Subject: Re: A new container in works: Flex - fast insertion vector
Posted by sergei on Fri, 21 Sep 2007 13:17:48 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 21 September 2007 15:06Quote:
Sorting would beat it for a static set. But would it, if the set was changing (adding/removing elements), and at any given time I might need the order? Add/remove is O(logn), iteration should be linear, and sorting would cause the iteration to become O(nlogn) (sort+iterate), unless add/remove maintains sort order. And even in the best case of flex, add/remove would be O(sqrtn), worse than O(logn), right?

That is correct,

- there is a bit more about O - something like "generic speed". Flex seems to be faster up to 160 elements (in fact, even sorted U++ Vector is faster there than std::set).

- I believe I do a lot of programming, yet I never had encountered a problem that would require continual sorted iteration.

(BTW, I have another idea of container, something much more traditional, basically sorted variant of Index based on tree implemention; with method GetNextSorted that returns the index of next element in sequence. Given my experience with Index v.s. node based hashmap, I dare to say that such linearized tree implementation will be faster than node based.)

Sorted Vector - that's sort and add sorted, or sort on every change?

As for a problem, out of my head: graph with set of elements in each node, elements move between nodes during the algorithm, output result (with sorted nodes) on each step of algorithm. There probably are other cases too.

Index with linearized tree is like heap? That would be pretty good (Index should have good "unique" time). Having a map flavor of such a container would be even better.