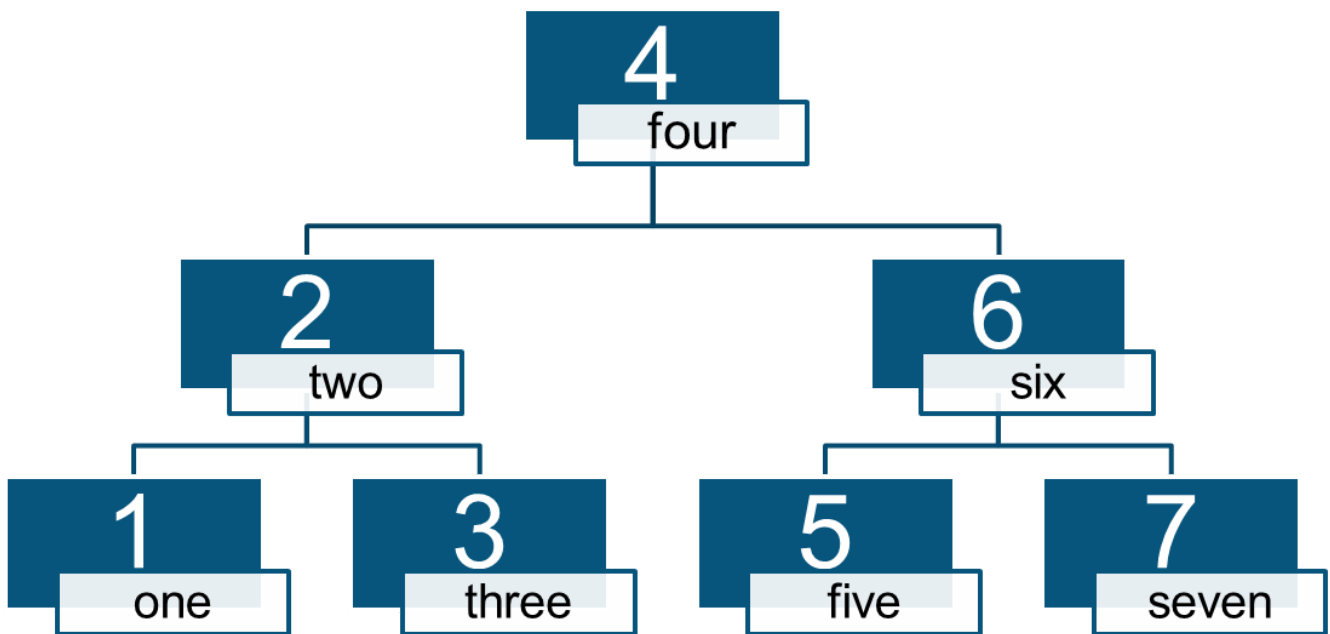


Maps

Now, we shall look at the features of `std::map` which make it such a popular container.



`std::map` is the by far the most frequently used associative container. The reason is simple. It combines a often sufficient enough [performance](#) with a very convenient interface. You can access its elements via the index operator. If the key doesn't exist, `std::map` creates a key/value pair. For the value, the default constructor is used.

🔑 **Consider `std::map` as a generalization of `std::vector`**

Often `std::map` is called an associative array. Array, because `std::map` supports the index operator like a sequential container. The subtle difference is that its index is not restricted to a number like in the case of

difference is that its index is not restricted to a number like in the case of `std::vector`. Its index can be almost any arbitrary type.

The same observations hold for its namesake `std::unordered_map`.

In addition to the index operator, `std::map` supports the `at` method. The access via `at` is checked. So if the request key doesn't exist in the `std::map`, a `std::out_of_range` exception is thrown.