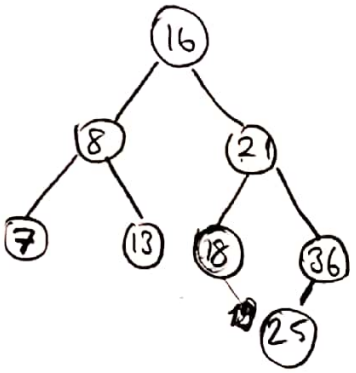
CSE 222 Midterm Examination

Muhammed
Bedir
ULUAY
1901042697

(1-) predecessor => largest element which is smaller than given value



16 21
8 13 36
18 7

we need to return root.left -> most right

Best	Aug	Worst
$\Theta(1)$	$\Theta(\log n)$	$\Theta(n)$

- Best and Worst are very rarely

```
public Node<E> findPred(E item){//Wrapper
    return findPred(root, item)
}

private Node<E> findPred(Node local, E item){
    if(local == null)
        return null;
    if(local.data.compareTo(item)==0){
        return precessor(root.left); //Sending left
    }
    else if(local.data.compareTo(item) >0){
        return findPred(local.right, item);
    }else{
        return findPred(local.left, item);
    }
}

private Node<E> precessor(Node local){ // We are return the most right node
    if(local.right != null)
        return precessor(local.right)
    return local;
}
```

5-)

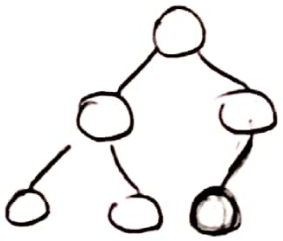
```
public void update(int index, E[] newArr){

    for(E e : newArr){
        put(index, e);
        index++;
    }

}
```

```
public boolean put(int index, E data){
    if(index > table.length)
        return false;
    table[index] = data;
    moveUpWard(index);
    moveDownWard(index);
}
```

Max Heap

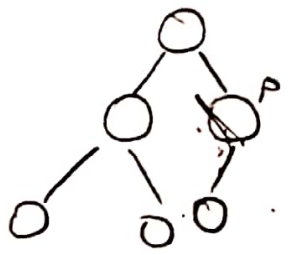


```
private void moveUpWard(int index){
    int child = index;
    int parent = (child-1)/2;
    while(parent >=0 && table[child].compareTo(table[parent] >0){
        swap(parent, child);
        child = parent;
        parent = (child-1)/2;
    }
}
```

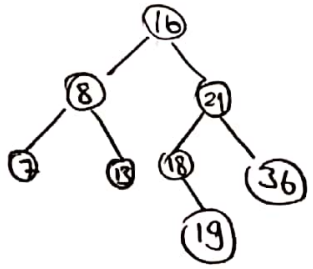
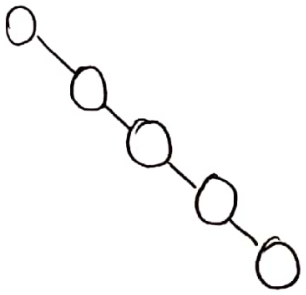
```
private void moveDownWard(int index) {
  int parent = index;
  while (true) {
    int leftchild = 2*parent + 1;
    int rightchild = leftchild + 1;
    if (leftchild > table.length)
      break;
    int maxChild = lef child;
    if(rightChild < table.length
      && table[rightchild].compareTo(table[leftchild] > 0){
      maxChild = rightchild;
    if (table[parent].compareTo(table.[maxChild) < 0) {
      swap(parent, maxChild);
    }
    else {
      break;
    }
  }
}

private void swap(int index1, int index2) {
    E tmp = table[index1];
    table[index1] = table[index2]
    table[index2] = tmp
}
```

max heap



4-) Time Complexity

Best	Avg	Worst
$\Theta(1)$	$\Theta(\log n)$	$\Theta(n)$
null		

19 = (21)

5-) Time Complexity

- We are using a lot of method in the update function but all insertion operation is happening in logaritmic time.

n = length of table

m = length of getting array as parameter

Best	Avg	Worst
$\Theta(1)$	$\Theta(m * \log n)$	$\Theta(m * \log n)$
$m=0$		

- Because we consider some rules to be complete binary tree so that this cannot be happening in linear time

(1, [13])