C++ map versus multimap for multi-key lookup

My data conceptually looks like:

```
"BLUE" :  (3 , 10,  15, 1220,  44040)
"RED"  :  (44, 523,  122143,  323233)
"BANANA" : (....)
```

Build-time is not important. For lookups where I have multiple keys, where I want to combine all the value lists, and sort them; should I represent this as a map or a multimap in C++ for the fastest outcome?

In other words, since the value vectors attached to the keys are variable length, should I have a map with key:vector, or multimap with key1:int1, key1:int2, etc?

The aim is to write a function where input = (key1, ..., keyN) and the output is a sorted list of all the values.

c++ map multimap

asked Jan 5 '12 at 21:50

Deniz
451 ● 8 ● 16

2 Answers

The `map<string, vector<int>>` solution is easier to understand and code against, and probably more efficient, space-wise, since you're arranging groups of values into contiguous storage rather than creating a node for each value. For the same reason, it's probably also more efficient algorithmically.

answered Jan 5 '12 at 21:55

Marcelo Cantos
100k ● 16 ● 198 ● 261

If you're going to be populating only once before doing any lookup, the best way is a vector that you sort after it's populated: `vector<pair<string, int> >`.

answered Jan 5 '12 at 22:03

Mark Ransom
152k ● 13 ● 134 ● 319

Thank you for your answer, could you provide some more detail please? Populating map or multimap? Sorting the string-int pairs of the multimap? I'm a bit confused. – Deniz Jan 6 '12 at 7:52