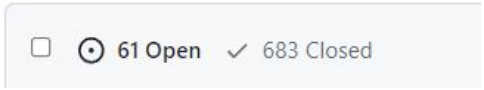
The case against binary trees

Stop using them.

Meeting
Feb 6th 2024

(But first!) **What's new in Dplug?**



back to a nice 1:10 ratio

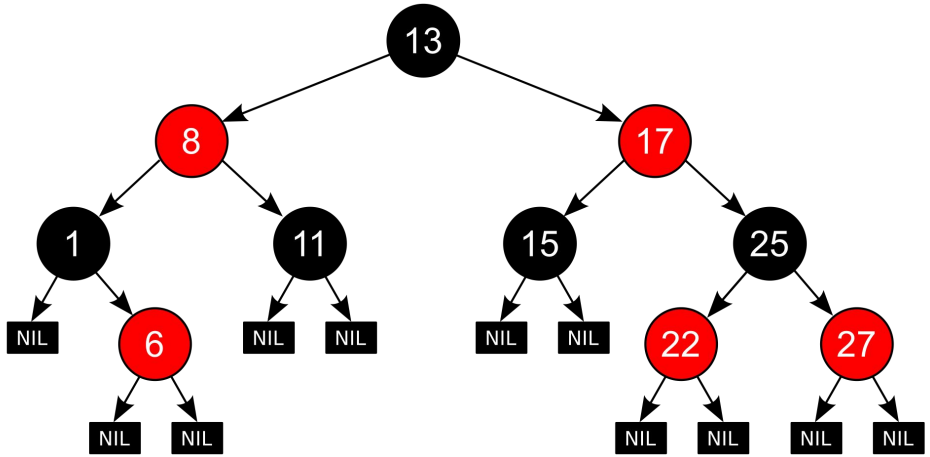
- Updated roadmap 2024 => https://github.com/AuburnSounds/Dplug/wiki/Roadmap

- Important macOS bug fixes (advice: move to v14.3.0+)
  https://github.com/AuburnSounds/Dplug/issues/835
  https://github.com/AuburnSounds/Dplug/issues/741

- More D talks at DConf online:
  https://dconf.org/2024/online/index.html

Dplug v14.2 uses a modified **Red-Black tree** from Phobos, for Map and Set.

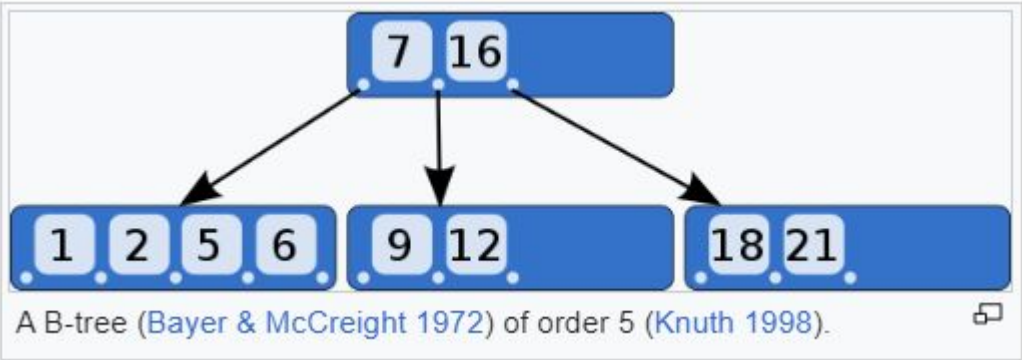
One item per Node.



Reference: https://en.wikipedia.org/wiki/Red%E2%80%93black_tree

Dplug v14.3 uses instead a **B-Tree** of order 32.

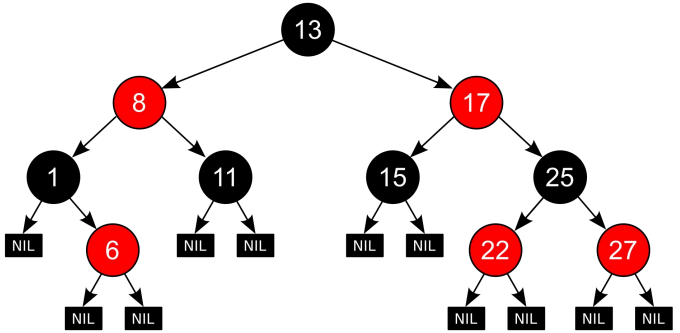
Each node can contain
31 items instead of just one.

