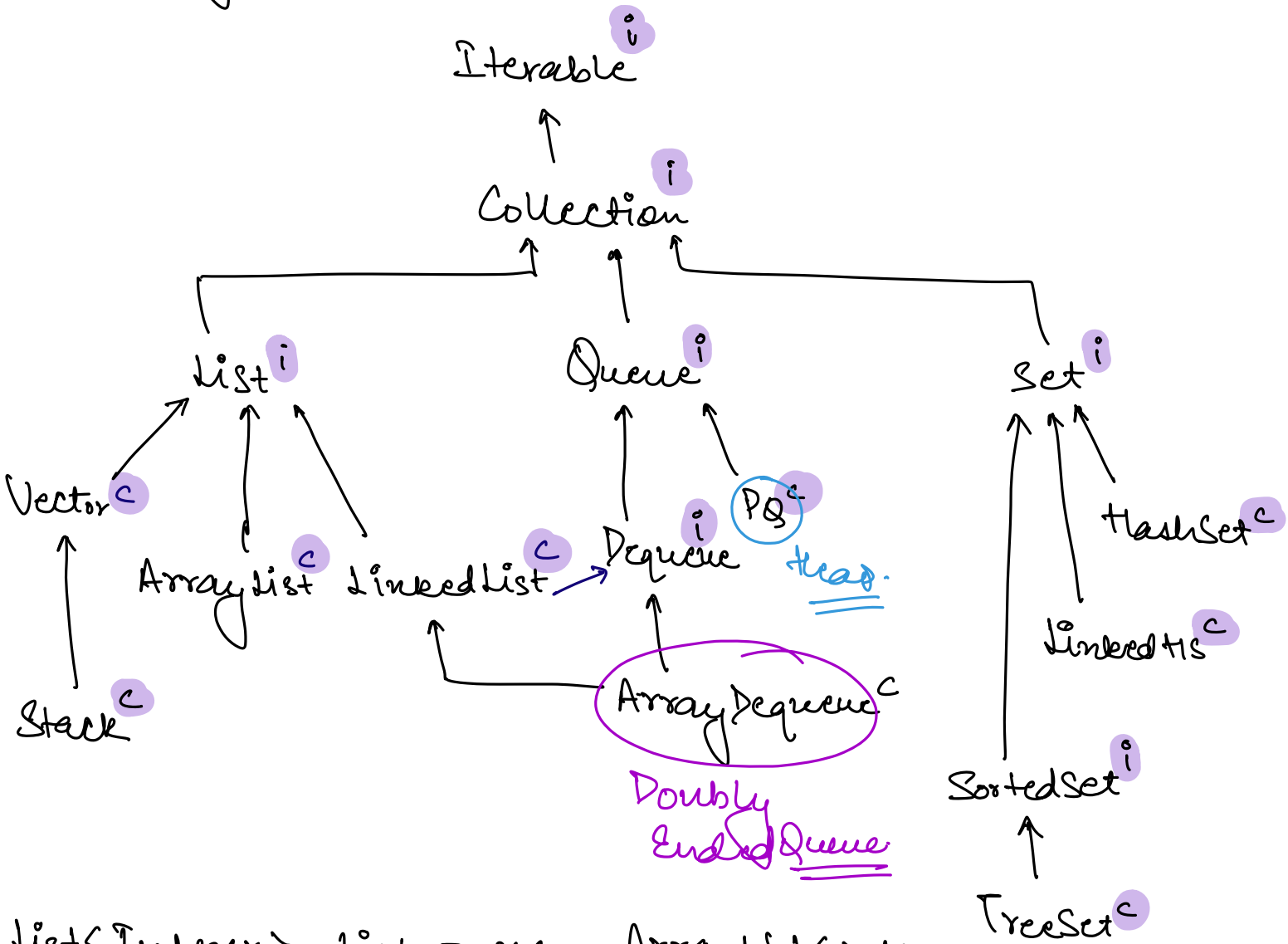
Agenda.

