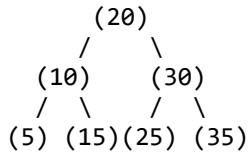


Copilot

Sure! Here are visual representations of the structures of `map` and `unordered_map` in C++:

`map`

A `map` in C++ is typically implemented as a Red-Black Tree, which is a type of balanced binary search tree. Here's a simplified visual representation:



In this tree:

- Each node contains a key-value pair.
- The tree is balanced, ensuring that operations like insertion, deletion, and lookup have a time complexity of $O(\log n)$.

`unordered_map`

An `unordered_map` in C++ is implemented using a hash table. Here's a simplified visual representation:

Index 0: (10, [1, 2, 3]) -> (20, [4, 5, 6])
 Index 1: (30, [7, 8, 9])
 Index 2: (40, [10, 11, 12])

In this hash table:

- Each index in the array corresponds to a bucket.
- Each bucket contains a linked list of key-value pairs that hash to the same index.
- The average time complexity for operations is $O(1)$, but it can degrade to $O(n)$ in the worst case due to hash collisions.

These visualizations should help you understand the underlying structures of `map` and `unordered_map`. If you have any more questions or need further clarification, feel free to ask!