# Maximum Subarray Sum

- Given an integer array, find the maximum sum among all subarrays possible.
- To find maximum subarray sum (contiguous) of given 1-D array.
- Array may contain negative and positive numbers which makes this a difficult problem.
- Brute force will take O(N^2) time while a divide and conquer approach will take O(N log N) time.
- If all the array entries were positive, then the maximum-subarray problem would present no challenge, since the entire array would give the greatest sum.
- Example: input array = [-3, 2, 3, 4, -4, -1]. Here maximum subarray is [2,3,4]. Hence maximum subarray sum is 9.

**Brute force approach:**

```
int max_Subarray_Sum ( int A[] , int n)
{
   int max_sum = 0
   for ( i = 0 to n-1)
   {
      sum=0
      for( j = i to n-1)
      {
        sum = sum +  A[j]
        if (sum > max_sum)
            max_sum = sum
      }
   }
```

```
   retun max_sum
}
```

Time Complexity: O(n^2)

**Divide and Conquer Approach:**

1. Divide an array in two halves.
2. Find maximum subarray sum in left half.
3. Find maximum subarray sum in right half.
4. Find maximum subarray sum which crosses the midpoint.
5. Maximum of step 2,3 and 4 is our answer.

**Pseudocode:**

```
Left_Right_MaximumSubarray(A,l ow,high)
      if (high == low)       // base case
          return A[low]
      else
          mid = (low + high) / 2
          leftsum = Left_Right_MaximumSubarray (A, low, mid)
          rigthsum = Left_Right_MaximumSubarray (A, mid+1, high)
          midcrosssum = Midpoint_Crosssum (A, low, mid, high)
          return maximum (leftsum, rightsum, midcrosssum)

Midpoint_Crosssum(A, low, mid, high)
      sum = 0
      leftsum = INT_MIN;        // INT_MIN – defines minimum value for an
      integer variable. Value is -2147483648
      for i = mid down to low
         sum = sum + A[i]
```

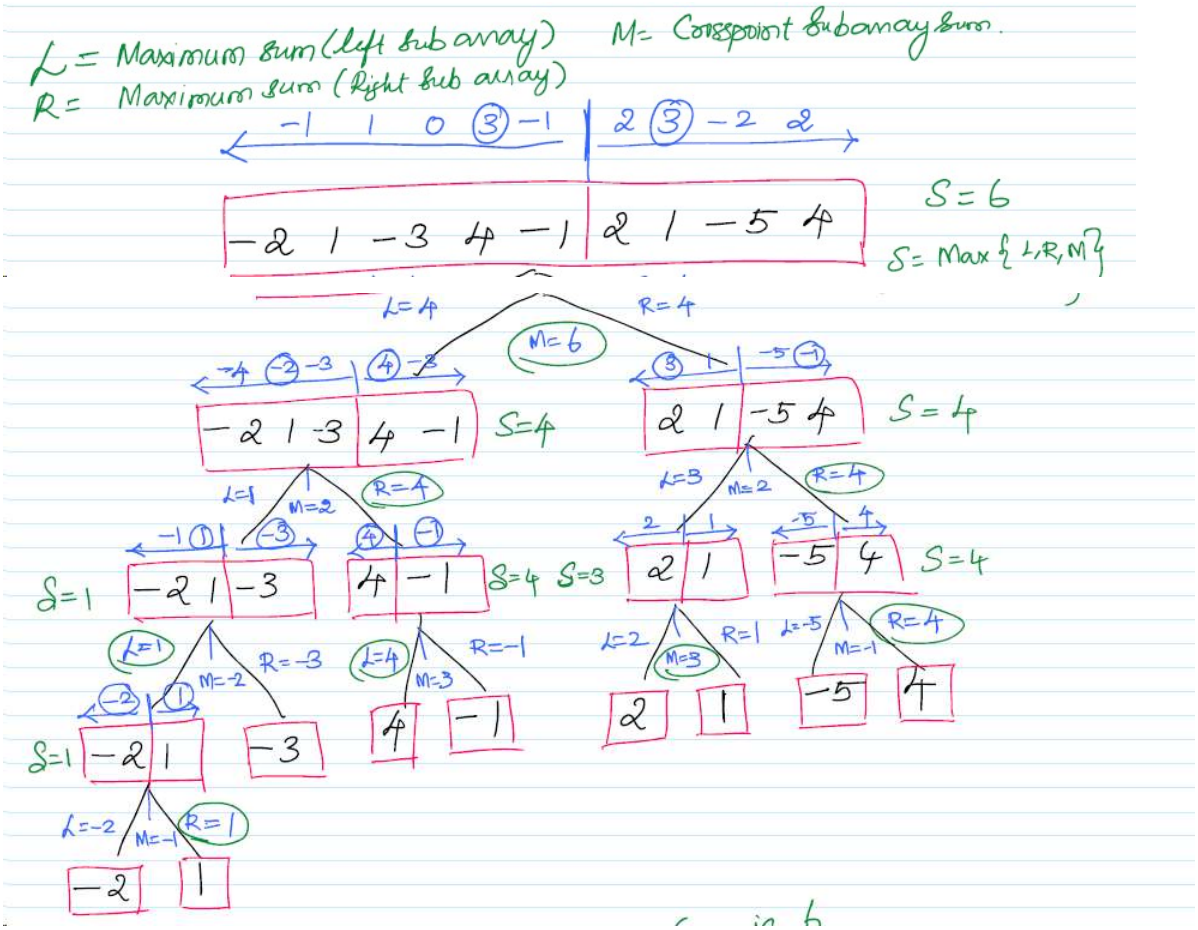
```
    if sum > leftsum
        leftsum = sum
sum = 0
rightsum = INT_MIN;
for i = mid+1 to high
    sum = sum + A[i]
    if sum > rightsum
        rightsum = sum
return leftsum + rightsum
```

