**PRACTICAL NO. 4**

Roll no.: 14

Batch: B1

Name: Abhishek Angadi

Aim: Implement maximum sum of subarray for the given scenario of resource allocation using the divide and conquer approach.

Code:

public class pract4{

    static class Result{

        int low;

        int sum;

        int high;

        public Result(int low, int high, int sum) {

            this.low=low;

            this.high=high;

            this.sum=sum;

        }

    }

    public static Result find\_maximum\_crossing\_subarray(int arr[], int low,

int mid, int high){

        int left\_sum = Integer.MIN\_VALUE;

        int sum=0;

        int max\_left=0, max\_right=0;

        for(int i=mid;i>=low;i--){

            sum+=arr[i];

            if(sum>left\_sum){

                left\_sum = sum;

                max\_left=i;

            }

        }

          int right\_sum = Integer.MIN\_VALUE;

          sum=0;

          for(int j=mid+1;j<=high;j++){

            sum+=arr[j];

            if(sum>right\_sum){

                right\_sum=sum;

                max\_right=j;

            }

          }

        return new Result(max\_left, max\_right, left\_sum+right\_sum);

    }

    public static Result find\_maximum\_subarray(int arr[], int low, int high){

        int mid=0;

        if(low==high){

            return new Result(low, high, arr[low]);

        }

        else{

            mid=(low+high)/2;

            //Left max subarray

           Result left = find\_maximum\_subarray(arr, low, mid);

           Result right = find\_maximum\_subarray(arr, mid+1, high);

           Result cross = find\_maximum\_crossing\_subarray(arr, low, mid, high);

           if((left.sum>=right.sum)&&(left.sum>=cross.sum)){

            return new Result(left.low, left.high, left.sum);

           }

           else if((right.sum>=left.sum)&&(right.sum>=cross.sum)){

            return new Result(right.low, right.high, right.sum);

           }

           else{

            return new Result(cross.low, cross.high, cross.sum);

           }

        }

    }

    public static void main(String[] args) {

       int arr[] = {-2, 1, -3, 4, -1, 2, 1, -5, 4};

        Result result = find\_maximum\_subarray(arr, 0, arr.length - 1);

        System.out.println("Low Index: " + result.low);

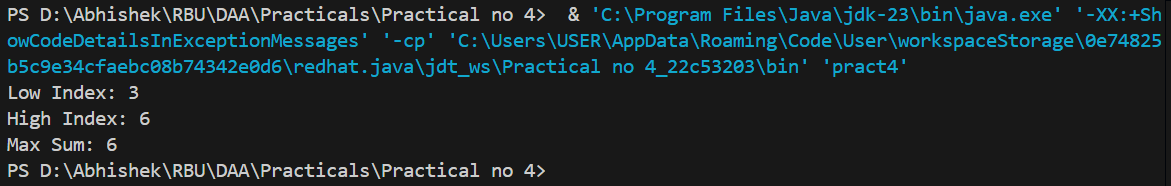
        System.out.println("High Index: " + result.high);

        System.out.println("Max Sum: " + result.sum);

    }

}

Output:



Conclusion:

Hence, we successfully implemented algorithm for maximum subarray using Divide and Conquer approach.

Github: https://github.com/Shadow3456rh/