背包专题之完全背包一维优化

其实在讲解01背包的一维优化时，已经透露过一点点完全背包的优化方式。还记得01背包的一维遍历顺序吗？

对，就是从右向左，因为如果01背包从左向右来遍历就会把某样东西重复的放入。等等，重复放入，这不正是完全背包所需要的吗？

来看代码：

```
//从放入第一件物品开始
for (int i = 1; i <= n; i++) {
   //从前向后滚动，从当前物品的重量开始判断
      for (int j = w[i]; j <= bagV; j++) {
       //使用一维数组进行优化
            f[j] = max(f[j], f[j - w[i]] + v[i]);
   }
}
```

编码

现在，我们将1268进行重写

```
#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++) {
       //从前向后滚动
            for (int j = w[i]; j <= bagV; j++) {
          //使用一维数组进行优化
                  f[j] = max(f[j], f[j - w[i]] + v[i]);
      }
   }

   //完全背包的最大值在最后一个格子中
      cout << "max=" << f[bagV];

   return 0;
}
```