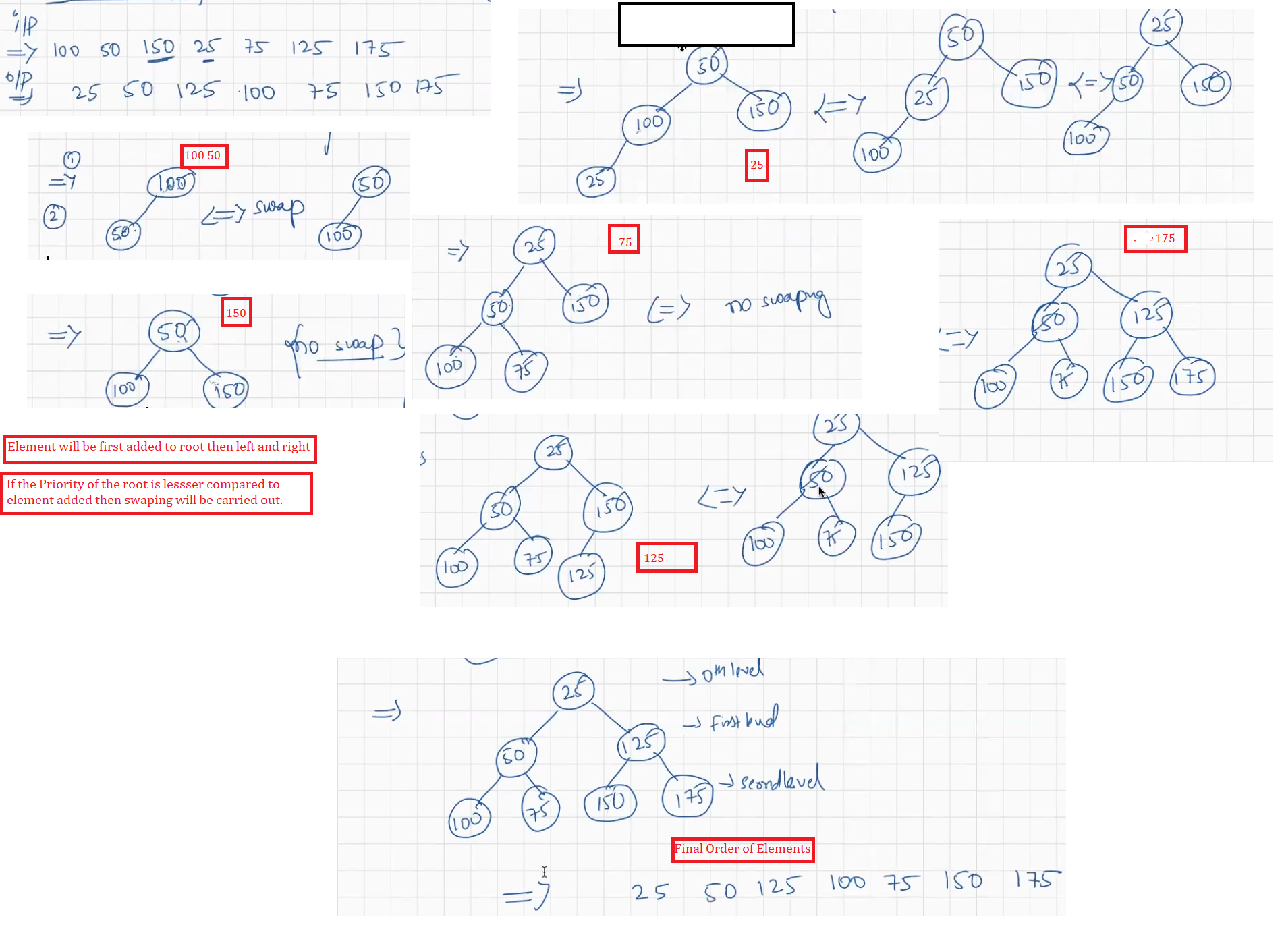
Priority Queue:

* It follows Min\_heap Datastructre.
* It implements Queue interface
* Index based accessing is not allowed.
* Priority Queue doesn’t permit null

