Reg.No : 24BCE0554
Name : Partha Pratim Gogoi
Topic : Maximum Sub-array Sum

Algorithm

```
1  getMaxCrossSum(arr, low, mid, high)                          [T.C: O(n)]
   1.1  leftSide = INT_MIN
   1.2  sum = 0
   1.3  for i mid→low                                           [T.C: O(n)]
      1.3.1   sum+=arr[i]
      1.3.2   leftSide = max(leftSide, sum)
   1.4  rightSide = INT_MIN
   1.5  sum = 0
   1.6  for i low→mid                                           [T.C: O(n)]
      1.6.1   sum+=arr[i]
      1.6.2   rightSide = max(rightSide, sum)
   1.7  return leftSide + rightSide
2  getMaxSubarraySum(arr, low, high)                          [T.C: O(nlogn)]
   2.1  if low==high then return arr[low]
   2.2  mid = low + (high-low)/2
   2.3  return max(getMaxSubarraySum(arr,low, mid),     [T(n)=T(n/2)]
                   getMaxSubarraySum(arr,mid+1,high),   [T(n)=T(n/2)]
                   getMaxCrossSum(arr,low,mid,high))      [T(n)=O(n)]
3  getMaxSubarray()
```

Time Complexity

getMaxSubarray()

T.C. = T(n/2) + T(n/2) + O(n)
= 2T(n/2) + O(n)

By Master's Theorem

T.C. = O(nlogn)

Total Time Complexity = O(nlogn)

Source Code

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <climits>
using namespace std;

//////////////////////
/// Cross sum logic ///
//////////////////////
long long getMaxCrossSum(const vector<int>& arr, int low, int mid, int high) {
      // --- Initialize sum --------- //
      long long sum = 0;

      // --- Left side -------------- //
      long long leftSide = INT_MIN;
      sum = 0;
      for (int i=mid; i>=low; i--) {
            sum += arr[i];
            leftSide = max(leftSide, sum);
      }
      // --- Right side ------------- //
      long long rightSide = INT_MIN;
      sum = 0;
      for (int i=mid+1; i<=high; i++) {
            sum += arr[i];
            rightSide = max(rightSide,sum);
      }
      // --- Debugging --- //
      // cout << leftSide << endl;
      // cout << rightSide << endl;
      // cout << leftSide+rightSide << endl << endl;

      return leftSide + rightSide;
}
//////////////////
/// Propagation ///
//////////////////
int getMaxSubarraySum(const vector<int>& arr, int low, int high) {
      if (low == high) return arr[low];

      // --- Initialize mid -------- //
      int mid = low + (high-low)/2;

      // --- Get the maximum possible subarray ------- //
      // --- CASE 1: Left subarray sum is largest ---- //
      // --- CASE 2: Right subarray sum is largest --- //
      // --- CASE 3: Cross subarray sum is largest --- //
      int leftSum = getMaxSubarraySum(arr, low, mid);
      int rightSum = getMaxSubarraySum(arr, mid+1, high);
      int crossSum = (int) getMaxCrossSum(arr, low, mid, high);

      // --- Find Max sub-array sum value --- //
      int maxSum = max(leftSum, rightSum);
      maxSum = max(maxSum, crossSum);

      return maxSum;
}
```

```
/////////////////
/// Intiation ///
/////////////////
int maxSubarraySum(vector<int>& arr) {
      return getMaxSubarraySum(arr, 0, arr.size()-1);
}

///////////////////
/// Driver Code ///
///////////////////
int main() {
      vector<int> arr = {-2, 1, -3, 4, -1, 2, 1, -5, 4};
      int maxSum = maxSubarraySum(arr);

      cout << "Maximum sub-array sum possible in given array is: ";
      cout << maxSum << endl;

      return 0;
}
```

Sample Output

```
24BCE0554

◄ 0s ◎ ../'[53] - Maximum Subarray Sum'/a.out
Maximum sub-array sum possible in given array is: 6
```