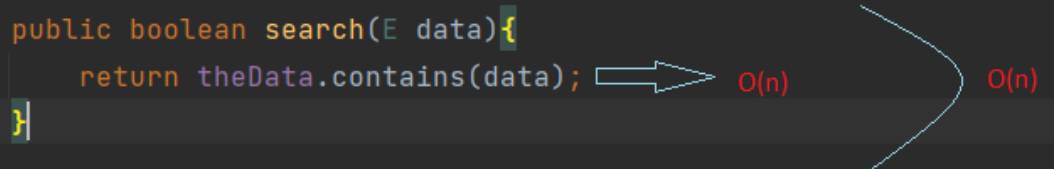
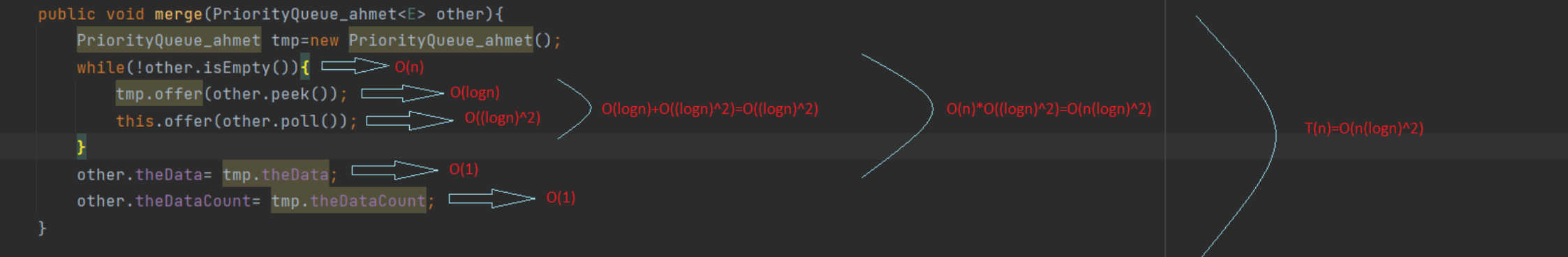
PART 1


public boolean search(E data){
return theData.contains(data);
}
O(n)
O(n)


public void merge(PriorityQueue_ahmet<E> other){
PriorityQueue_ahmet tmp=new PriorityQueue_ahmet();
while(!other.isEmpty()){
O(n)
tmp.offer(other.peek());
O(logn)
this.offer(other.poll());
O((logn)^2)
}
O(logn)+O((logn)^2)=O((logn)^2)
O(n)*O((logn)^2)=O(n(logn)^2)
T(n)=O(n(logn)^2)
other.theData= tmp.theData;
O(1)
other.theDataCount= tmp.theDataCount;
O(1)
}

```java
public boolean remove_largest_element(int a){
    if(a<1 || a>size()-1){
        System.out.println("yavaş");
    }
    ArrayList<E> tmp=new ArrayList<>();
    ArrayList<Integer> tmp2=new ArrayList<>();

    tmp.addAll(theData);
    tmp2.addAll(theDataCount);
    int index;
    E tmpe = null;
    for(int i=0;i<a;i++){
       tmpe= this.poll();
    }
    theData=tmp;
    theDataCount=tmp2;
    index=theData.indexOf(tmpe);

    if(theDataCount.get(index)>1){
        theDataCount.set(index,theDataCount.get(index)-1);
        return true;
    }

    swap(index, minChild: size()-1);
    theData.remove( index: size()-1);
    theDataCount.remove( index: size()-1);
    boolean flag=false;
    if(index>0){
        while(true){
            if(index>0 && theData.get(index).compareTo(theData.get((index-1)/2))>0){
                swap(index, minChild: (index-1)/2);
                index=(index-1)/2;
                flag=true;
            }
            else{
                if(flag==true){
                    return flag;
                }
                else
                    break;
            }
        }
    }
}
```

O(1)

O(n)

O(1)

O(logn)

O(nlogn)

O(1)

O(n)

O(1)

O(1)

O(1)

O(logn)

