Milestone 4

- https://leetcode.com/problems/majority-element

```
class Solution {
    public int majorityElement(int[] nums) {
        int majority=nums[0];
        int c=1;
        for(int i=1;i<nums.length;i++)
        {
            if(nums[i]==majority)
                c++;
            else
                c--;
            if(c==0)
            {
                majority=nums[i];
                c=1;
            }
        }
        int count=0;
        for(int i=0;i<nums.length;i++)
        {
            if(nums[i]==majority)
                count++;
        }
        if(count>(nums.length/2))
            return majority;
        else
            return 0;
    }
}
```

- https://leetcode.com/problems/non-overlapping-intervals

```
class Solution {
    public int eraseOverlapIntervals(int[][] intervals) {
        int l = intervals.length;
        if(l == 0) return 0;
        Arrays.sort(intervals, (a, b)->{
            return a[1] - b[1];
        });
        int end = intervals[0][1];
        int c = 1;
        for(int i = 1; i < l; i++){
            if(intervals[i][0] >= end){
                end = intervals[i][1];
                c++;
            }
        }
        return l - c;
    }
}
```

- https://leetcode.com/problems/kth-largest-element-in-an-array

```
class Solution {
        int[] N;
    public int findKthLargest(int[] nums, int k) {
        if(nums.length == 1)
            return nums[0];
        N = nums;
        return quickSelect(0, nums.length - 1, k - 1);
    }
    public int quickSelect(int l, int r, int k) {
        while(l <= r)
        {
```

```
        int p = partition(l, r);
        if(p == k)
            return N[k];
        else if(p > k)
            r = p - 1;
        else
            l = p + 1;
    }

    return N[k];
}
public int partition(int left, int right) {
    int R= (left + right) / 2;
    int p = right;

    int tmp = N[R];
    N[R] = N[p];
    N[p] = tmp;

    int P = left;
    for(int i=left ; i<right ; i++)
    {
        if(N[i] >= N[p])
        {
            tmp = N[i];
            N[i] = N[P];
            N[P] = tmp;
            P++;
        }
    }

    tmp = N[P];
    N[P] = N[p];
    N[p] = tmp;
```

```
        return P;
    }
}
```

- https://leetcode.com/problems/sort-an-array

```
class Solution {
   public int[] sortArray(int[] nums)
    {
        mergeSort(nums, 0, nums.length - 1);
        return nums;
    }
    void merge(int a[], int beg, int mid, int end)
    {
        int i, j, k;
        int n1 = mid - beg + 1;
        int n2 = end - mid;
        int[] LeftArray= new int[n1];
        int[] RightArray= new int[n2];
        for (i = 0; i < n1; i++)
        LeftArray[i] = a[beg + i];
        for (j = 0; j < n2; j++)
        RightArray[j] = a[mid + 1 + j];

        i = 0;
        j = 0;
        k = beg;
        while (i < n1 && j < n2)
        {
            if(LeftArray[i] <= RightArray[j])
            {
                a[k] = LeftArray[i];
                i++;
            }
            else
            {
                a[k] = RightArray[j];
                j++;
            }
            k++;
        }
        while (i<n1)
        {
            a[k] = LeftArray[i];
            i++;
            k++;
        }

        while (j<n2)
        {
            a[k] = RightArray[j];
            j++;
            k++;
        }
    }
    void mergeSort(int nums[], int b, int e)
    {
        if (b < e)
        {
            int m = (b + e) / 2;
            mergeSort(nums, b, m);
            mergeSort(nums, m + 1, e);
            merge(nums, b, m, e);
        }
    }
}
```

- https://leetcode.com/problems/hand-of-straights

```
class Solution {
    public boolean isNStraightHand(int[] hand, int groupSize) {
        if(hand.length%groupSize!=0)
            return false;
        TreeMap<Integer,Integer> card_count=new TreeMap();
        for(int card:hand)
        {
            if(!card_count.containsKey(card))
            {
                card_count.put(card,1);
            }
            else
            {
                card_count.replace(card,card_count.get(card)+1);
            }
        }

        while(card_count.size()>0)
        {
            int min=card_count.firstKey();
            for(int card=min;card<min+groupSize;card++)
            {
                if(!card_count.containsKey(card))
                    return false;
                int count=card_count.get(card);
                if(count==1)
                    card_count.remove(card);
                else
                    card_count.replace(card,count-1);
             }
        }
     return true;
    }
}
```

- https://leetcode.com/problems/string-without-aaa-or-bbb

```
class Solution {
    public String strWithout3a3b(int a, int b) {
        StringBuffer sb=new StringBuffer();

        while(a>0 && b>0)
        {
            if(a>b)
            {
                sb.append("aa").append("b");
                a-=2;
                b-=1;
            }
            else
            {
                if(a==b && a==1)
                {
                    sb.append("a").append("b");
                    a-=1;
                    b-=1;
                    continue;
                }
                sb.append("a").append("bb");
                a-=1;
                b-=2;
            }
        }
        while(a>0)
        {
            sb.append("a");
            a--;
        }
        while(b>0)
        {
            sb.insert(0,"b");
            b--;
        }
        return sb.toString();
    }
}
```