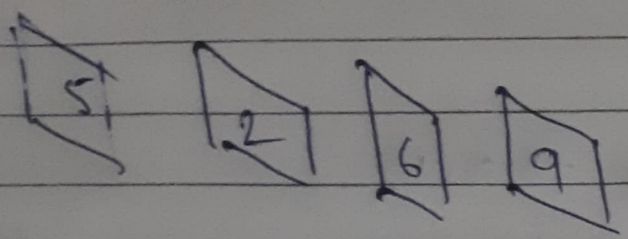
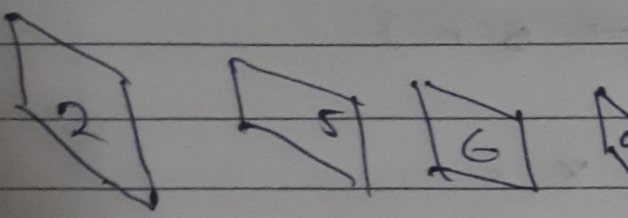
Insertion Sort:




: Card example.




: sorted.

arr[] = {4, 12, 11, 20}.

4 11 12 20 : sorted.

arr[]:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 10 | 1 | 7 | 4 | 8 | 2 | 11 |

round 1: 1 10 7 4 8 2 11

round 2: 1 7 10 4 8 2 11

round 3: 1 4 7 10 8 2 11

4: 1 4 7 8 10 2 11

5: 1 2 4 7 8 10 11

6: 1 2 4 7 8 10 11

Total round = (n-1)

why I.S.?

- Adaptable
- Stable
- n (small) performance is good.
  partially sorted performance is good.

Complexity

Space complexity → O(1)

