CLASS 19

25 February 2023 14:04

CLASS-19

unordered_map <int, int> ma;

- insert()
- Find()
- erase()
- size()
- empty()
- count()

unique key

ma.find(1);

ma.end()

iterator

ma.count(1) = 1

= 0

key

No of element in map with key = 1

MAP → STL

map <int, int> ma

unordered map

→ Random values

Ordered_map → value sort acc to keys

{key, value}

unordered_map <int, int, hashfun>

Hash function

Hash function

→ Hash function → int, char, string

pair <int, int>

set

Red

vector<int>

Map

- size()
- clear()
- insert
- Find

{1, 2} {2, 3} {3, 4} {4, 1}

"abc"

→ a7 + a8 + a9

upper

ma.lower_bound(key)

bound()

lower_bound( )

↓

(n)

→ insert
→ find
→ erase
→ empty()
→ count()
→ begin()
→ end()
→ lower_bound()
→ upper_bound()

ma.lower_bound(...)

↳ iterator

2 (5)

ma.end()

key

sorted array

map → sort by keys
(default)
ascending

comparator

sort( ) → ascending
customize sort
comparator
↳ userdefined datatype

class Person
public:
{ int age;
string name;
}

Person vector<Person> V;

sort(V1.begin(), V1.end(), cmp)
→ int, string, characters, Pair

P1
P2
Pn

cmp

unordered_map<int, int, → customized sorting
HashFunction> → userdefined Datatype

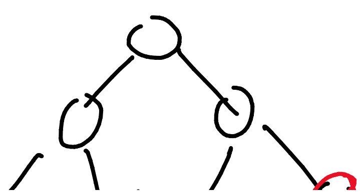
sort(, cmp)

