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:

```
PS D:\Abhishek\RBU\DAA\Practicals\Practical no 4>  & 'C:\Program Files\Java\jdk-23\bin\java.exe' '-XX:+Sh
owCodeDetailsInExceptionMessages' '-cp' 'C:\Users\USER\AppData\Roaming\Code\User\workspaceStorage\0e74825
b5c9e34cfaebc08b74342e0d6\redhat.java\jdt_ws\Practical no 4_22c53203\bin' 'pract4'
Low Index: 3
High Index: 6
Max Sum: 6
PS D:\Abhishek\RBU\DAA\Practicals\Practical no 4>
```

Conclusion:

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

Github: https://github.com/Shadow3456rh/DAA-RbuPracticals/tree/main/Practical%20no%204