素 A_1 到 A_7)。

我们看到，在形成本例中的最大和子序列的三种方式中，最好的方式是包含两部分的元素。于是，答案为 11。图 2-7 提出了这种策略的一种实现手段。

```
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, int right )
7       {
8           if( left == right )  // Base case
9               if( a[ left ] > 0 )
10                  return a[ left ];
11              else
12                  return 0;
13
14          int center = ( left + right ) / 2;
15          int maxLeftSum  = maxSumRec( a, left, center );
16          int maxRightSum = maxSumRec( a, center + 1, right );
17
18          int maxLeftBorderSum = 0, leftBorderSum = 0;
19          for( int i = center; i >= left; i-- )
20          {
21              leftBorderSum += a[ i ];
22              if( leftBorderSum > maxLeftBorderSum )
23                  maxLeftBorderSum = leftBorderSum;
24          }
25
26          int maxRightBorderSum = 0, rightBorderSum = 0;
27          for( int i = center + 1; i <= right; i++ )
28          {
29              rightBorderSum += a[ i ];
30              if( rightBorderSum > maxRightBorderSum )
31                  maxRightBorderSum = rightBorderSum;
32          }
33
34          return max3( maxLeftSum, maxRightSum,
35                       maxLeftBorderSum + maxRightBorderSum );
36      }
37
38      /**
39       * Driver for divide-and-conquer maximum contiguous
40       * subsequence sum algorithm.
41       */
42      public static int maxSubSum3( int [ ] a )
43      {
44          return maxSumRec( a, 0, a.length - 1 );
45      }
```

图 2-7　算法 3