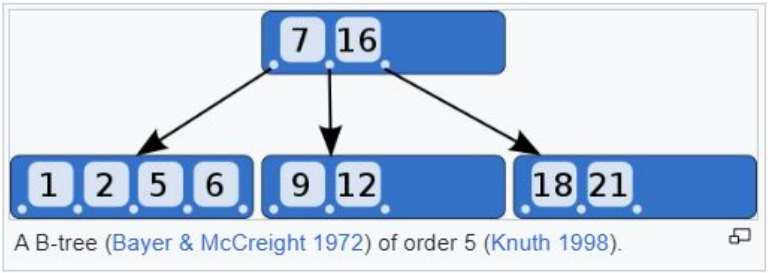


Reference: https://en.wikipedia.org/wiki/B-tree

Common points:

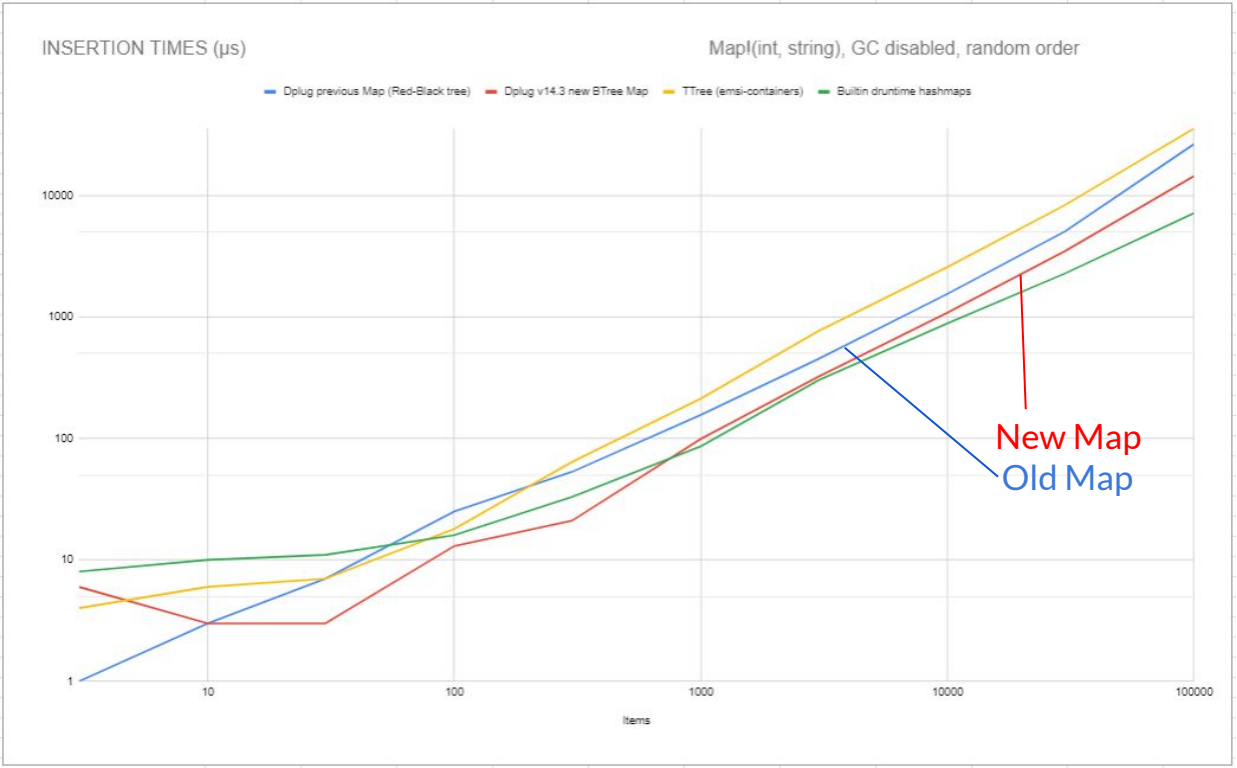
- O(log(N))
  - insertion,
  - deletion
  - look-up
- Always balanced
- A basis for std::map
  - std::set, etc.





