**std::map**

Self-balancing BST 🡪 Guaranteed O(log(n)) when find(), insert(), erase()

std::map uses std::pair 🡪 Sort the keys

Example - Domain Name Lookup

* Text

  Description automatically generatedKey: domain name as an std::string
* Value: IP address as an int
* std::pair <std::string, int>

1. **Creating and using std::map**

Graphical user interface, text, application, email

Description automatically generated

1. **Searching std::map**

dns\_lookup\_map.find (<key>)

Returns an iterator to the pair (returns dns\_lookup\_map.end() if not found)

We can **then** access the key and value of the pair by through the iterator

Graphical user interface, text, application

Description automatically generated

1. **Iterating over std::map**

Since the map is a balanced BST, **std::map iterator advances as an in-order traversal** (incrementing keys for us automatically)

Diagram

Description automatically generated

1. **Duplicate Keys**

Maps **DO NOT SUPPORT DUPLICATE KEYS**. If we try to insert a duplicate key, we the compiler will basically **ignore** the new one.

There is also something calledMultiMap that can handle duplicate keys (???)

Table

Description automatically generated

Graphical user interface

Description automatically generated with low confidence