背包专题之01背包一维优化

	容量		0	1	2	3	4
	重量	价值	价值	价值	价值	价值	价值
无	0	0	0	0	0	0	0
物品1	1	1500					
物品2	4	3000					
物品3	3	2000					

我们观察这个表格，在填写的时候，我们需要通过上一行的数据(黄色格子)，计算出当前行的某一个容量值(绿色格子)。其他空间则被完全的浪费掉了。那么，我们能不能把i-1行的数据拷贝到第i行后，再进行计算呢？

我们现在开始尝试，首先建立如下表格。

	容量		0	1	2	3	4
	重量	价值	价值	价值	价值	价值	价值
物品	0	0	0	0	0	0	0

这样，当我们想计算一个物品的价值是，那么我们就可以直接在这些格子中进行计算。如下所示，我们现在已经计算好了选择物品1后各容量的最大价值。

	容量		0	1	2	3	4
	重量	价值	价值	价值	价值	价值	价值
加入物品1	1	1500	0	1500	1500	1500	1500

现在，我们需要计算加入物品2后，容量为4时的最大价值。音响的重量为4，正好等于容量。另外，物品2的价值为3000，比1500大，因此，我们只需要直接用3000替换这个1500就可以。并且，替换之后也满足我们的题意，即最后一个格子为最大价值。如下所示：

	容量		0	1	2	3	4
	重量	价值	价值	价值	价值	价值	价值
加入物品2	4	3000	0	1500	1500	1500	3000

当然，如果存在剩余容量空间，我们只需要和01背包一样向前寻找剩余空间的价值，然后加在总和处即可。

现在，我给出01背包一维数组的转移方程：

状态转移方程：dp[j] = 两者中较大的一个 —— dp[j]

v[i] + dp[j-w[i]]

压缩成一维数组以后，另一个比较重要的事情就是遍历顺序。

01背包的遍历顺序是从右向左！为什么是这样呢。仔细观察下图，这是原有的01背包图像：

	容量		0	1	2	3	4
	重量	价值	价值	价值	价值	价值	价值
无	0	0	0	0	0	0	0
物品1	1	1500					
物品2	4	3000					
物品3	3	2000					

很明显，绿色的部分是来源于上一行的数据，即黄色的部分。当我们把数组合并成一维之后，如果按照从左到右的顺序进行填写，那么就会出现绿色格子在尚未计算之前，黄色的数据就被修改了，导致结果错误，如下图所示：

	容量		0	1	2	3	4
	重量	价值	价值	价值	价值	价值	价值
加入物品	1	3000	0	3000	6000	0	0

在上面的示例中，按照从左向右的顺序计算后，容量2处又重复累加了一次3000，导致数据错误。

最后，我们将1267进行一维数组的改写

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
int f[201] = {0};     //动态规划表

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++) {
        //从后向前滚动
       for (int j = bagV; j >= w[i]; j--) {
            //使用一维数组进行优化
          f[j] = max(f[j], f[j - w[i]] + v[i]);
        }
    }

    //01背包的最大值在最后一个格子中
    cout << f[bagV];

    return 0;
}
```