⇒ Collections framework

⇒ Inbuilt Data structures provided by JAVA.

List, Set, Map, PriorityQueue, Queue, Stack, ...

Hierarchy.



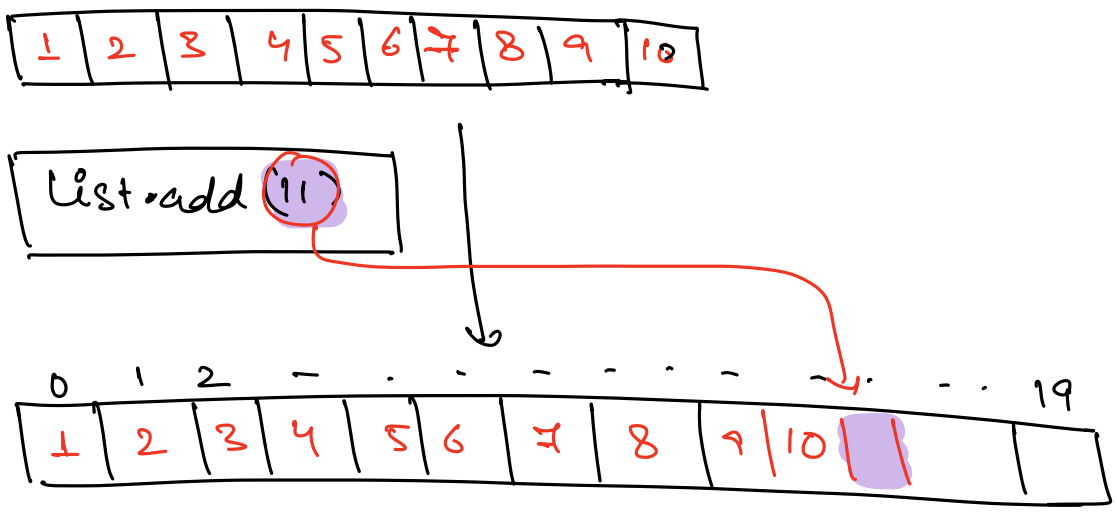

```
List<Integer> list = new ArrayList<>();
                   = new Vector<>()
                   = new LinkedList<>()
                   = new Stack<>();
```

ArrayList.

↳ Dynamic Array ⇒ Contigous Memory Loc^{n}.

Size=~~0~~ 10

Capacity=10



TC of add operation in AL: O(1) amortized.

Amortized: We do one costly operation to make the future operations optimal.

set(i, value) & add(i, value)

⇓ ⇓

O(1) O(N) {Shifting}.

ArrayList: Not thread safe.

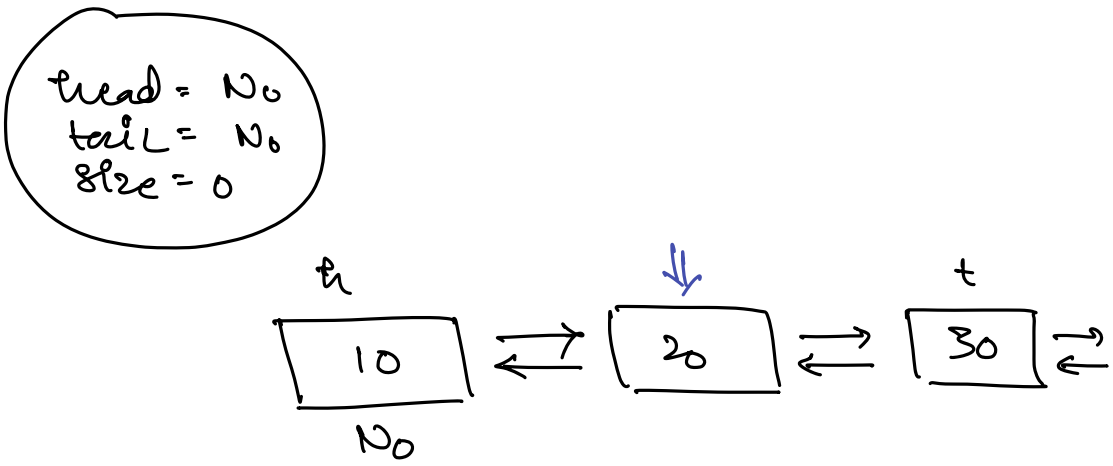
⇒ Vector : Synchronized | Thread-safe.