T(n)=O(nlogn)

```
    int leftchild,rightchild;
    while(true){
        if(index*2+1>=size()-1){
            break;
        }
        leftchild=index*2+1;
        rightchild=leftchild+1;
        if(theData.get(leftchild).compareTo(theData.get(rightchild))>0 && theData.get(index).compareTo(theData.get(leftchild))<0){
            swap(index,leftchild);
            index=leftchild;
        }
        else if(theData.get(rightchild).compareTo(theData.get(leftchild))>0 && theData.get(index).compareTo(theData.get(rightchild))<0){
            swap(index,rightchild);
            index=rightchild;
        }
        else
            break;
    }
    return true;
}
```

```
public E set(E item) {
    int index=theData.indexOf(lastReturned);
    int leftchild,rightchild;
    leftchild=index*2+1;
    rightchild=leftchild+1;
    theData.set(index,item);
    while(true){
        if(index*2+1>=size()){
            break;
        }
        leftchild=index*2+1;
        rightchild=leftchild+1;
        if(theData.get(leftchild).compareTo(theData.get(rightchild))>0 && theData.get(index).compareTo(theData.get(leftchild))<0){
            swap(index,leftchild);
            index=leftchild;
        }
        else if(theData.get(rightchild).compareTo(theData.get(leftchild))>0 && theData.get(index).compareTo(theData.get(rightchild))<0){
            swap(index,rightchild);
            index=rightchild;
        }
        else
            break;
    }
    while(true){
        if(theData.get(index).compareTo(theData.get((index-1)/2))>0){
            swap(index, minChild: (index-1)/2);
            index=(index-1)/2;
        }
        else
            break;
    }
    return null;

}
```

O(n)

O(1)

O(n)

O(1)

O(logn)

O(logn)

PART 2

```java
private Node<E> add(Node<E> localRoot, E item)
{
    if (localRoot == null) {
        addReturn = true;
        PriorityQueue_ahmet tmp=new PriorityQueue_ahmet();
        tmp.offer(item);
        return new Node<E>(tmp);
    }
    if (localRoot.data.size()<7 || localRoot.data.containQueue(item)){
        localRoot.data.offer(item);
        return localRoot;
    }
    else if (item.compareTo(localRoot.data.peek()) < 0) {
        localRoot.left = add(localRoot.left, item);
        return localRoot;
    }
    else {
        localRoot.right = add(localRoot.right, item);
        return localRoot;
    }
}
```

```
private Node<E> delete(Node<E> localRoot, E item) {
    if (localRoot == null) {
        deleteReturn=-1;
        return localRoot;
    }
    int compResult = item.compareTo(localRoot.data.peek());
    int index=localRoot.data.containQueueNumber(item);
    if(index>-1) {

        if(localRoot.data.size()>1 || localRoot.data.theDataCount.get(index)>1){
            deleteReturn=localRoot.data.theDataCount.get(index);
            deleteReturn--;
            localRoot.data.remove_element(item);
            return localRoot;
        }
        deleteReturn = 0;
        if (localRoot.left == null) {
            return localRoot.right;
        }
        else if (localRoot.right == null) {
            return localRoot.left;
        }
        else {
            if (localRoot.left.right == null) {

                localRoot.data = localRoot.left.data;
                localRoot.left = localRoot.left.left;
                return localRoot;
            }
            else{

                Iterator_ahmet iter=localRoot.data.iterator();
                iter.next();
                iter.set(findLargestChild(localRoot.left));
                return localRoot;
            }
        }

    }
    else if (compResult < 0) {
        localRoot.left = delete(localRoot.left, item);
        return localRoot;
    }
```

O(1)

O(1)

O(n)

O(n)

O(nlogn)

O(1)

O(1)

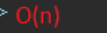
O(logn)

O(logn)

O(logn)

T(n)=O(n.(logn)^2)

```java
private int find(Node<E> localRoot, E target) {
    if (localRoot == null)
        return -1;
    int compResult = target.compareTo(localRoot.data.peek());
    int index=localRoot.data.containQueueNumber(target);
    if (index>-1)
        return localRoot.data.theDataCount.get(index);
    else if (compResult < 0)
        return find(localRoot.left, target);
    else
        return find(localRoot.right, target);
}
```



```java
private ArrayList<Integer> find_mode(Node<E> localRoot,ArrayList<Integer> mode){
    if(localRoot==null){
        return mode;
    }
    mode=find_mode(localRoot.right, mode);
    if(mode.get(0).compareTo(localRoot.data.modequeue().get(0))<0){
        mode=localRoot.data.modequeue();
    }
    mode=find_mode(localRoot.left,mode);

    return mode;
}
```