Algorthim countElementOfSeq(seq,D)

```
Iter:=seq.elements()
Cnt:=0
While(iter.hasNext()) then do
      E:=iter.nextObject()
      Cnt:=D.findValue€
 If(cnt===null) then
       D.insertItem(e,1)
 Else
    Cnt:=cnt+1;
     D.insertItem(e,cnt)
```

```
_countElementsOfSeq(seq, D) {
        // count the elements in seq and
store the count for each candidate in
Dictionary D
     let iter=seq.elements()
     let cnt=0;
     while(iter.hasNext()){
        let e=iter.nextObject()
        cnt=D.findValue(e)
        if(cnt===null){
            D.insertItem(e,1)
        }else{
            cnt++;
            D.insertItem(e,cnt)
        }
     }
    }
```

Algorthim countElementOfArray(arr,D)

For n of arr

Key:=D.findValue(n)

key:=D.findValue€

If(cnt===null) then

D.insertItem(n,1)

Else

key:=key+1;

D.insertItem(n,key)

```
_countElementsOfArray(arr, D) {
        // count the elements in seq and
store the count for each candidate in
Dictionary
        let key;
        for(let n of arr){
          key=D.findValue(n)//return the
values of the given id
          if(key===null)
             D.insertItem(n,1)
          else{
              key=key+1;
              D.insertItem(n,key)
          }
      }
    }
```

Algorthim findWinnesrFromDictionary(D)

```
  Iter:=D.items()
Max:=0;
While(iter.hasNext()) then do
    Item:=Iter.nextObject()
If(item.value()>max) then
     Winner=[]
     Winners.push(item)
     Max=item.value()
Else if item.value===max) the
  Winners.push(item)
Return winners;
```

```
  _findWinnersFromDictionary(D) {
        // The count for each candidate
should be in Dictionary D
        // Iterate through the Items (ID,
count) and find the winners and put in the
array
        let iterD = D.items();
        let winners = [];
```

```
        let max=0;
        while(iterD.hasNext()){
           let item= iterD.nextObject()
            if(item.value()>max){
                  winners=[]
                  winners.push(item)
                  max=item.value()
            }else if(item.value()===max){
                winners.push(item)
            }
        }
    return winners;
    }
```

Algorthim inserSeqIntoPQ(seq,PQ)

P:=seq.first()

PQ.insert(p.element(),p.element())

While(!seq.isLAst(p))

P:=seq.after(p)

PQ.insertItem(p.elelment(),p.element()

```
  _insertSeqIntoPQ(seq, PQ) {
        // insert the elements (candidate
ID's) from Sequence seq into the Priority
Queue PQ
        let p = seq.first();

PQ.insertItem(p.element(),p.element())
```

```
        while(!seq.isLast(p)){
            p=seq.after(p)

PQ.insertItem(p.element(),p.element())
        }
    }
```

Algorthim insertArtrayIntoPQ(arr,PQ)

For id of arr

PQ.insertItem(id,id)

```
_insertArrayIntoPQ(arr, PQ) {
        // insert the elements (candidate
ID's) from Sequence seq into the Priority
Queue PQ
    for(let id of arr){
       PQ.insertItem(id,id)
     }
    }
```

Algorthim findWinnerPriortyQuea(PQ)

```
      Max:=0;
   Current:=PQ.removeMin()
  Cnt:=1
while !PQ.isEmpty() do
  next:=PQ.removeMin()
if(current===next)
     cnt:=cnt+1;
else  if cnt>max
        winner.push((curr,cnt))
      max:=cnt;
  if cnt===max
       winner.push((curr,cnt))
cnt:=1;
current:=next;

return winner;
```

```
    _findWinnersFromPQ(PQ) {
          // Traverse the Priority Queue and
determine the winners
          let winners = [];
          let max=0;
```

```
        let cnt=0;
        let cur=PQ.removeMin()
        while(!PQ.isEmpty()){
         let next=PQ.removeMin()
         if(cur===next)
            cnt++;
        else{
            if(cnt>max){
                max=cnt;
                winners=[]
                winners.push(new
Pair.Item(cur,cnt))
            }else if(cnt===max){
                winners.push(new
Pair.Item(cur,cnt))
            }
        cnt=1;
        cur=next
        }
        }
        if(cnt>max){
            max=cnt;
            winners=[]
            winners.push(new
Pair.Item(cur,cnt))
        }else if(cnt===max){
            winners.push(new
Pair.Item(cur,cnt))
        }
        return winners;
```

```
}
```