stl map and unordered_map difference

See more: C++ Linux

Rate this: ☆☆☆☆☆

i want to understand that major difference between map and unordered_map in C++.

which one i suppose to use and which one is best?

Posted 14-Nov-13 20:05pm
ranjithkumar81 1.3K

Add a Solution

4 solutions

Top Rated Most Recent

Solution 1

Rate this: ★★★★★

An ordered map is usually implemented on the basis of an ordered tree, for example red-black-tree in STL's case. Tree manipulations are relatively expensive, but the tree uses very little storage.

An unordered map is usually implemented as a hash table; the newer STL implementation offer such a map as well. Hash table are very fast, but have some storage overhead and can only grew to a certain number of nodes before they become inefficient.

So if you have a set of values for which you can estimate the maximum number of nodes and for which you don't need to iterate through the nodes in an ordered fashion, the unordered map is usually the best choice. In all other cases you better use an ordered (tree-based) map.

[AMENDMENT - A comment by pasztorpisti]

Depending on the scenario and the allocators in use trees can extremely fragment memory while a simple yet effective hashmap is only 2 arrays. Most of the time I use only hashmaps and a special kind of ordered map (built on top of an orderedset) that is an array that is always sorted. Depending on the frequency of different operations and the number of items trees may become the better solution but in my opinion that is relatively rare.

Posted 14-Nov-13 20:21pm **Edited** 17-Nov-13 4:24am v2
nv3 **32.6K**

Comments

CPallini at 15-Nov-13 3:19am

5.

pasztorpisti at 17-Nov-13 8:29am

+5, but be careful saying that "the tree uses very little storage". Depending on the scenario and the allocators in use trees can extremely fragment memory while a simple yet effective hashmap is only 2 arrays. Most of the time I use only hashmaps and a special kind of ordered map (built on top of an orderedset) that is an array that is always sorted. Depending on the frequency of different operations and the number of items trees may become the better solution but in my opinion that is relatively rare.

nv3 at 17-Nov-13 8:57am

I can fully agree what you said and have the same tendency of using hashmaps whenever possible. May I take you comment and add it literaly to the main entry?

pasztorpisti at 17-Nov-13 9:21am

Of course. :-)

Solution 2

Rate this:

What't the difference between using Google[^] or not?

Posted 14-Nov-13 21:21pm
CPallini **362.5K**

Comments

ranjithkumar81 at 15-Nov-13 3:47am

hello, i could not find the proper answer in Google as per my search.

Solution 3

Rate this: ★★★★★

Hi,
map:

- Usually implemented using red-black tree.
- Elements are sorted.
- Relatively small memory usage (doesn't need additional memory for the hash-table).
- Relatively fast lookup: O(log N).

unordered_map:

- Usually implemented using hash table.
- Elements are not sorted.
- Requires additional memory to keep the hash-table.
- Fast lookup O(1), but constant time depends on the hash-function which could be relatively slow.

refrence: (http://learninger.com[^])

Posted 16-Nov-13 21:28pm **Edited** 16-Nov-13 23:06pm v2
fkassaie 1.6K

Solution 4

Rate this: ★★★★★

None of these is "best". All of these data structures have their own characteristics. These characteristics are often described by listing the time complexity of the operations (insert, remove, add, ...) provided by the data structure. Here is a relatively brief yet good list of such a list: http://stackoverflow.com/questions/7294634/what-are-the-time-complexities-of-various-data-structures[^]

Each data structure has its own advantages and disadvantages, there is no perfect. Which data structure to use? Analyze the activity of your program, which operations does it use often? Choose a data structure whose advantages include most of the operations you often use. It will have a few slow operations but you know that you won't use those often...

Sometimes reality is different than what you would expect. Often data structures built from arrays are much faster if constructed/organized in a smart way because of better locality / cache-friendliness but this is also a hardware dependent thing (I referred to today's average desktop computers).

Posted 17-Nov-13 2:38am
pasztorpisti **37.9K**

Add your solution here

Preview

...

Existing Members

Sign in to your account

Your Email

Password

Forgot your password?

...or Join us

Download, Vote, Comment, Publish.

Your Email

Optional Password

I have read and agree to the Terms of Service and Privacy Policy

Please subscribe me to the CodeProject newsletters

Submit your solution!

When answering a question please:

1. Read the question carefully.
2. Understand that English isn't everyone's first language so be lenient of bad spelling and grammar.
3. If a question is poorly phrased then either ask for clarification, ignore it, or **edit the question** and fix the problem. Insults are not welcome.

Let's work to help developers, not make them feel stupid.