Ordered hash map: search tree optimized by a hash table

Petar Ivanov
Valentina Dyankova
Biserka Yovcheva

Shumen University, Bulgaria

Use case: sorted databases with **few updates** vs **many finds**

Widely used approaches:

- **Hash map** – fast finds but **no ordering**!
- **Balanced search tree (BST)** – ordered but **slow finds**!

Solution:
wrap **BST + hash map**

Binary search tree (BST) for ordered data

Hash map for fast finds

Optimizations

- Keys **only** in BST (hash map of pointers)
- Values **only** in BST (hash map of pointers)
- Fast erase – marking as erased

Complexity

Asymptotic estimate	map	ordered hash map	ordered hash map* (fast erase)	unorderedmap
insert	logN	logN	logN*	**1**
find	logN	**1**	**1**	**1**
erase	logN	logN	**1**	**1**
next/prev traversal	**logN**	**logN**	**logN***	n/a or N
ordered traversal	**N**	**N**	**N***	n/a or NlogN

N* – # of ever inserted elements

Experiments

[sec]	map	ordered hash map	ordered hash map*	unorderedmap
insert	2.68	3.60	3.63	**0.83**
find	2.67	**0.51**	**0.63**	**0.49**
erase	3.32	4.05	**0.94**	**0.91**
next/prev traversal	**0.21**	**0.22**	0.94	n/a

Applications

- Sensitive hashing and other locality-sensitive hashes
- All STL map/set uses for hashable elements

Conclusion

Open source C++ classes and experiments at

https://github.com/petar-ivanov/ordered-hash