挑战性题目DSCT601：背包问题

问题描述：有一个容量为V的背包，要求往背包中装入价值尽可能多的物品。这些物品分别有两个属性：体积w和价值v，且每种物品至多有一个，背包可以不被装满，求装入后的最大价值。

|  |
| --- |
| input.txt |
| 4 5  1 2  2 4  3 4  4 5 |

输入输出格式：

以文件方式读入数据，文件名以参数形式给出。输入文件中，第一行为两个整数N、V，用空格隔开，表示背包的容量V与物品的数量N。接下来N行每行由两个整数wi与vi组成，用空格隔开，表示物品i的体积和价值。请以printf()方式输出最大价值即可。

约定0 < N, V ≤ 1000000，且0 < wi,vi ≤ 4294967296

样例输入：DSCT601\_2018270103012.exe input.txt

样例输出：8

思路分析[[1]](#footnote-1)与算法描述[[2]](#footnote-2)：

本问题为经典的0-1背包问题，可以用动态规划的思路进行求解。

本问题中有两个因素会影响最终的取值，分别是背包的容积以及放入的物品编号。对于每一个物品而言，我们都可以选择放或不放，而这就将影响最终的结果。不妨按照顺序考虑的思路，考虑完1号物品后考虑2号，接着考虑3号…那么我们可以在考虑第i件物品的时候，假定前面的i-1件物品已经考虑好了，前i-1件都有可能被选中或者没有选中。如果此时我们假定前i-1个物品的选择都已经满足最优，即在给定的容积下，这样的前i-1件物品的选择可以使背包内物品的总价值最大，那么容易发现此时该问题满足最优子结构与无后效性的性质，即考虑第i件(即当前物品)并做决策时，可以直接基于以往的最优决策。因此，可以使用动态规划进行考虑。

从上述论述中可以看出，自变量(或状态)应该设为物品编号i与背包容积v。而函数值Bag[i][v]即为：在背包容积为v的情况下，纳入前i件可能的物品后总的背包价值。是否纳入当前物品可以由如下判断得出：若在预先为第i件物品留出vi的容积的情况下，装入前i-1件可能物品的价值大于直接在当前容积只装入前i-1件可能物品的价值，那么我们则将第i件物品装入背包。具体表达式如下：



下面进行优化。由于当我们更新Bag[i]的时候，只需要Bag[i-1]的数据，故我们可以将上述数组化为一维数组。此时，为了在更新时不改变未被更新的值，我们需要从数组的尾端向前做更新，也即先从背包容积大的情况更新起。这种做法在逻辑上并没有什么特殊之处，只是为了能够将所需的空间降低至的复杂度。当然，算法的时间复杂度将一直是。

[【动态规划】01背包问题 - 弗兰克的猫 - 博客园 (cnblogs.com)](https://www.cnblogs.com/mfrank/p/10533701.html)

[动态规划之背包问题系列 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/93857890)

1. 算法思路不但要给出解题的算法内涵，还应该分析该算法的时间复杂度。此页背面也可以答题，但不应续页。 [↑](#footnote-ref-1)
2. 代码的写作和测试建议使用GCC、G++等通用C/C++编译器进行编译以利于跨平台的性能测试。作业纸质版本和测试的可执行文件请提交给主管助教。测试样例命令格式为：DSCT601\_2018270103012.exe input.txt。 [↑](#footnote-ref-2)