HASHING DATA STRUCTURE

- #include < unordered_set >
- An unordered_set is implemented using a hash table
- where keys are hashed into indices of a hash table so that the insertion is always randomised
- All operations on the unordered_set takes constant time O(1) on an average

  which can go up to linear time O(n) in worst case

  which depends on the internally used hash function, but practically they perform very well and generally provide a constant time lookup operation.

Hello World

HASHING DATA STRUCTURE

- Set vs unordered_set

- Set -> key are stored in ordered fashion

  unordered_set -> keys are stored in unordered fashion

- Set is internally implemented as RED BLACK Tree

  unordered_set is internally implemented HASHING

- Set operation -> Time complexity O(log n)

  unordered_set operation -> Time complexity O(1)

Hello World

HASHING DATA STRUCTURE

- #include< unordered_set >

- FUNCTIONS

insert() unordered_set doest not have duplicates key

size()

clear()

begin()

end() auto it = s.find(key);

find() s.erase(key) s.erase(it)

erase()

count() It is substitute of find () function

find () function return the iterator to that element

count () function return the ‘1’ if it is present or ‘0’ if not

Hello World