↳ Slow.

⇒ LinkedList.

↳ Non continous.

↳ Doubly linked list.



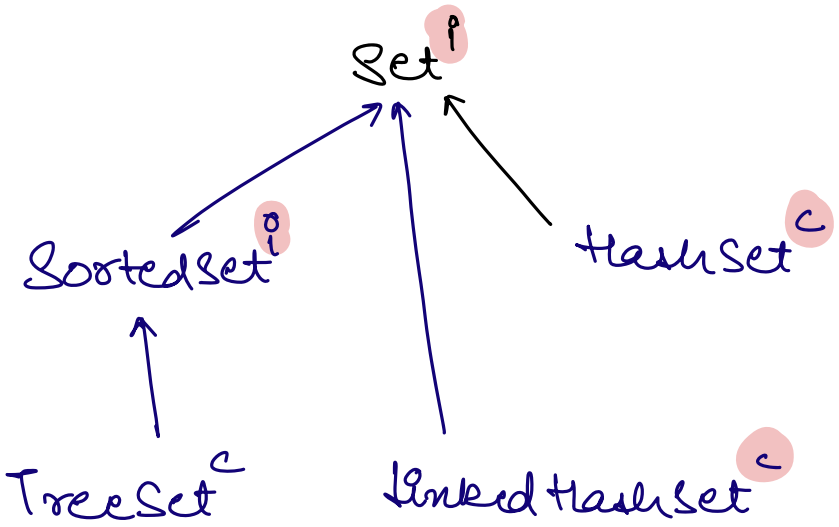
Stack

↳ Push(x) → add(x)

Pop()

Peek()

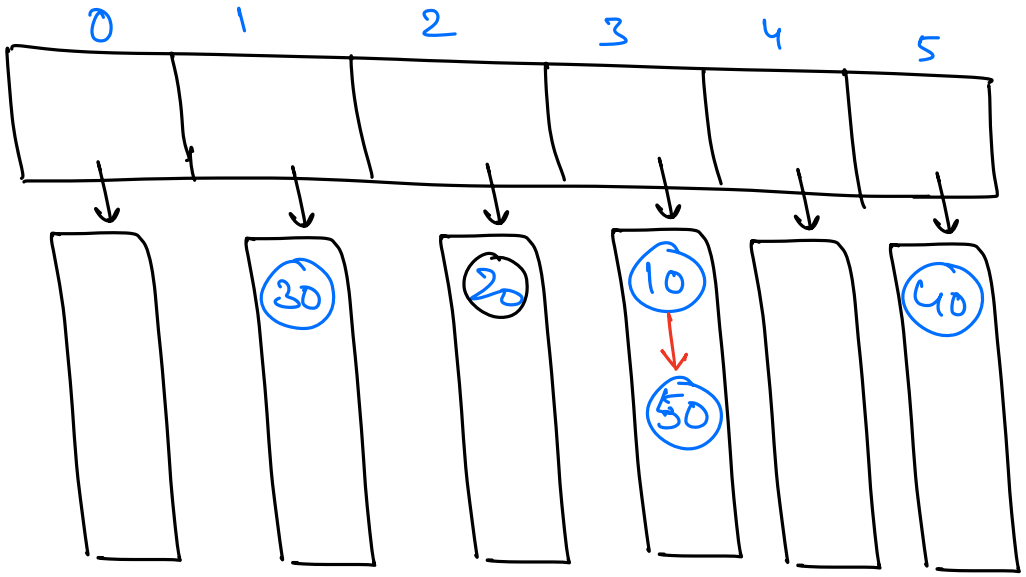
Set Interface.




```
Set<Integer> set = new HashSet<>();
```

```
Set.add(10)
Set.add(20)
Set.add(30)
Set.add(40)
Set.add(50)
```