bool cmp(int a, int b)
{ return a > b;
}

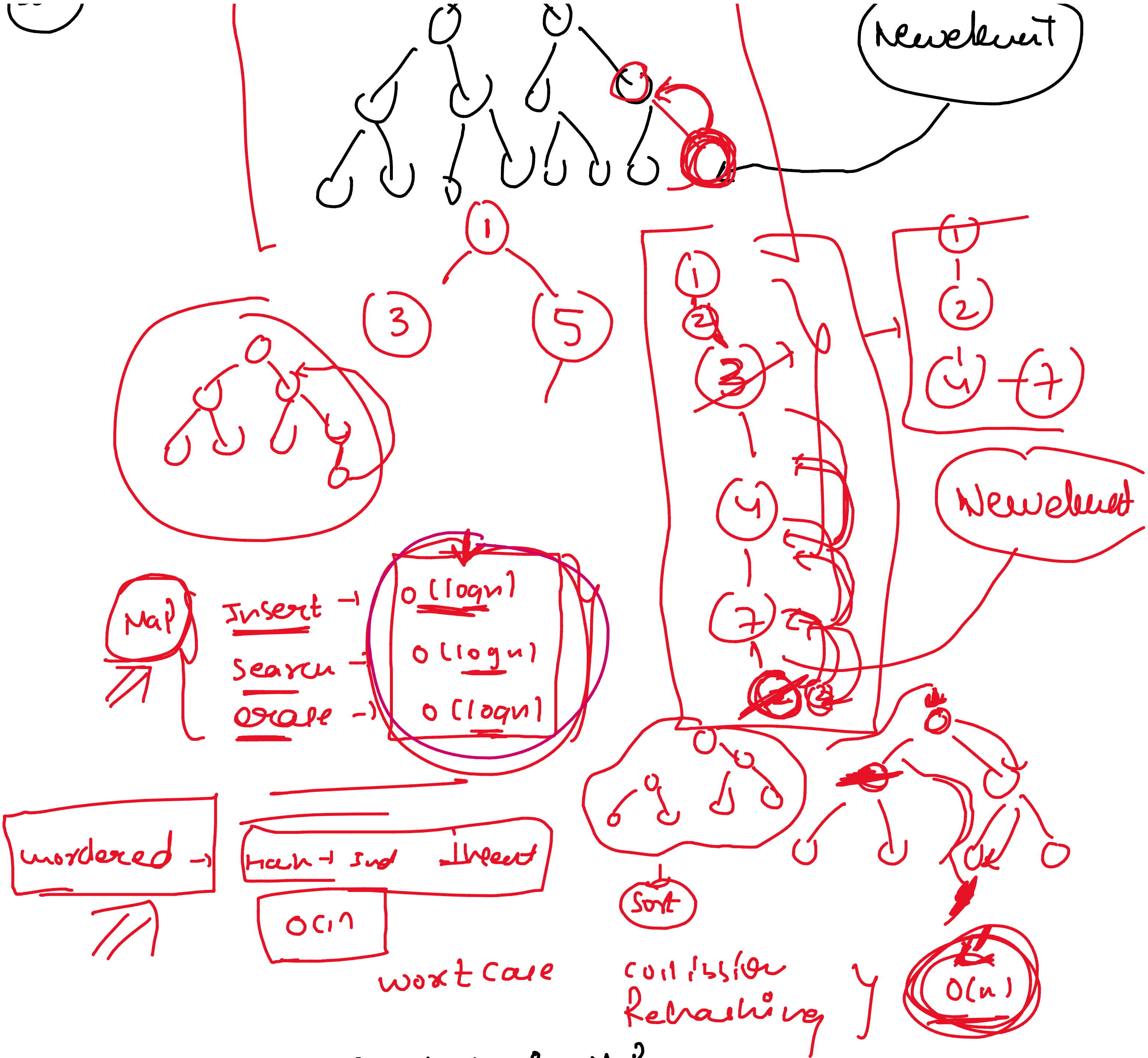
↓
Hashtable
randomly
idx

Map<int, int, cmp> ma
↳ Red Black Tree
↑↑
Self balancing BST

Sorted order



Newelement



Custom Comparators in Map

1. Function Pointer
2. Functors
3. lambda Expression

functors

bool ...( )

class comp {

descending sort

map → comparator

bool cmp (int a, int b)
{
return _
}

class comp {
public:

bool operator() (const int a, const int b)
{ return a>b;
}
}

---

function pointers

function < bool (int , int) > a = comp

map<int, int, function < bool (int , int) > > ma (a);

bool comp (int a , int b)
{ }

lambda

map

unique key, value

ma["mango"] = 100 ;

ma["mango"] = 50 ;

cout<< ma["mango"]

Multimap

→ same like map

unique keys

Multiple same keys value pair

{1, 2, 4}
{1, 3}

ma[1] x

map

ma[1] = 2

ma[1] = x

Sort

[ ] [ ]

multimap

1
2
3
(1)

- size()
- clear()
- empty()
- insert()
- erase()
- find()
- lb

map
ma[1], y

unordered multimap

same key
value
pair

unordered-map

key = value

Set

value = key = value

Map

{key, value}

map<int, int> ma

set<int> s

pair<int, int> string, char

- log(n), log(n), log(n)

- begin(); empty() find() upper-bound()
  end(); insert() clear()
  size(); erase() lower bound

set<int, cmp> s;

