逻辑航线信息学奥赛系列教程

背包专题之01背包

一本通 1267：01背包问题

题目描述

一个旅行者有一个最多能装 M 公斤的背包，现在有 n 件物品，它们的重量分别是W1，W2，...,Wn,它们的价值分别为C1,C2,...,Cn，求旅行者能获得最大总价值。

输入

第一行：两个整数，M(背包容量，M<=200)和N(物品数量，N<=30)；

第2..N+1行：每行二个整数Wi，Ci，表示每个物品的重量和价值。

输出

仅一行，一个数，表示最大总价值。

输入样例

```
10 4
2 1
3 3
4 5
7 9
```

输出样例

```
12
```

解析

任何背包类问题的本质都是**是否将物品装入背包**，如果能够理解这个本质，就能够解决一切背包问题。我们将这个例题抽象成图表，具体内容如下，我们需要在空白处填入装入第i件物品后的最大价值。

物品信息			容量
索引	重量	价值	10
0	0	0	0
1	2	1	
2	3	3	
3	4	5	
4	7	9	

首先，尝试装入第一个物品。它的重量是2，小于背包上限，因此可以装入。所以，最终价值是1。如下图所示：

物品信息			容量
索引	重量	价值	10
0	0	0	0
1	2	1	1
2	3	3	
3	4	5	
4	7	9	

接下来，我们装入第2个物品，最终价值为4。装入第3个物品后，最终价值为9。如下图所示：

物品信息			容量
索引	重量	价值	10
0	0	0	0
1	2	1	1
2	3	3	4
3	4	5	
4	7	9	

物品信息			容量
索引	重量	价值	10
0	0	0	0
1	2	1	1
2	3	3	4
3	4	5	9
4	7	9	

现在，我们开始放入第4个物品。这时，出现一个问题，那就是此时背包中只剩下1个容量了，是无法装入第4个物品的。那么我们到底装不装入它呢，该如何抉择呢？

很明显，如果装入第4个物品后，整体的价值比当前的最大价值大，那么我们就肯定选择它。

那么装入第4个物品后的价值该如何计算呢？通过分析，我们能够得出这样一个结论：假如我们能够知道装入第4件物品前整个背包的最大价值，那么就能够求出装入第4个物品后的整体价值。

但是，我们并不知道装入第4件物品前整个背包的最大价值……

再来分析一下，我们可以知道，装入第4件物品前的最大价值，应该出现在背包容量只使用了3个重量的时刻。用表格描述如下：

物品信息			背包容量										
索引	重量	价值	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	2	1											1
2	3	3											4
3	4	5											9
4	7	9											

其中，红色的格子就是装入第4件物品前整个背包的最大价值。

另外一个问题是为什么它出现在第3行呢？这是因为第3行代表的是已经对前三个物品进行了选择，是最优的结果。以此类推，第2行代表的是已经对前两个物品进行了选择。

现在，我们需要计算出红色格子的数值。

物品信息			背包容量										
索引	重量	价值	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	2	1											
2	3	3											
3	4	5											
4	7	9											

首先，第0行的全部数值肯定都为0，因为什么都没有装入嘛。现在，表格中的空白位置就是需要我们计算填入的数据，每一个空格都代表在那个时刻整个背包的最大价值，而最终的最大价值就是这个表格中的最后一个格子。

在前面叙述中，我们知道问题的关键是物品的装入与否，那么我们现在开始依次尝试装入。

首先，是第一个物品。它的重量是2，价值是1。很明显，在背包容量分别为0和1时，由于它的重量超过了背包的容量，因此无法填入，此时背包的价值就是0，结果如下：

物品信息			背包容量										
索引	重量	价值	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	2	1	0	0									
2	3	3											
3	4	5											
4	7	9											

当背包容量大于等于2时，物品1可以被装入，又因为每件物品只能装入1个，因此后续所有表格的价值均为当前物品的价值，即为1，如下所示：

物品信息			背包容量										
索引	重量	价值	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1	1	1
2	3	3											
3	4	5											
4	7	9											

在上面的装载过程中，我们其实忽略了一个问题，那就是当背包装入物品1的时候，其实还有很多空余，这些空余该怎么处理呢？我们继续往下看，现在，我们尝试装入第二个物品。

第2个物品的重量是3，很明显，当背包的容量小于3时，这个物品是无法被装入的。那么此时背包的价值就应该等同于只装入物品1时的价值，图示如下：

物品信息			背包容量										
索引	重量	价值	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1	1	1
2	3	3	0	0	1								
3	4	5											
4	7	9											

