算法 1:

F[i][x]表示到达第 i 个星球，且钻头能力值为 x 的最大收入值。

x 因为是实数，所以要取一定的精度。对于数值范围都在 100 的本题，n 在 10 左右的时候毫无压力。

复杂度：O(nx) x 为精度范围。

期望得分：10-30

算法 2:

F[i][x][y]表示到达第 i 个星球，且之前开采过 x 次，维修过 y 次。

因为本题开采和维修对钻头的影响都是定值。所以钻头能力就是 $w*k^x*c^y$

复杂度:O(n^3)

期望得分：30

算法 3:

对于 20% k=100 的数据，钻头开采一次就永久损坏了。所以只需记录维修过几次即可。

复杂度:O(n^2)

期望得分：20　　（结合算法 2 为 50）

算法 4:

与算法 2 一样的状态设计。但是 x,y 的范围不需要与 n 相同。因为在随机情况下，开采和维修的次数寥寥无几（结合次幂考虑）。

复杂度:O(nyx) x,y 为你自己选择的范围

期望得分：30-80

算法 5:

与算法 2 一样的状态设计，但是使用 DFS 来进行 DP 过程，这样就不会遍历到没有被访问到的状态，同时可以自己加上一些简单的贪心判断来减少状态数量。

复杂度:O(?)

期望得分：70-100

算法 6:

与前 5 种做法截然不同。前 5 种做法的最大瓶颈就是“当前钻头能力”，下面我们尝试不存储“当前钻头能力”。

F[i]表示前 i 个星球的最优收入。很明显这是不行的，因为当前钻头能力会切实影响到后面的过程，不严谨的说，当前钻头能力有“后效性”。

但是这个当前钻头能力对后程的影响无非就是乘上一个数值。（就好像初始钻头能力为 w，实际上你可以按 1 来做，最后再把 ans 乘上 w）。

正难则反，F[i]表示第 i--n 个星球的最优收入，且假设从第 i 个星球开始时钻头能力为 1。

转移过程就变得简单：如果在第 i 个星球开采，那么第 i+1--n 个星球的初始钻头能力就是 1*(1-0.01k)。换句话说，就是 F[i+1]*(1-0.01k)。

所以 F[i]=max{F[i+1],F[i+1]*(1-0.01k)+a[i]}

对于维护型星球，大同小异。就系数和代价的正负而已。

复杂度:O(n)

期望得分：100