SETS

Explanation: Sets are a fundamental data structure used to store unique elements and allow for efficient membership testing. In C++, sets are implemented using red-black trees, providing ordered storage and logarithmic time complexity for insertion, deletion, and search operations.

Key Points:

- Sets do not store duplicate elements; inserting the same element multiple times results in only one instance.
- Useful for removing duplicates from a collection of elements.
- Created using set<datatype> in C++.
- Two versions: set (ordered) and unordered_set (unordered, uses hash tables).
- Operations like insertion, deletion, and search have a time complexity of (O(\log n)) for set and (O(1)) on average for unordered_set.
- Membership testing can be done using count method, which returns 0 or 1.

MAPS

Explanation: Maps are an extension of sets that store key-value pairs, where each key is unique and associated with a specific value. In C++, maps are implemented using red-black trees, ensuring ordered storage and logarithmic time complexity for operations.

Key Points:

- Keys in a map are unique, and each key is associated with a value.
- Created using map<key_type, value_type> in C++.
- Operations like insertion, deletion, and search have a time complexity of (O(\log n)).

- Values are accessed using the index operator [].
- Membership testing can be done using count method, which returns 0 or 1.
- Iteration through a map yields key-value pairs in sorted order based on the key.

UNORDERED SETS

Explanation: Unordered sets are similar to sets but do not maintain any order among elements. They use hash tables for storage, providing average constant time complexity for insertion, deletion, and search operations.

Key Points:

- Unordered sets do not store duplicate elements.
- Created using unordered_set<datatype> in C++.
- Operations like insertion, deletion, and search have an average time complexity of (O(1)).
- Membership testing can be done using count method, which returns 0 or 1.
- Iteration through an unordered set does not guarantee any specific order of elements.
- Preferred over set when ordering is not required due to faster operations.

UNORDERED MAPS

Explanation: Unordered maps store key-value pairs without maintaining any order among elements. They use hash tables for storage, providing average constant time complexity for insertion, deletion, and search operations.

Key Points:

- Keys in an unordered map are unique, and each key is associated with a value.
- Created using unordered_map<key_type, value_type> in C++.
- Operations like insertion, deletion, and search have an average time complexity of (O(1)).
- Values are accessed using the index operator [].
- Membership testing can be done using count method, which returns 0 or 1.
- Iteration through an unordered map yields key-value pairs in any order.
- Preferred over map when ordering is not required due to faster operations.