Subject: New containers - naming
Posted by mirek on Sun, 03 Feb 2013 18:46:31 GMT
View Forum Message <> Reply to Message

Hi,

in past weeks, I was working on a new container that will become a basis of couple of other containers.

The new container, which I have named 'InVector', excels at inserting or removing elements at arbitrary locations (trading this feature for somewhat slower operator[]).

It is so fast that it is possible to create 'std::set' equivalent (means, elements are added at UpperBound index, so that InVector stays sorted at all times and it has log(n) search of elements), which is quite superior to set (which is implemented with using binary trees). It is moderately faster and it consumes significantly less memory.

InVector thus makes possible to create a new set of containers, alike to 'Array, Index, VectorMap, ArrayMap', where instead of hashing binary search is used. Of course, hashing is still much faster, but the advantage of binary search is the ability to perform range searches. Also, it consumes less memory than hashing.

Now one small funny problem is how to name these new containers. My current state of mind is somehing like

InArray (Array counterpart)
Order (Index counterpart)
OrderVector (VectorMap counterpart)
OrderArray (ArrayMap counterpart)

Any better ideas?

Mirek

P.S.: InVector can be seen in sandbox/InVector, with tests and benchmarks...

Subject: Re: New containers - naming
Posted by dolik.rce on Sun, 03 Feb 2013 19:39:25 GMT
View Forum Message <> Reply to Message

Hi Mirek,

That great, now Il be even more addicted to NTL...

Just to make sure, InVector stands for "insert vector"?

IMHO it would be great if the name of the new containers contained the information how it differs

from the regular ones. I think the biggest advantage of this will be the range searches, so what about something like RangeIndex, RangeVector and RangeArray?

BTW: Will you tell us how it works, or is it left as an exercise for the reader? I suspect you told me once about this idea in past over a beer, but I want spoil it just yet for other curious programmers here

Best regards,
Honza

Subject: Re: New containers - naming
Posted by mirek on Sun, 03 Feb 2013 20:17:58 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Sun, 03 February 2013 14:39Hi Mirek,

That great, now Il be even more addicted to NTL...

Just to make sure, InVector stands for "insert vector"?

Well, either that or that it supports "in" positions. Of course, better ideas are welcome

Quote:
IMHO it would be great if the name of the new containers contained the information how it differs from the regular ones.
I think the biggest advantage of this will be the range searches, so what about something like RangeIndex, RangeVector and RangeArray?

Well, my original line of thinking was that the main difference is that keys are ordered... (also note std::unordered_map etc...). But OrderedIndex is perhaps too long, so "Order".

Quote:
BTW: Will you tell us how it works, or is it left as an exercise for the reader? I suspect you told me once about this idea in past over a beer, but I want spoil it just yet for other curious programmers here

The basic storage is Vector<Vector<T>>, the size of inner vectors is kept between some thresholds, otherwise they are split / merged. For such small amount of data, Vector was always faster at inserting/removing than any node based structures.

The key to provide relatively fast index access is sort of numeric binary tree, which is easy to maintain on inserts/removes (removes are not implemented yet) and still quite fast for operator[]. Plus, there is a per thread cache to speed up simple scans. And a good thing is that Find[Upper/Lower]Bound methods can compute the index of an element very cheaply.

Mirek

Subject: Re: New containers - naming
Posted by navi on Sun, 03 Feb 2013 22:36:14 GMT
View Forum Message <> Reply to Message

wow. new container for NTL. awesome.
maybe an acronym prefix? like "aid" (Arbitrary insert delete)
e.g. aidArray, aidVector etc
also from description it sounds very similar to List.

Quote from wiki...Lists are typically implemented either as linked lists (either singly or doubly linked) or as arrays ...
... Each element in the list has an index. The first element commonly has index 0 or 1 (or some other predefined integer). Subsequent elements have indices that are 1 higher than the previous

It is possible to retrieve the element at a particular index.
It is possible to traverse the list in the order of increasing index.
It is possible to change the element at a particular index to a different value, without affecting any other elements.
It is possible to insert an element at a particular index. The indices of higher elements at that are increased by 1.
It is possible to remove an element at a particular index. The indices of higher elements at that are decreased by 1.

