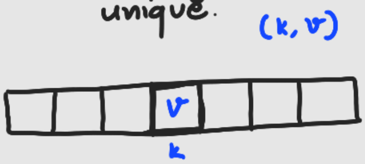
Hashmaps CHEATSHEET

Hashmaps typically talk about key-value pairs to store data.
* all keys are unique.

→ Can be visualized as an array
BUT each index need not be an integer

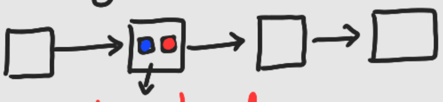
a [key] = value



Basic operations
1. insert (k key, v value)
2. getValue (k key)
3. delete key (k key)

Implementation?

1. Using Linked Lists



each node has three variables.
key, value and next pointer

Since linked list is used, all operations are O(N) → Drawback of using LL.

2. Using Balanced Binary Search Trees.

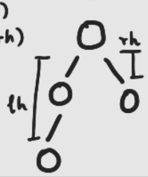
The implementation is INBUILT in C++
– All operations are O(log N)

Comes under the 'map' structures.
Here, all the keys are sorted.

* the functions for both maps and unordered-maps are same.

DIFFERENCE
maps → sorted keys
unordered-maps → not sorted keys

Balanced BST
left height (lh)
right height (rh)
|lh - rh| ≤ 1
for each node.



3. Using Hash tables
The implementation is also INBUILT in C++. – All operations are O(1)
Comes under the 'unordered-map' structures where the keys are NOT sorted.

→ maps — Balanced BST O(log N)

→ unordered-maps — Hash tables O(1)

Inbuilt Hashmaps.
(STL Library)

unordered_map < $\underbrace{\text{key, value}}_{\text{can be any type of variable.}}$ > m.

1. Insertion
Two ways.

→ using pairs

```
pair<string, int> p
    = new pair
        <string, int>
          ("abc, 2);

m.insert(p);
```

Create a 'pair' class and insert

```
template <typename T>
class Pair {
    T key,
    T value
    Pair <T key, Tvalue)
    {
      this→key = key
      this→value = value
    }
```

→ directly
`m [ "abc" ] = 2;` SIMPLER and EASIER

2. Accessing
Two ways
→ m.at("abc")
returns the value of key "abc"
If not present, results in ERROR

→ m ["abc"]
returns the value.
If not present, it creates a key "abc" with value as 0.

So. how to check if a key is present?
if m.count ("abc") > 0
then PRESENT

3. Deletion m.erase ("abc")

or m.erase() – to erase the entire map.

How to traverse the Maps?
use ITERATORS

Creating an iterator for a map

```
unordered-map < k, v> : : iterator it ;
it = m. begin( ) ;
```

Accessing keys and values:
it → first (KEY)
it → second (VALUE)

Templates
inbuilt structure that helps in assigning any data type
T data
T can be int, double etc.