithm. (https://en.wikipedia.org/wiki/External_sorting)

I was also reading that the windows sort command could make short work of a huge file. That sounds like a good point to start at. Thanks for the link.

mrkoolaid - November 1, 2016, 12:56 pm

(Nov 1, 2016 12:41 PM)severos Wrote: [\[CITE\]](#) That sounds like a good point to start at. Thanks for the link.

I once tried to make a porn site using the pornhub db, it's quite fucking huge. I resorted to splitting the file into chunks and sorting it with a merge sort i found online. I never got the porn site up, I think we know what distracted me, let's leave it at that lol

Ninja - November 1, 2016, 6:43 pm

How about just a normal database?

severos - November 1, 2016, 9:55 pm

(Nov 1, 2016 06:43 PM)Ninja Wrote: [\[CITE\]](#) How about just a normal database?

Assignment says I can't use it, problem would have been solved long ago if I was able to use one QQ

Game0ver - November 1, 2016, 11:49 pm

What about a hashset or even coding your own table to work to your advantage using chaining or partitioning.

Either way should be faster than dictionary.

Ninja - November 2, 2016, 9:10 am

(Nov 1, 2016 09:55 PM)severos Wrote: [\[CITE\]](#) Assignment says I can't use it, problem would have been solved long ago if I was able to use one QQ

Well then this sucks. ;_;

Snoweyes - November 2, 2016, 10:16 am

Im not sure, but what about a lookup table? Check the links below if it suits for you.

https://en.wikipedia.org/wiki/Lookup_table

[https://msdn.microsoft.com/en-us/library/bb460184\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/bb460184(v=vs.110).aspx)

<https://www.saschabaumann.com/c-loopup-vs-dictionary/>

severos - November 2, 2016, 10:24 am

Thanks all for the links, I'll be reading them when I'm home.

Ninja - November 2, 2016, 11:36 am

(Nov 2, 2016 10:24 AM)severos Wrote: [\[CITE\]](#) Thanks all for the links, I'll be reading them when I'm home.

Also just a tip: You probably want to add threading to your algorithm if it is actually a huge amount of data you are working with.

severos - November 2, 2016, 11:49 am

Ok, from a fast lookup on the Lookup table, it doesn't seem to be what I'm looking for as it says it's built at compile time while my data is being created in runtime, also it shouldn't allow mutliple values for a single key (unless I've understood something wrong in those tables).

(Nov 2, 2016 11:36 AM)Ninja Wrote: [\[CITE\]](#) Also just a tip: You probably want to add threading to your algorithm if it is actually a huge amount of data you are working with.

Multi threading will be added later, after making it work in a single thread and getting the algorithm correct I'll be aiming for performance enhancement.

The-One - November 2, 2016, 12:25 pm

You could;
Partition the list

Lazy load the list (on-demand)

Use a native language

Also, what is the size of one single object and what kind of types does it hold?

Game0ver - November 2, 2016, 12:28 pm

Again, look into hash table...

A simple way to look at how a hash table work is...

You are storing data using arrays (can be 2d(bucketing) or 1d(partitioning, chaining..) depending on your implementation)

At runtime arrays are stored together in memory making them easily accessible by an index (in cpp we can use a pointer and just increase the pointer..).

Anyways moving forward in order to access any item in array we can insert an index eg.

arr[600]...

Which will perform in $O(1)$ or constant time..

In order to lookup a value in a linked list or dictionary (most implementations, including the ones provided by .net) have to loop through each item linked in the linked list style list in order to locate what you are looking for..

e.g.

list[600]

would

```
int target = 600;
int count = 0;
for(node n = start; start != end;)
{
    count++;
    if(count == target)
        return n;
    else
        n = n->next node...
}
```

which in worst case would require $O(n)$ time to perform.

So when using a hash table you have an algorithm that will calculate the index of an array to store your data at.

e.g. lets say your table is strings and a hash algorithm calculates the index of "hackfroums" to be 67... when ever you request that data again, the hash algorithm will calculate 67 again.

So accessing the data would only be $O(n)$ time because we know exactly where the data is in the array.

The only time your run time is going to start decreasing is when you start getting collisions which will then depending on your remember.. (chaining, bucketing, partitioning, etc).

severos - November 2, 2016, 9:24 pm

(Nov 2, 2016 12:25 PM)The-One Wrote: [CITE](#) You could;
Partition the list

Lazy load the list (on-demand)

Use a native language

Also, what is the size of one single object and what kind of types does it hold?Thanks for the suggestions, I'll look into them and hope to find a solution.

A single object (key value pair in dictionary): the key is string of a name (variable length but usually

not larger than 10 characters) and the value is an object of 2 longs and a byte (around 17 bytes of row data, a bit extra for object and other stuff .Net does). Using a Dictionary I'm able to reach a bit less than 12million objects.

@GameOver

Thanks for the explanation, I'll look further into it and might get inspired to something.

Mistermad - November 3, 2016, 12:34 am

Why not make your own list?

and create a new Add method so that you can check if the value you want to add is already in the list or not?

The only problem I can see is that this will get slow once you have a few items.

```
Code class NewList<T> : List<T>
{
    public new void Add(T item)
    {
        //Check whether item is already in the list
    }
}
```

severos - November 3, 2016, 1:31 am

(Nov 3, 2016 12:34 AM)Mistermad Wrote: [\[CITE\]](#) Why not make your own list?

and create a new Add method so that you can check if the value you want to add is already in the list or not?

The only problem I can see is that this will get slow once you have a few items.

Because I don't want access time $O(n)$ structure, imagine that I have to check over 100million objects just to add one... and imagine the time I have to build the 100million objects at first place -think it's $O(n^2)$ -

Snoweyes - November 3, 2016, 10:34 am

(Nov 2, 2016 09:24 PM)severos Wrote: [\[CITE\]](#) Thanks for the suggestions, I'll look into them and hope to find a solution.

A single object (key value pair in dictionary): the key is string of a name (variable length but usually not larger than 10 characters) and the value is an object of 2 longs and a byte (around 17 bytes of row data, a bit extra for object and other stuff .Net does). Using a Dictionary I'm able to reach a bit less than 12million objects.

@GameOver

Thanks for the explanation, I'll look further into it and might get inspired to something.

I found this article which might help you:

<http://www.codeproject.com/Tips/837172/Creating-PartitionedDictionary>

Its what The-One mentioned, partitioning the dictionary.

This StackOverflow answer explains why you get a memory out of range exception:

<http://stackoverflow.com/a/3657189>

severos - November 3, 2016, 11:23 am

(Nov 3, 2016 10:34 AM)Snoweyes Wrote: [\[CITE\]](#) I found this article which might help you:

<http://www.codeproject.com/Tips/837172/Creating-PartitionedDictionary>

Its what The-One mentioned, partitioning the dictionary.

This StackOverflow answer explains why you get a memory out of range exception:

<http://stackoverflow.com/a/3657189>

Thanks for the link, I'll read it soon.

As for the out of memory exception, I tried running the diagnostic tools and it was reading 1GB of memory before throwing the exception, no idea why tho.

Khamul - November 3, 2016, 4:25 pm

I guess you could implement a HashTable with linear probing? Could be good although deletion would be a problem.