...Lists can be implemented as self-balancing binary search trees holding index-value pairs, providing equal-time access to any element (e.g. all residing in the fringe, and internal nodes storing the right-most child's index, used to guide the search), taking the time logarithmic in the list's size, but as long as it doesn't change much will provide the illusion of random access and enable swap, prefix and append operations in logarithmic time as well...

As the name implies, lists can be used to store a list of records. The items in a list can be sorted for the purpose of fast search (binary search)....

...lists are easier to realize than sets, a finite set in mathematical sense can be realized as a list with additional restrictions, that is, duplicate elements are disallowed and such that order is irrelevant. If the list is sorted, it speeds up determining if a given item is already in the set but in order to ensure the order, it requires more time to add new entry to the list. In efficient implementations, however, sets are implemented using self-balancing binary search trees or hash tables, rather than a list.

Subject: Re: New containers - naming
Posted by Didier on Mon, 04 Feb 2013 19:10:45 GMT
View Forum Message <> Reply to Message

Hi Mirek,

great job as usual

For the naming I think that it must be consistent for all new containers so that developers can easily identify just by reading:
instead of :
Quote:InArray (Array counterpart)
Order (Index counterpart)
OrderVector (VectorMap counterpart)
OrderArray (ArrayMap counterpart)

I would rather put:
OrderArray (Array counterpart)
OrderIndex (Index counterpart)
OrderVector (VectorMap counterpart)
OrderArray (ArrayMap counterpart)

The prefix should always be the same: 'order', 'in' or anything else but I think it should be as explicit as possible so 'order' seems correct to me

Subject: Re: New containers - naming
Posted by lectus on Mon, 04 Feb 2013 20:40:53 GMT
View Forum Message <> Reply to Message

Why not

OArray
OIndex
OVectorMap
OArrayMap

where O stands for Order?

Subject: Re: New containers - naming
Posted by mdelfede on Tue, 05 Feb 2013 21:35:33 GMT
View Forum Message <> Reply to Message

BSArray
BSIndex
BSVectorMap
BSArrayMap

Subject: Re: New containers - naming
Posted by zsolt on Tue, 05 Feb 2013 23:10:59 GMT
View Forum Message <> Reply to Message

I dont like this BS thing

InsVector or InVector would be good names. They could mean "Insert optimized".

Subject: Re: New containers - naming
Posted by mirek on Wed, 06 Feb 2013 06:56:14 GMT
View Forum Message <> Reply to Message

zsolt wrote on Tue, 05 February 2013 18:10I dont like this BS thing

InsVector or InVector would be good names. They could mean "Insert optimized".

I guess InVector/InArray are settled.

What I am not sure is Index/*Map names.

Ok, as Order<T> has no fans, let us be more explicit:

SortedIndex<T, Less>
SortedArrayIndex<T, Less>
SortedVectorMap<K, V, Less>
SortedArrayMap<K, V, Less>

Subject: Re: New containers - naming
Posted by mirek on Wed, 06 Feb 2013 06:58:52 GMT
View Forum Message <> Reply to Message

Didier wrote on Mon, 04 February 2013 14:10Hi Mirek,

great job as usual

For the naming I think that it must be consistent for all new containers so that developers can easily identify just by reading:
instead of :
Quote:InArray (Array counterpart)
Order (Index counterpart)
OrderVector (VectorMap counterpart)
OrderArray (ArrayMap counterpart)

I would rather put:
OrderArray (Array counterpart)
OrderIndex (Index counterpart)

OrderVector (VectorMap counterpart)
OrderArray (ArrayMap counterpart)

The prefix should always be the same: 'order', 'in' or anything else but I think it should be as explicit as possible so 'order' seems correct to me

Please note that InArray/InVector are fundamentally different, they are not ordered, they are just vectors with fast insert.

Order is "std::set"-like container that is sorted at all times.

Subject: Re: New containers - naming
Posted by zsolt on Wed, 06 Feb 2013 10:36:35 GMT
View Forum Message <> Reply to Message

Quote:SortedIndex<T, Less>
SortedArrayIndex<T, Less>
SortedVectorMap<K, V, Less>
SortedArrayMap<K, V, Less>

These would be good names, I think.
