Subject: Re: what about VectorBiMap / ArrayBiMap ?
Posted by kohait00 on Fri, 30 Apr 2010 14:32:12 GMT
View Forum Message <> Reply to Message

@Mirek:

direction is meant to be key -> value, or value -> key;

say i have a pair of Strings, in a capabale VectorBiMap,
and add "MyKey" | "MyValue" pair (here spoken as key/value), then i will be able to find location of "MyValue" *VIA HASH* of "MyKey", what is trivial and possible with VectorMap.

but what if i want to be able to find location of "MyKey" (which is same location of "MyValue") *VIA HASH* of "MyValue"?

this is currently not possible, except manually using 2 Indexes, separately driving their api together. But is quite cool.

usecase: (simple and stupid) a dictionary 2 languages with exactly a pair of words like "Buenos Dias" <=> "Hi" (stupid)
and a *HIGH* speed translation in *BOTH* directions needed.

another usecase is a communication protocol translator, which depending on strings sends special values, and answers with values, which map to strings again.

this would need *2* Indexes with same stuff in it but swaped K-T

i had some ideas like the following:

```
template<class K, class T, class HashFnK = StdHash<K>, class HashFnT = StdHash<T>>
class VectorBiMap
 : public Index<K, HashFnK>
 , public Index<T, HashFnT>
{

 //following the api
//find T with K hash
//find K with T hash
};

template<class K, class T, class HashFnK = StdHash<K>, class HashFnT = StdHash<T>>
class VectorBiMap2
 : public MoveableAndDeepCopyOption<VectorBiMap2<K, T, HashFnK, HashFnT> >
 , public AMap< K, T, Vector<T>, HashFnK >
{
 typedef AMap< K, T, Vector<T>, HashFnK > B;
 HashBase hash2;
```

```
 HashFnT hashfn2;

 //following the api
//find T with K hash
//find K with T hash
};
```

i took a look in to implementation and it looks like there is no way except for coding an additional Interface, AIndex is not suitable AMap neither, but its a mix of both

PITFALL: all hash containers of Upp allow hash collision (handled internally with linked list). for a bijectional relation, this is not allowed ! or to define this to be a right unique relation (surrection)

i'll try it..any hints/comments/evaluations on the idea welcome

PS: in case of using Index, this would imply to use non const operator[] there like dicsribed above i think

---