Set.add(20)



add(x)
contains(x)
} O(1) Avg Complexity

⇒
```
add(int x) {
    int bucket = hashfun(x);
    bool found = false;
    for (Node node : A[bucket]) {
        if (node.value.equals(x)) {
            found = true;
            break;
        }
    }
    if (!found) {
        Node u = new Node(x)
        A[bucket].add(u);
    }
}
```
O(λ)

⇒ Contains
```
bool containt(x) {
    int bucket = hashfun(x);
    for (Node node : A[bucket]) {
        if (node.value.equals(x)) {
            return true;
        }
    }
    return false;
}
```
O(λ)

λ ⇒ Avg # of elements / bucket

load factor.

λ goes beyond the threshold value, Java increases the # of buckets.

↓

Rehashing.

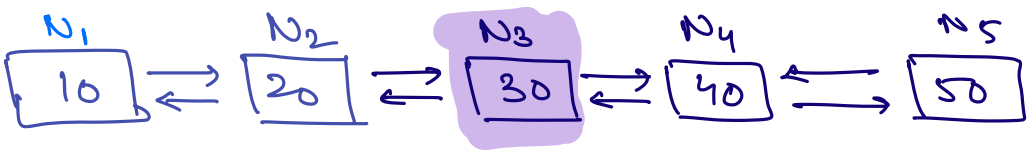
⇒ HashSet is unordered.

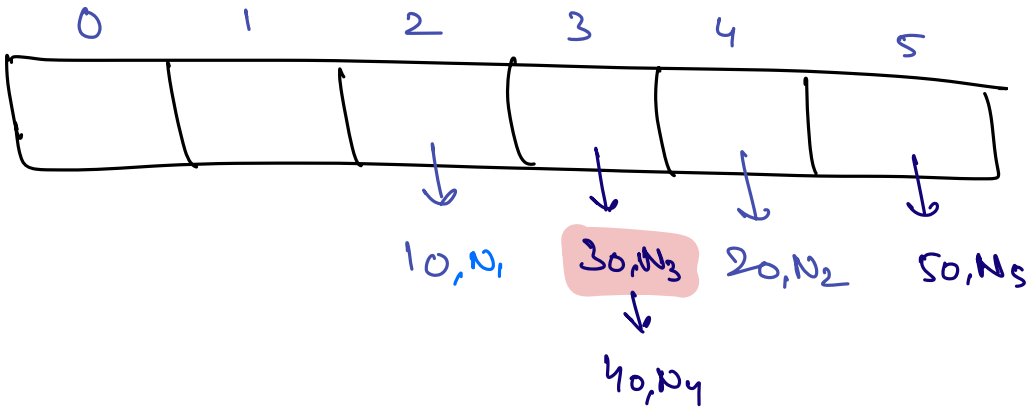
⇒ LinkedHashSet. ⇒ Maintains insertion order.

↳ Ordered.

```
Set<Int> lhs = new LHS<>();
lhs.add(10)
lhs.add(20)
lhs.add(30)
lhs.add(40)
lhs.add(50)
```

DLL





⇒ TreeSet → Balanced BST (RB Tree).

↳ Sorted Order.

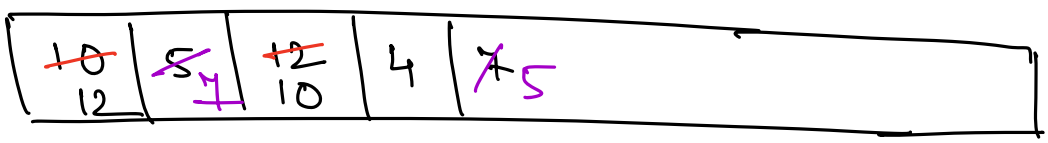
⇒ By default Treeset provides natural ordering, but using comparator we can override it.