**Time Complexity:**

T(n) = 2T(n/2) + n

T(n) = O(n.log(n))  //you may apply master method to derive this TC

## Example: 1

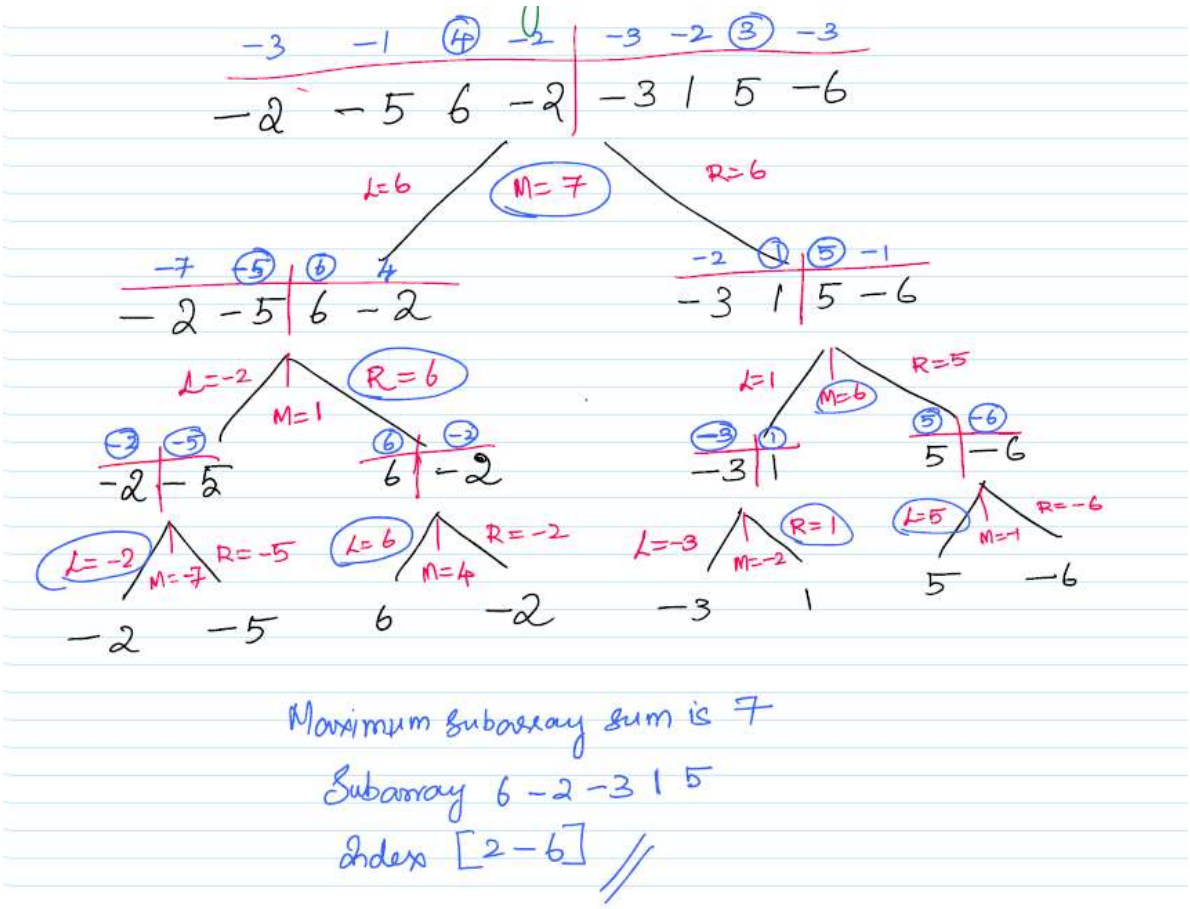



Maximum subarray Sum is 6

Array index: [3-6]

Array elements [4, -1, 2, 1]

## Example: 2




Maximum subarray sum is 7

Subarray 6 -2 -3 1 5

Index [2-6] //