unique key

- Red Black Tree / AVL Tree

Balanced BST

↳ Red Black Tree / AVL Tree — Balanced BST

Map

- unordered
- ordered multimap — sorted
- unordered Multimap — unsorted

Set

- unordered set
- ordered multiset — sorted
- unordered multi — unsorted

Open Hashing

1. (Seprate chaining) → linked list
2. closed Hashing →
3. Double Hashing →

Value → Hash Fn → idx → Hash function → idx2

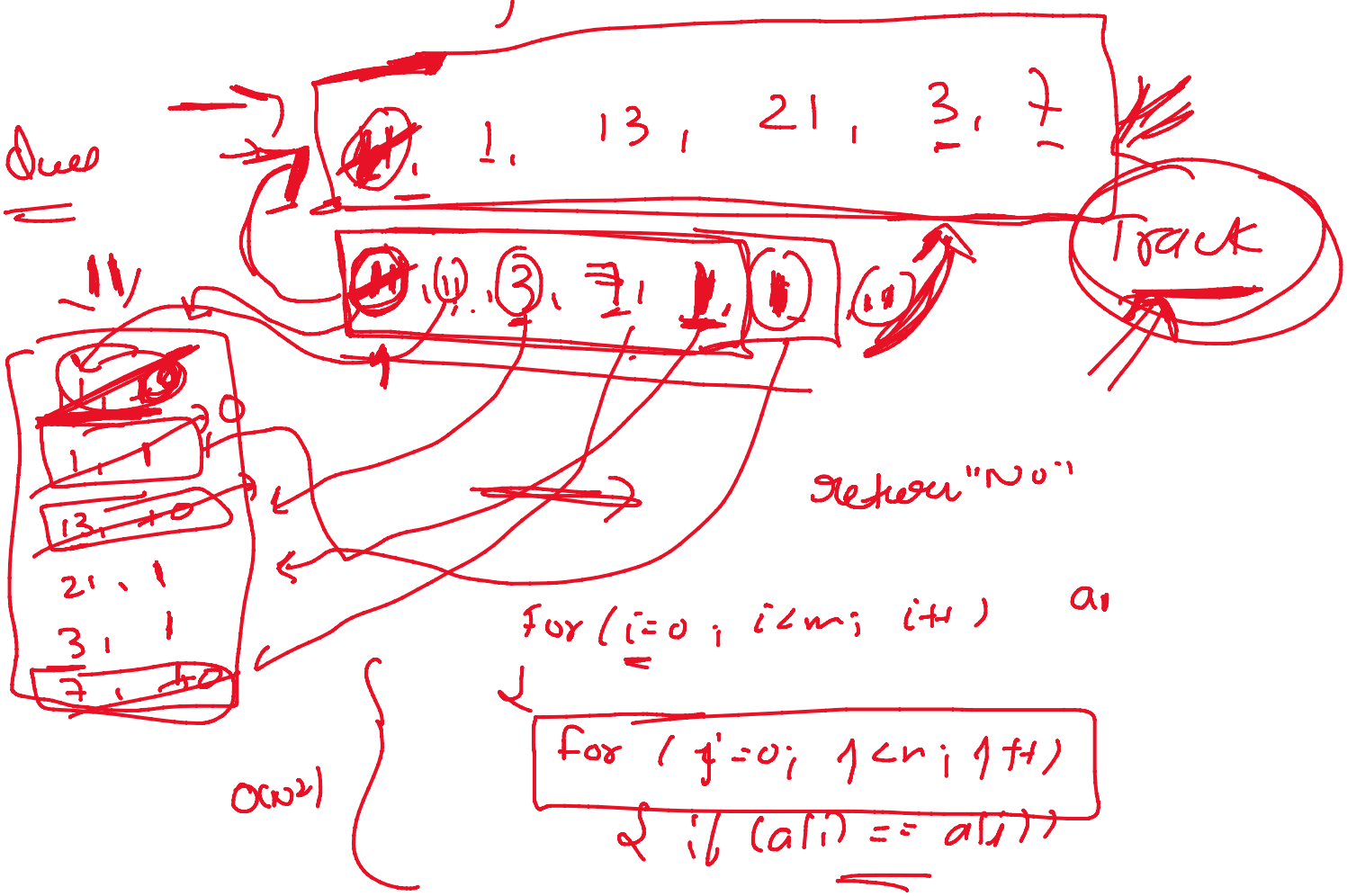
newvalue — Collision

linear Closed Hashing :-

Quadratic :- 1st, 4th, 9th - - - -

Map → Atlassian
Salesforce
Sprinklr

Maps intervals



for (i=0; i<m; i++) a

for (j=0; j<n; j++)
if (a[i] == a[j])

O(N²)