1. 计算问题的复杂度是针对问题的输入规模来说的，0-1 背包问题的输入规模除了物品个数 n 之外，还包括每个物品的重量，价值，背包的容量上限 W 等。0-1 背包问题的解法的复杂度为 O(nW)，是关于 n 和 W 的多项式时间复杂度。但是，输入数据的规模要通过 2 进制的位数来表示，即要用 log2(W)个字符表示出 W，所以设输入规模（长度）为 m，则 W = 2^m，则复杂度是 O(n*2^m)，是指数复杂度，所以 O(nW)的运行时间，可以看做是指数级别的。
2. （1）首先，如果只调用这个多项式时间子例程零次，我们所做的就是多项式的额外工作量，因此整个过程只需要多项式时间。现在，假设调用多项式时间子例程 n+1 次，并且前 n 次调用花费多项式时间。这意味着在最后一次调用之前，所有数据适合的空间是多项式级别的。因此无论传递给最后一个多项式时间子例程的参数是什么，其大小都以某个多项式为界，所以最后一次调用所用的时间也是多项式级别的。所以最终花费的总时间就是两个多项式之和，也是多项式时间。
   （2）取一个例子，让多项式时间子例程是对输入求平方的函数，对其调用多项式次数。假设算法取一个整数 x 为输入，然后将其平方 log(x)次，是多项式次数的调用。但是最终结果的值是 x^(2^logx) = x^x = 2^(logx^(2^logx))，作为输出需要指数级别的 bit 来表示，所以总的时间是指数级别的。
3. 首先 Set partition 问题证明是 NP 问题，取 certificate 为 S 集合两个子集之一，只需要线性的时间把其中的元素加起来求和，并且确定得到的值是 S 中所有元素之和的一半即可，所以是 NP 问题。
   由于 subset sum 问题是 NPC 的，下证 subset sum 问题可以归约到 Set partition 问题。假设 x 等于 S 中所有元素之和，然后在集合 S 中添加 x+t 和 2x-t 形成新的集合 S'，则 S'的所有元素之和为 4x。我们不可能同时取 x+t 和 2x-t，因为它们的和是 3x，即 S 中元素之和的三倍。但是把问题传给 Set partition 问题，划分得到的两个子集之和应该均为 2x，那么取包含 2x-t 的子集，其他元素之和应该恰为 t，且这些元素都属于集合 S，因此它们是 subset sum 问题的解，即把 subset sum 问题归约到了 Set partition 问题。
   所以 Set partition 问题也是 NPC 的。
4. 创建一个和金矿大小相同的 N*M 的二维矩阵 DP[n][m]，其中 DP[i][j]表示矿工到达第 i 行第 j 列时得到的最多的黄金数目。（V[i][j]表示第 i 行第 j 列的格子的黄金数目）
   而得到 DP[i][j]有三种不同的情况：
   1．矿工向右移动到 DP[i][j]，即从 DP[i][j-1]来。那么[i, j]格子的最大值就等于[i, j-1]格子的最大值加上[i, j]格子的黄金数目，即 DP[i][j] = DP[i][j-1] + V[i][j]
   2．矿工向右上移动到 DP[i][j] ，即从 DP[i+1][j-1]来。那么[i, j]格子的最大值就等于[i+1, j-1]格子的最大值加上[i, j]格子的黄金数目，即 DP[i][j] = DP[i+1][j-1] + V[i][j]
   3．矿工向右下移动到 DP[i][j] ，即从 DP[i-1][j-1]来。那么[i, j]格子的最大值就等于[i-1, j-1]格子的最大值加上[i, j]格子的黄金数目，即 DP[i][j] = DP[i-1][j-1] + V[i][j]
   在这三种情况中取最大值，即为 DP[i][j]的值。因此递归关系式如下：
   DP[i][j] = V[i][j] + max(DP[i][j-1], DP[i+1][j-1], DP[i-1][j-1])
   初始化第一列的 DP[i][j] = V[i][j]，然后依次更新，最后取最后一列中的最大值即为全局最大值。