```
1     /**
2      * Recursive maximum contiguous subsequence sum algorithm.
3      * Finds maximum sum in subarray spanning a[left..right].
4      * Does not attempt to maintain actual best sequence.
5      */
6     private static int maxSumRec( int [ ] a, int left,
7                                             int right )
8     {
9         int maxLeftBorderSum = 0, maxRightBorderSum = 0;
10        int leftBorderSum = 0, rightBorderSum = 0;
11        int center = ( left + right ) / 2;
12
13        if( left == right )  // Base case
14            return a[ left ] > 0 ? a[ left ] : 0;
15
16        int maxLeftSum  = maxSumRec( a, left, center );
17        int maxRightSum = maxSumRec( a, center + 1, right );
18
19        for( int i = center; i >= left; i-- )
20        {
21            leftBorderSum += a[ i ];
22            if( leftBorderSum > maxLeftBorderSum )
23                maxLeftBorderSum = leftBorderSum;
24        }
25        for( int j = center + 1; j <= right; j++ )
26        {
27            rightBorderSum += a[ j ];
28            if( rightBorderSum > maxRightBorderSum )
29                maxRightBorderSum = rightBorderSum;
30        }
31
32        return max3( maxLeftSum, maxRightSum,
33                     maxLeftBorderSum + maxRightBorderSum );
34    }
35
36    // Publicly visible routine
37    public static int maxSubSum4( int [ ] a )
38    {
39        return maxSumRec( a, 0, a.length - 1 );
40    }
```

Figure 7.18 Divide-and-conquer algorithm for maximum contiguous subsequence sum problem

Figure 7.19 graphically illustrates how the algorithm works for $N = 8$ elements. Each rectangle represents a call to `maxSumRec`, and the length of the rectangle is proportional to the size of the subarray (and hence the cost of the scanning of the subarray) being operated on by the invocation. The initial call is shown on the first line. Notice that the size of the subarray is N. This represents the cost of the scanning for the third case. The initial call then makes two recursive calls, yielding two subarrays of size $N/2$. The cost of each scan in case 3 is