Time complexity → $O(n^2)$

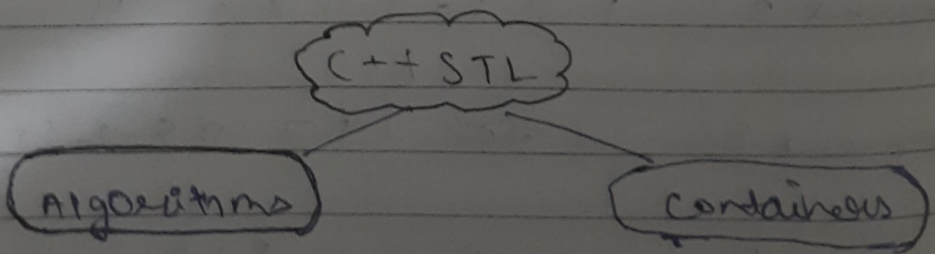
x x x x x x

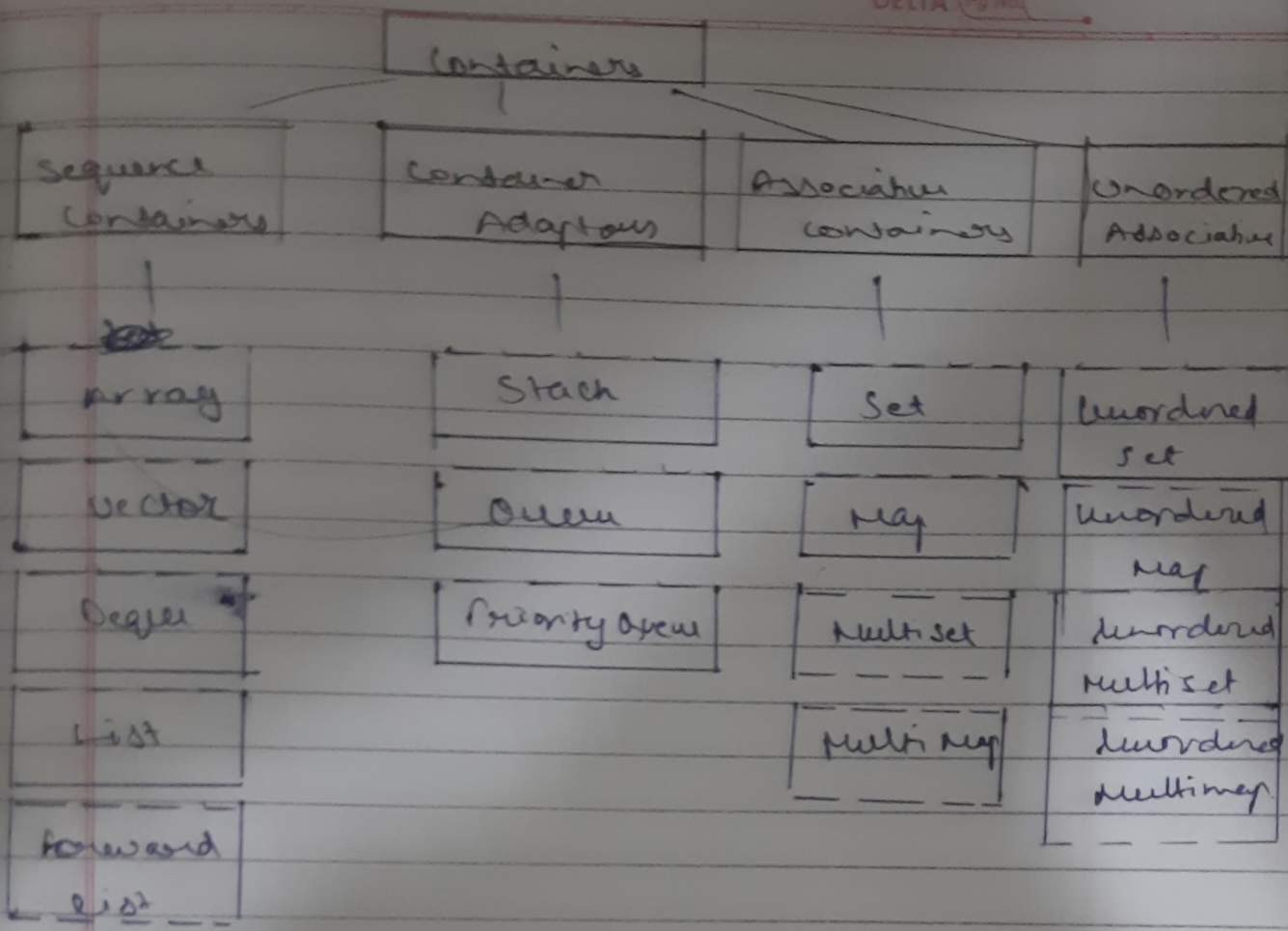
1st → 1' comparison

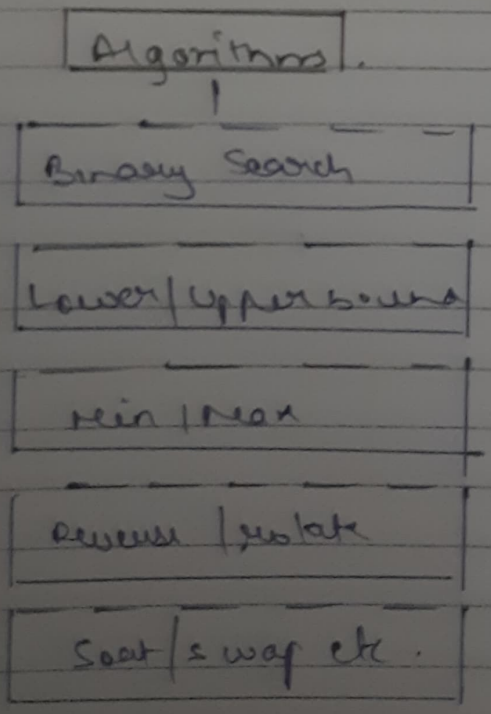
(n-1)

Best case → already sorted → O(n)

worst case → reverse '' → $O(n^2)$




Containers
Sequence containers
Container Adaptors
Associative containers
Unordered Associative
array
vector
Deque
List
forward list
Stack
Queue
Priority Queue
Set
Map
Multi set
Multi Map
Unordered set
Unordered Map
Unordered Multi set
Unordered Multimap


Algorithms
Binary Search
Lower/Upper bound
Min/Max
Reverse/rotate
Sort/swap etc.