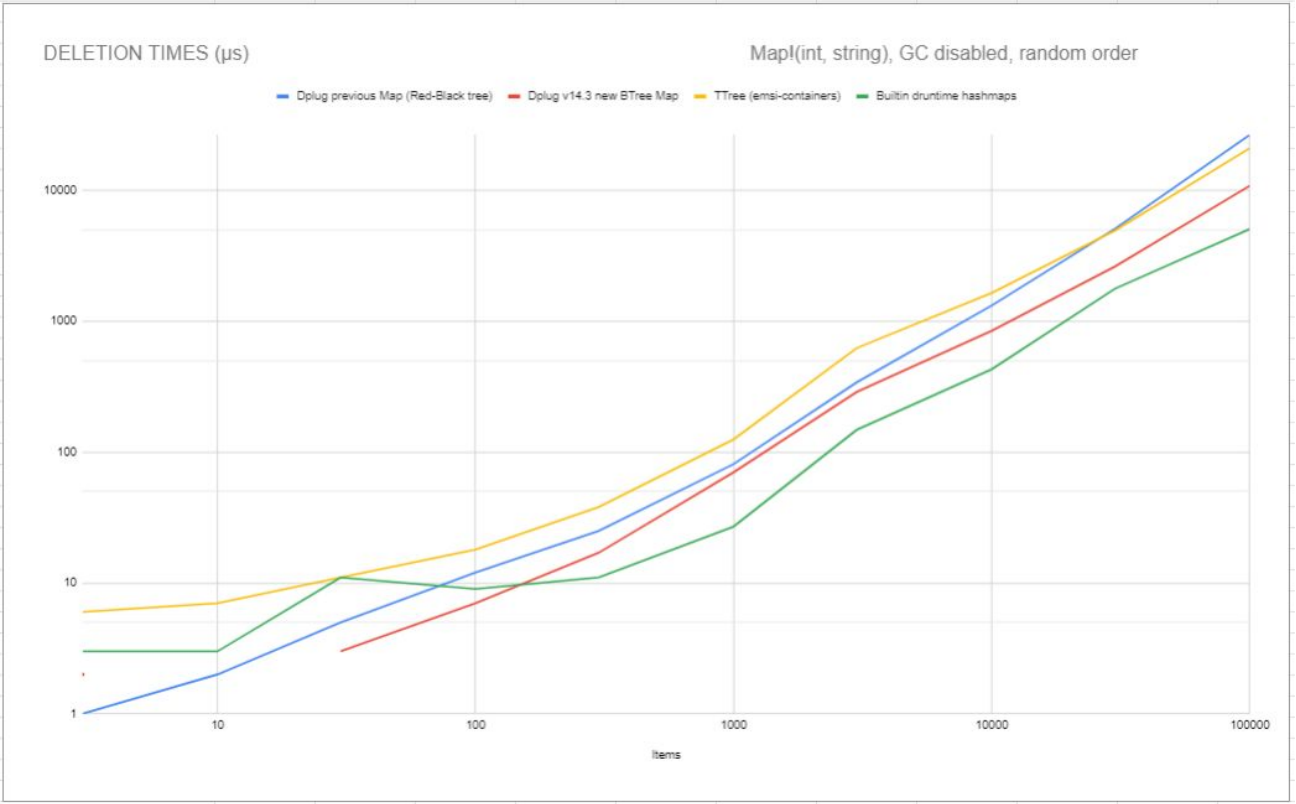
Nodes allocated with malloc in Dplug.

