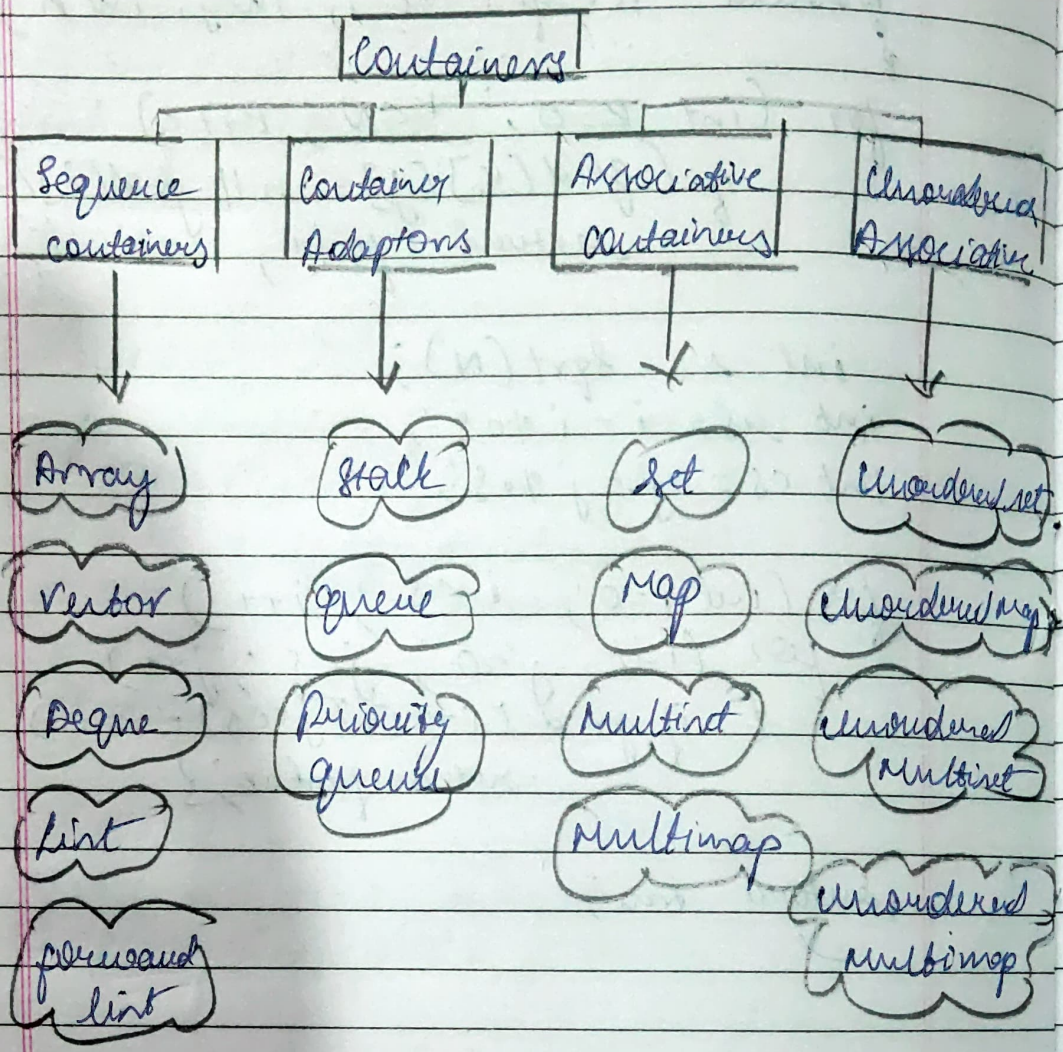
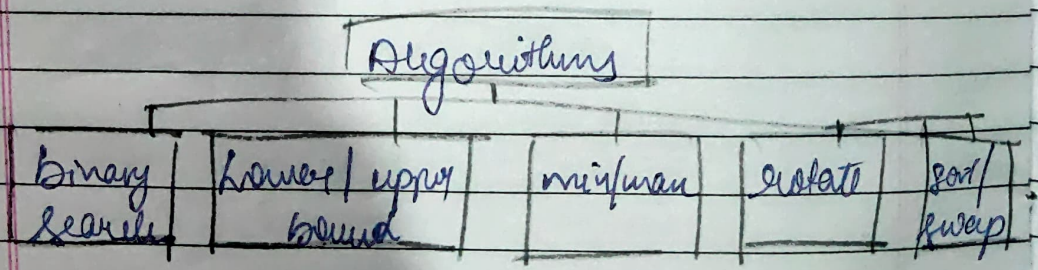
STL


Containers
Sequence containers
Container Adaptors
Associative containers
Unordered Associative
Array
Vector
Deque
List
forward list
stack
queue
Priority queue
Set
Map
Multiset
Multimap
Unordered set
Unordered map
Unordered Multiset
Unordered multimap


Algorithms
Binary Search
Lower / upper bound
min/max
rotate
sort / sweep

⇒ Array: (Static)

array<int, 4> a = {1, 2, 3, 4}

a.size() → 4

a.front() → 1

a.end() → 4

a.empty() → 0

a.at(2) → 3

⇒ vector: (Dynamic array)

vector<int> v

v.capacity() → Memory allocated.

v.push_back(1); → insert element

v.at(0) → 1

v.front() / v.back()

v.pop_back() → remove last element.

vector<int> a(5,1); → size of vector
→ initialize all elements with 1.

vector<int> last(a);

→ Deque

deque<int> d;

d.push_back(1);
d.push_front(2); 2, 1

d.pop_back();
d.pop_front();

d.at(1);

d.erase(range);

⇒ List

list<int> l;

l.push_back(1);
l.push_front(2);

=> Stack

stack <string> S; S.push( );

S.top( );

=> Queue:

queue <string> q;

q.push( );

=> Priority ~~Loop~~ queue

priority - queue <int> q; (max heap)

priority_queue <int, vector<int>,
greater <int> mini>
(min heap)

q.push(1);
q.push(3);
q.push(2);
q.push(0);

for (int i=0; i<q.size(); i++) {
cout << q.top(); q.pop();
}

→ **Set** : All elements are unique & sorted order

```
set <int> s;
s.insert(5);
s.erase(2);
s.count(5);   checks if 5 is present or not.
```

⇒ **map** : One key points to one value.

```
map<int, string> m;

m[1] = "Rajath";
m[2] = "K";
m[3] = "Hello";

m.count(-5) => checks if the key is present or not.

m.first (key), m.second (value)
```

→ Algorithms

* Binary search (Finding 5 in vector)

binary_search (v.begin(); v.end(), 5);

lower_bound (:, :, :);

upper_bound (:, :, :);

max (a, b);

min (a, b);

swap (a, b);

reverse (s.begin(), s.end());

rotate (v.begin(), v.begin()+1, v.end);

sort (v.begin(), v.end());

introsort (Quick, heap, insertion)