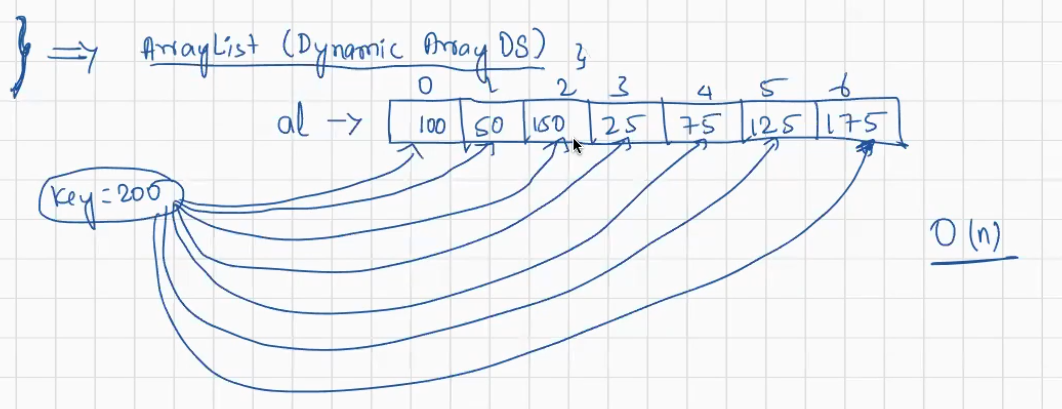
Highest Priority element will be at the start of the collection.



Highest Priority Element will be present at the root.

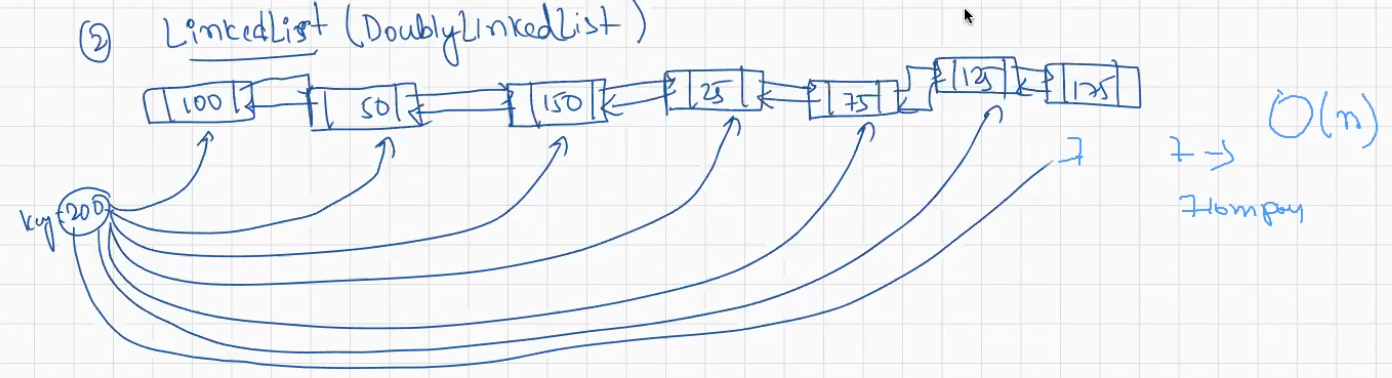
Search:

Searching in Array List might take O(n).

As we have 7 Objects here 7 Comparisions will happen.

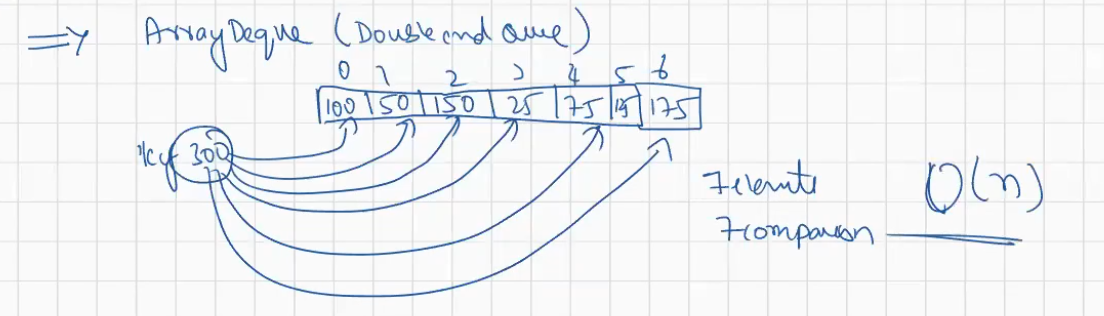
We need to go to Each of the index and search if the element is matching.

