Divide and Conquer

BINARY SEARCH -

```
def binarySearch(n, a, low, high):
    if low > high:
        return -1
    else:
        mid = int(low+high)/2
        if A[mid] == n:
            return mid+1
        elif A[mid] < n:
            low = mid+1
        else:
            high = mid-1
    return binarySearch(n, a, low, high)
```

Multiplying Polynomials (NORMAL APPROACH):

```
def multPoly(a, b):
    p = []
    for i in range(0, len(a)):
        p.append(0)
    for i in range(0, len(a)):
        for j in range(0, len(b)):
            p[i+j] += a[i] + b[j]

    return p
```

$A = 3x^2 + 2x + 5$

$B = 5x^2 + x + 2$

$AB = 3x^2(5x^2+x+2) + 2x(5x^2+x+2) + 5(5x^2+x+2)$

$= [15x^4 + 3x^3 + 6x^2] + [10x^3 + 2x^2 + 4] + [25x^2 + 5x + 10]$

$\Rightarrow 15x^4 + 3x^3 + 6x^2$

$10x^3 + 2x^2 + 4$

$25x^2 + 5x + 10$

$15x^4 + 13x^3 + 33x^2 + 9x + 10$

$A = [3,2,5]$; $B = [5,1,2]$

$i = 0$:

$j = 0 \rightarrow P[0+0] = P[0] \Rightarrow 15$

$j = 1 \rightarrow P[0+1] = P[1] \Rightarrow 3$

$j = 2 \rightarrow P[0+2] = P[2] \Rightarrow 6$

$i = 1$:

$j = 0 \rightarrow P[1+0] = P[1] \Rightarrow 3 + 10 = 13$

$j = 1 \rightarrow P[1+1] = P[2] \Rightarrow 6 + 2 = 8$

$j = 2 \rightarrow P[1+2] = P[3] \Rightarrow 4$

$i = 2$:

$j = 0 \rightarrow P[2+0] = P[2] \Rightarrow 8 + 25 = 33$

$j = 1 \rightarrow P[2+1] = P[3] \Rightarrow 4 + 5 = 9$

$j = 2 \rightarrow P[2+2] = P[4] \Rightarrow 10$

$\therefore P = [15, 13, 33, 9, 10]$

Multiplying Polynomials (KARATSUBA APPROACH):

(MULTIPLYING LARGE NUMBERS)

$$\boxed{\begin{array}{l} X = 146 \,\vdots\, 123 \\ Y = 581 \,\vdots\, 621 \end{array}}$$

```
def karatsuba(x, y):
    if x<10 and y<10:
        return x*y
    else:
        n = max(len(str(x)), len(str(y)))
        half = n //2
        a = x // (10**half)
        b = x % (10**half)
        c = y // (10**half)
        d = y % (10**half)
        ac = karatsuba(a, c)
        bd = karatsuba(b, d)
        ad_plus_bc = karatsuba(a+b, c+d) - ac - bd
        return ac*(10**(2*half)) + ad_plus_bc*(10**half) + bd
```

$a = x \,//\, (10^{**}half) \rightarrow 146123 \,//\, 10^3 \Rightarrow 146.123$ (but cuz $//$) $= 146$

$b = x \,\%\, (10^{**}half) \rightarrow 146123 \,\%\, 10^3 \Rightarrow 146.123$ (but cuz $\%$) $= 123$

$X = 146 \times 10^3 + 123$ & $Y = 581 \times 10^3 + 621$

$\Rightarrow X = a \times 10^3 + b$ & $Y = c \times 10^3 + d$

$\therefore X \times Y = (a * 10^3 + b) \times (c \times 10^3 + d)$

$= (ac)10^6 + \underbrace{(ad + bc)}_{\substack{\text{written as} \\ (a+b)*(c+d) - ac - bd}} 10^3 + bd$

$$\boxed{\therefore X \times Y = (ac)10^6 + [(a+b)*(c+d) - ac - bd]10^3 + bd}$$

Merge Sort-

```
def merge(a):
    if len(a)>1:
        left_arr = a[0:len(a)//2]
        right_arr = a[len(a)//2 : len(a)]

        merge(left_arr)
        merge(right_arr)

        i=0
        j=0
        k=0
        while (i<len(left_arr) and j<len(right_arr)):
            if left_arr[i] < right_arr[j]:
                a[k] = left_arr[i]
                i+=1
            else:
                a[k]= right_arr[j]
                j+=1
            k+=1
        while i<len(left_arr):
            a[k]= left_arr[i]
            i+=1
            k+=1
        while j<len(right_arr):
            a[k]= right_arr[j]
            j+=1
            k+=1
    return a
```

Quick Sort-

```
def quickSort(a, start, end):
    if start>=end:
        return
    else:
        p= partition(a, start, end)
        quicksort(a, start, p-1)
        quicksort(a, p+1, end)
    return a

def partition(a, start, end):
    pivot= a[start]
    left= start+1
    right = end
    while True:
        while left<=right and a[left]<pivot:
            left+=1
        while left<=right and a[right]>pivot:
            right-=1
        if left<=right:
            a[left], a[right] = a[right], a[left]
        else:
            break
    a[start], a[right] = a[right], a[start]
    return right
```

Moore's Voting Algorithm:

(nothing to do with Divide & Conquer though)

```
def checkMajority(a):
    candidate = none
    num = 0
    count = 0

    for num in a:
        if count == 0:
            candidate = num
        if candidate == num:
            count += 1
        else:
            count -= 1

    count = 0
    for num in a:
        if num == candidate:
            count += 1
    if count > len(a)//2:
        return 1
    else:
        return 0
```

NOTES:

1. To check if any element in array has frequency greater than half length of array.
2. ONLY 1 such element can exist, if at all.
3. First for loop keeps track of a counter which will always have a value greater than 0 if a legit 'candidate' exists.
4. First for loop picks 1 candidate that MIGHT have a majority.
5. Second for loop confirms if the 'candidate' is a majority or not.

I) First For loop -

ⓐ Check if count is 0,
if yes, then start fresh & set 'candidate' as current no.

ⓑ Check if current no. is equal to the 'candidate' &
if yes, increment counter
else, decrement counter