大小为 N。

```
1       /**
2        * Cubic maximum contiguous subsequence sum algorithm.
3        */
4       public static int maxSubSum1( int [ ] a )
5       {
6           int maxSum = 0;
7
8           for( int i = 0; i < a.length; i++ )
9               for( int j = i; j < a.length; j++ )
10              {
11                  int thisSum = 0;
12
13                  for( int k = i; k <= j; k++ )
14                      thisSum += a[ k ];
15
16                  if( thisSum > maxSum )
17                      maxSum = thisSum;
18              }
19
20          return maxSum;
21      }
```

图 2-5 算法 1

第 2 个循环大小为 $N-i$，它可能要小，但也可能是 N。我们必须假设最坏的情况，而这可能会使得最终的界有些大。第 3 个循环的大小为 $j-i+1$ 我们也要假设它的大小为 N。因此总数为 $O(1\cdot N\cdot N\cdot N)=O(N^3)$。第 6 行总共的开销只是 $O(1)$，而语句 16 和 17 也只不过总共开销 $O(N^2)$，因为它们只是两层循环内部的简单表达式。

事实上，考虑到这些循环的实际大小，更精确的分析指出答案是 $\Theta(N^3)$，而我们上面的估计高 6 倍(不过这并无大碍，因为常数不影响数量级)。一般说来，在这类问题中上述结论是正确的。精确的分析由和 $\sum_{i=0}^{N-1}\sum_{j=i}^{N-1}\sum_{k=i}^{j}1$ 得到，该“和”指出程序的第 14 行被执行多少次。使用 1.2.3 节中的公式可以对该和从内到外求值。特别地，我们将用到前 N 个整数求和以及前 N 个平方数求和的公式。首先有

$$\sum_{k=i}^{j}1 = j-i+1$$

接着，得到

$$\sum_{j=i}^{N-1}(j-i+1) = \frac{(N-i+1)(N-i)}{2}$$

这个和是对前 $N-i$ 个整数求和而计算得出的。为完成全部计算，我们有

$$\begin{aligned}\sum_{i=0}^{N-1}\frac{(N-i+1)(N-i)}{2} &= \sum_{i=1}^{N}\frac{(N-i+1)(N-i+2)}{2}\\ &= \frac{1}{2}\sum_{i=1}^{N}i^2-\left(N+\frac{3}{2}\right)\sum_{i=1}^{N}i+\frac{1}{2}(N^2+3N+2)\sum_{i=1}^{N}1\\ &= \frac{1}{2}\,\frac{N(N+1)(2N+1)}{6}-\left(N+\frac{3}{2}\right)\frac{N(N+1)}{2}+\frac{N^2+3N+2}{2}N\\ &= \frac{N^3+3N^2+2N}{6}\end{aligned}$$