Maximum Product Subarray (CONT)

```
result = nums[0]
for num in nums[1:]:
    ~~temp_max~~
    cur_max = max(num, cur_max
        * num, cur_min * num)
    cur_min = min(num, cur_max *
        num, cur_min * num)
    result = max(result, cur_max)
return result
```

Minimum in Rotated Sorted Array

```
def findMin(nums):
  left, right = 0, len(nums) -1
  mid = 0
  while left < right:
     mid = int((left + right)/2)
     if nums[mid] > nums[right]:
        left = mid +1
     else:
        right = mid
  return nums[left]
```

Search in Rotated Sorted Array

```
def search(nums, target):
  left = 0
  right = len(nums) -1
  while left <= right:
    mid = (left + right)//2
    if nums[mid] == target:
       return mid
    if nums[left] <= nums[mid]:
    # left half is sorted
       if nums[left] <= target <= nums[mid]:
          right = mid -1
       else:
          left = mid +1
    else:
       if nums[mid] < target <= nums
                                [right]:
          left = mid +1
       else:
          right = mid -1
  return -1
```

Three Sum

```
def two_sum(arr, target):
  left, right = 0, len(arr) -1
  cur_sum = float('-inf')
  while left < right & cur_sum != target:
    cur_sum = arr[left] + arr[right]
    if cur_sum < target:
       left += 1
    elif cur_sum > target:
       right -= 1
  if cur_sum == target & left != right:
       return (left, right)
  return (-1, -1)
def three_sum(arr, target):
  arr.sort()
  set_indices = set()
  cnt = len(arr)
  for i in range(cnt):
     rem = target - arr[i]
     left, right = two_sum(arr, rem)
     if left != -1:
        set_indices.add((arr[
          left], arr[right],
              arr[i]))
  return list(set_indices)
```

Container w/ Most Water

```
def maxArea(heights):
  left = 0
  right = len(height) -1
  max_area = 0
  while left < right:
     width = right - left
     area = min(height[left], heigh-
              t[right]) * width
     max_area = max(max_area, area)
     if height[left] < height[right]:
        left += 1
     else:
        right -= 1
  return max_area
```

Sum of Two Integers (BIT)

```
def getSum(a, b):
   MASK = 0xFFFF FFFF
   INT_MASK = 0x7FFF FFFF
   while b != 0:
      sum_without_carry = (a^b) & MASK
      carry = ((a&b) << 1) & MASK
      a, b = sum_without_carry, carry
   # if a is negative, apply 1's complement
   # followed by not
   if a > INT_MAX:
      return ~(a ^ MASK)
   return a
```