Some numbers



Insertion

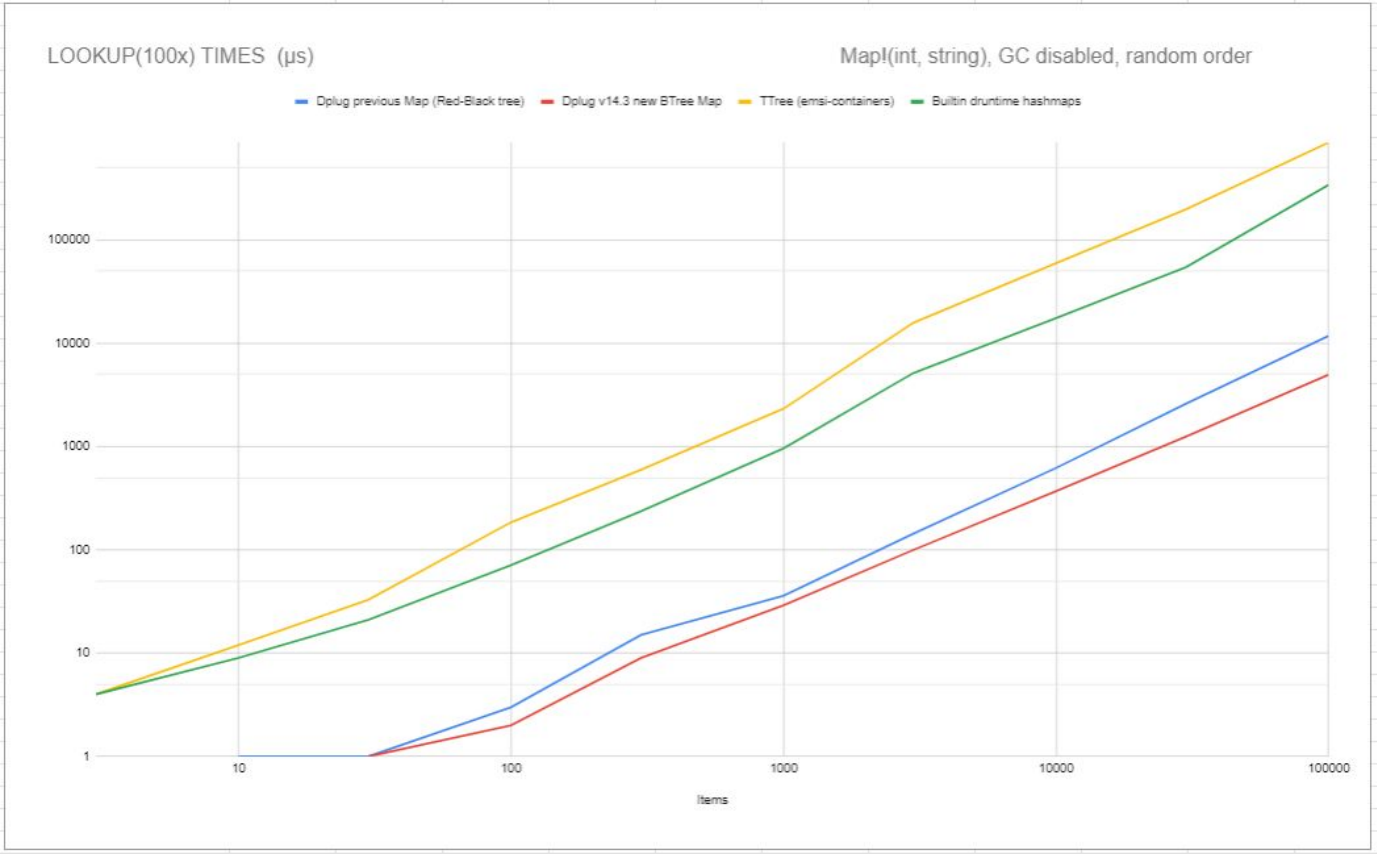
- Largely dominated by allocator times.
- GC.new faster than malloc, hence builtin hashmaps grow the fastest.
- BTree order 32 wins about **2x** over binary Red Black Tree. => less allocations.



Deletion

- Again, largely dominated by deallocation times.
- Test disables GC, so builtin hashmap release is fastest.
- BTree order 32 wins about **2x** over binary Red Black Tree, again.
  => less deallocations, less rebalance.

Deletion and Insertion have similar timings in hashmaps, B-Tree, Red-Black tree, and T-Tree.



Lookup

- builtin hashmaps get obliterated, probably because of the hash.
- Note sure what EMSI's **T-Tree** is doing here, it's 100x slower than Dplug. Must be a bug.
- BTree order 32 wins about **2x** over binary Red Black Tree, again.
  => because cache-friendly

Performance data: **look-up is about 200x faster than insertion or deletion.**

Conclusion

- **More than 2 children-per-node trounces 2 children-per-node.**
  - Same for binary heap vs n-heap
- Binary Red-Black trees are ubiquitous, yet slower than vanilla B-Tree.
- …and that's before we get to more optimizations: B+Tree, B*Tree, tuned order…
- Insertion and Deletion can be further optimized with specialized memory pool. The allocator is the bottleneck for say, parsing JSON values and putting them in a map.
- Put more children in your tree nodes if you can.
- B-Tree available in Dplug v14.3, nothing to do as user.
  Text-rendering will be a bit faster.