当背包的容量大于等于3的时候，物品2存在了两种选择：放或者不放。如何决定是否放入呢？

这个决定必然是背包的最大价值，我们需要比较物品2在放与不放的两种状态下，哪一种能够使背包达到最大的价值，然后选择这一种即可。即：

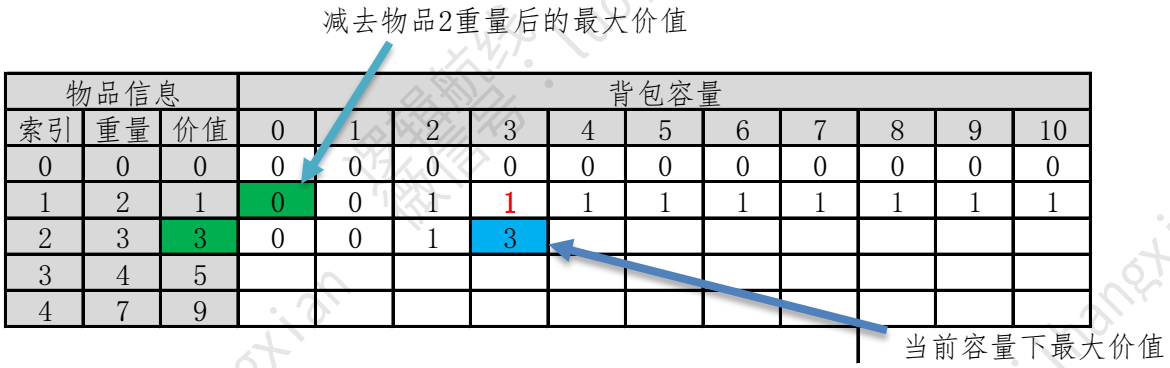
背包的最大价值 = max(不放置物品2的最大价值，放置物品2的最大价值)

先来考虑不放置的情况，很容易想到，如果不放入物品2，那么背包的价值就等价于只装入物品1的价值。如下图：

物品信息			背包容量										
索引	重量	价值	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1	1	1
2	3	3	0	0	1	1	1	1	1	1	1	1	1
3	4	5											
4	7	9											

再来考虑，如果放入物品2，那么计算会发生什么样的变化呢？

如同最初时刻的问题一样，我们需要找到没有放入物品2前的背包最大价值，很明显，它位于第1行第0列，即下图中蓝色箭头标记的位置。



物品信息			背包容量										
索引	重量	价值	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1	1	1
2	3	3	0	0	1	3							
3	4	5											
4	7	9											

上图中，两个绿色分别是未装入第2个物品时背包的最大价值以及第2个物品的价值，它们的数值和3，大于只装入物品1时的最大价值1，因此我们应该装入第2个物品。

总结一下这个状态转移方程，我们设：

dp[i][j]为装入前i个物品在容量j时能取得的最大价值

w[i]为第i个物品的重量

v[i]为第i个物品的价值

则有：

```
        //如果当前的背包的容量小于目标物品的重量，则价值等于前一个物品的价值
        if (j < w[i])
            dp[i][j] = dp[i - 1][j];
        else
            dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - w[i]] + v[i]);
```

按照上面这个方程，我们将表格填写完整

物品信息			背包容量										
索引	重量	价值	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1	1	1
2	3	3	0	0	1	3	3	4	4	4	4	4	4
3	4	5	0	0	1	3	5	5	6	8	8	9	9
4	7	9	0	0	1	3	5	5	6	9	9	10	12

最终，整个背包的最大价值即为12。

编码

```
#include<bits/stdc++.h>

using namespace std;

int bagV, n;

int w[31];              //商品的体积
int v[31];              //商品的价值

//动态规划表
int dp[31][201] = {{0}};

int main() {

    //记录最大承重和物品数量
    cin >> bagV >> n;
    //记录每个物品的重量和价值
    for (int i = 1; i <= n; i++) {
        cin >> w[i] >> v[i];
    }

    //从放入第一件物品开始
    for (int i = 1; i <= n; i++) {
        //从第一个格子开始尝试
        for (int j = 1; j <= bagV; j++) {
            //如果当前的格子的重量小于目标物品的重量，则价值等于前一个物品的价值
            if (j < w[i])
                dp[i][j] = dp[i - 1][j];
            else
                dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - w[i]] + v[i]);
        }
    }

    //01背包的最大值在最后一个格子中
```

```
    cout << dp[n][bagV];

    return 0;
}
```

逻辑航线培优教育，信息学奥赛培训专家。

扫码添加作者获取更多内容。