Linked List:

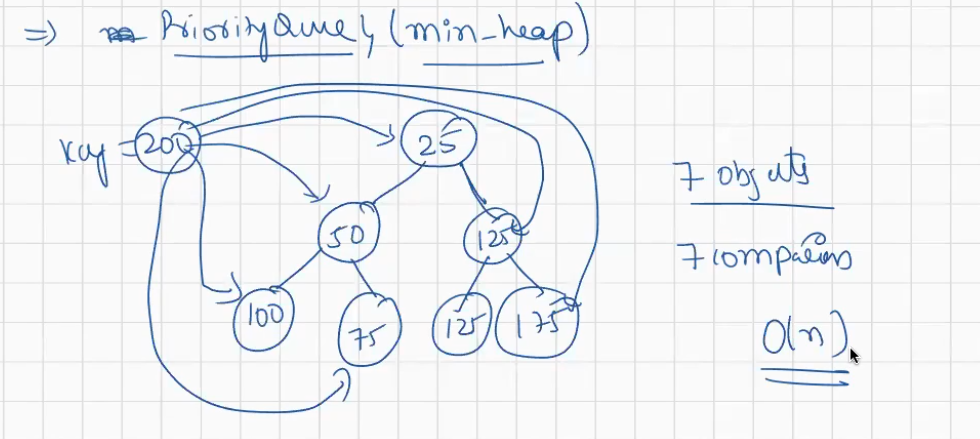
As we have 7 Objects here 7 Comparisions will happen.

We need to go to Each of the index and search if the element is matching.

Array Dequeue(Double Ended Queue):



Priority Queue:



All the above will take O(n) time for Searching

**TreeSet :**

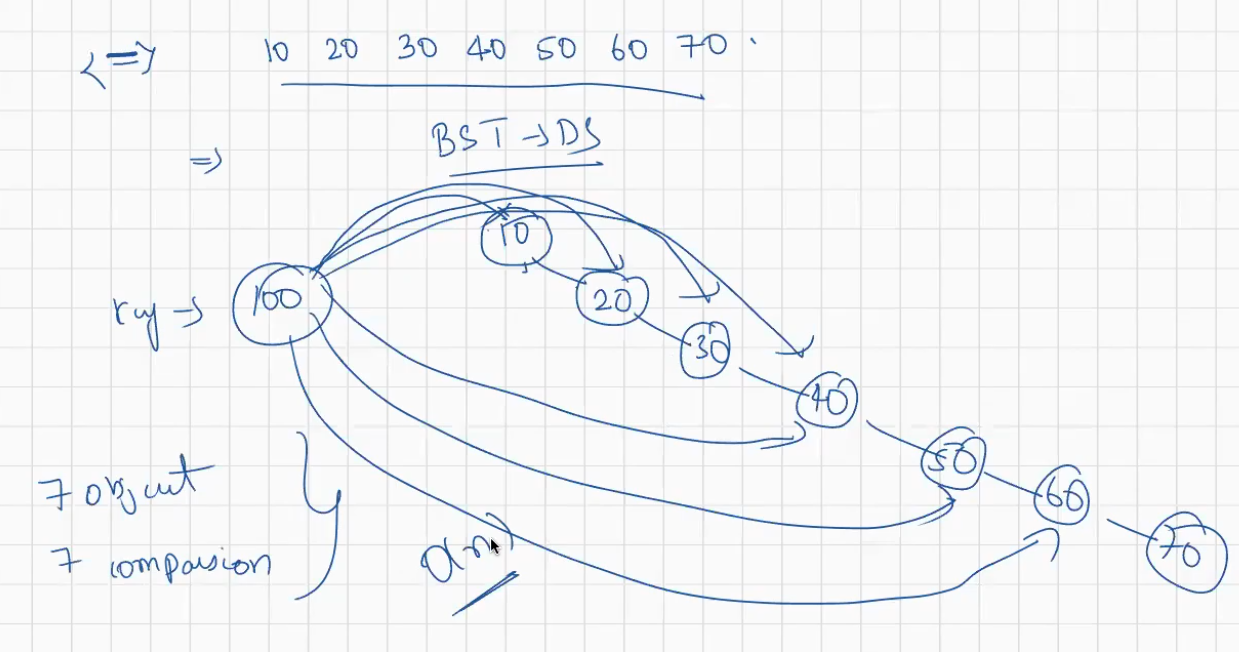
It follows Binary Search Tree Data Structure.