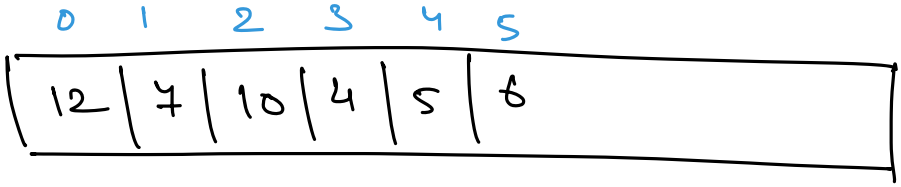
⇒ Priority Queue.

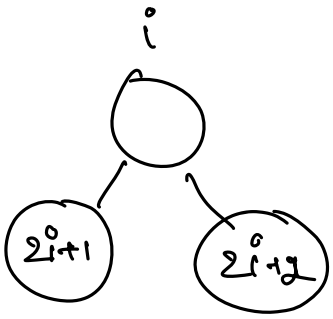
Max Heap.

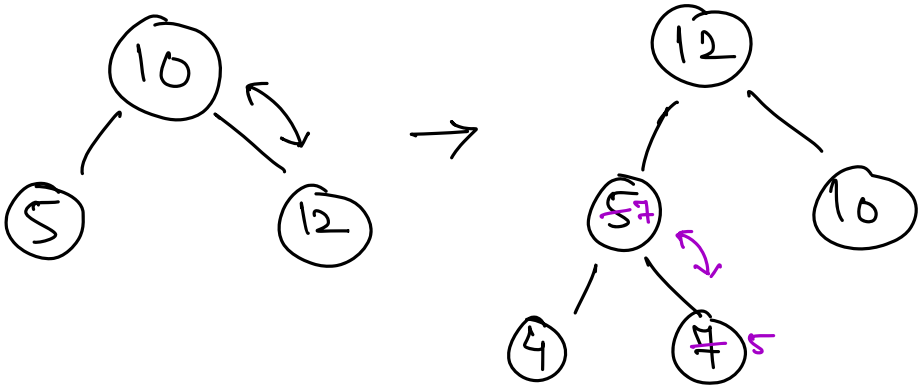
max_heapify

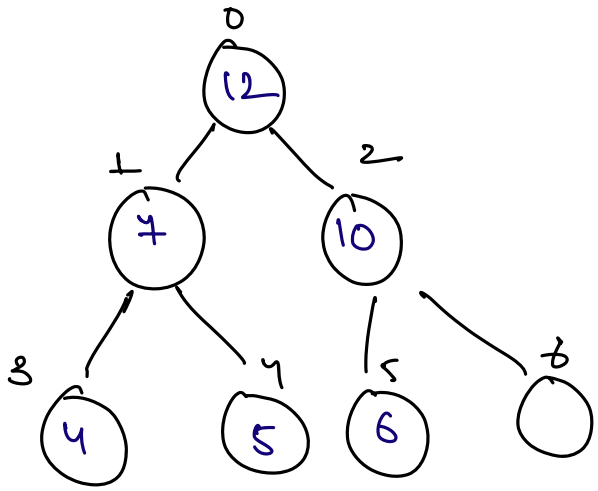
min

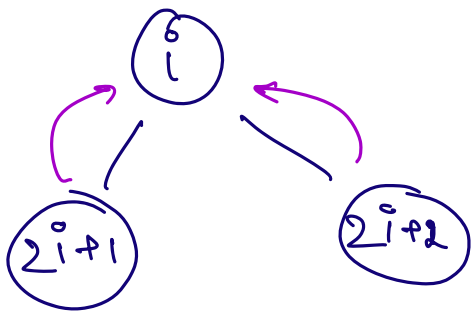












PriorityQueue < Min heap / Max heap.