Does not allow duplicates

Elements will be displayed in the sorted order.

If the data stored is forming of the type of tree same as Binary Search Tree then searching will be better.

In case of Skewed Tree: Binary search tree is not good for searching as it takes O(n) time;



Above tree is Skewd Tree

If the data stored is forming a Balanced binary tree, then it will be best for searching.

Whereas if the data is stored is resulting to Skewed tree then it will take O(n) time for Searching.

TreeSetExample.java

Hash Set:

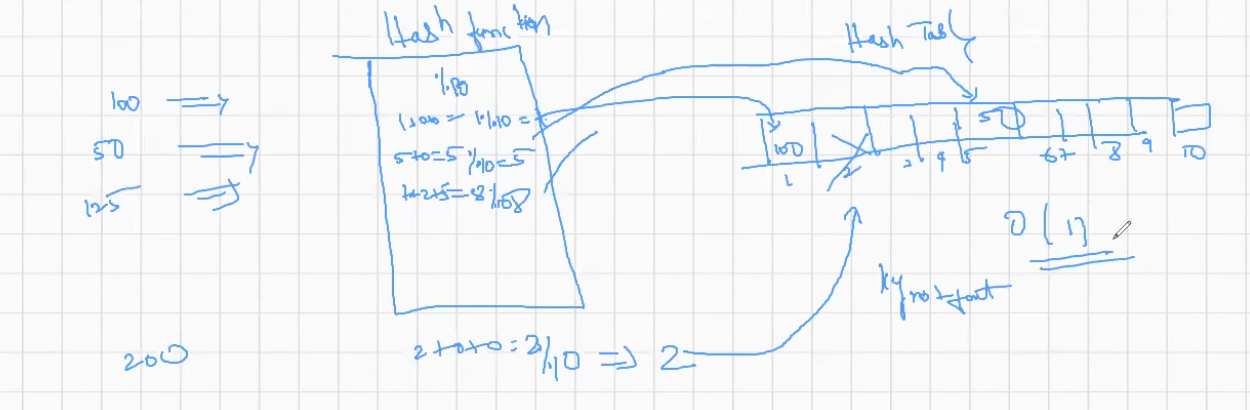
It follows Hashing Algorithm

Hash Function and Associated Hash table with Load factor of 75%.

Hash table contains buckets where the data will be stored.

HashSet does not store the data in the order of insertion. It stores the data based on the hash Function provided the input element will be stored in the specific bucket.

If the Hash Table is filled by 75% then it’s Size will be doubled.



Searching of the element takes constant time O(1). As the elements will be added to the hash table by applying hash function on that.

In LinkedHashSet elements will be stored in the insertion order.

import java.util.HashSet;

import java.util.LinkedHashSet;

public class HashSetandLinkedHashsetExample {

public static void main(String[] args) {

HashSet hs=new HashSet();

hs.add(100);

hs.add(50);

hs.add(150);

hs.add(25);

hs.add(125);

System.out.println("HashSet "+hs); //It does not preserve the Order of insertion //HashSet [50, 100, 150, 25, 125]

LinkedHashSet lhs=new LinkedHashSet();

lhs.add(100);

lhs.add(50);

lhs.add(150);

lhs.add(25);

lhs.add(125);

System.out.println("LinkedHashSet "+lhs); //It preserves the Order of Insertion //LinkedHashSet [100, 50, 150, 25, 125]

}

}

Even if we provide autoboxing will happen and convert primitive type data to Object.

Iterator itr=al1.iterator();

Itr.hasNext(); //It will check if there exists next Element

Itr.next(); // Goes the next element of the Collection and fetch that value.

To iterate over the elements we can use Iterator.

To iterate over the elements in the reverse order, we can use List Iterator.

ListIterator li=new ListIterator()

ListIterator is part of ArrayList and LinkedList.