1295：装箱问题

题目描述

有一个箱子容量为V（正整数，0≤v≤20000），同时有n个物品（0< n ≤30），每个物品有一个体积（正整数）。要求n个物品中，任取若干个装入箱内，使箱子的剩余空间为最小。

输入

第一行是一个整数V，表示箱子容量。

第二行是一个整数n，表示物品数。

接下来n行，每行一个正整数（不超过10000），分别表示这n个物品的各自体积。

输出

一个整数，表示箱子剩余空间。

输入样例

```
24
6
8
3
12
7
9
7
```

输出样例

```
0
```

解析

01背包的变体，我们现在来找出它与01背包原始模板的对应关系。

背包的最大容量：箱子的最大容量
背包的最大价值：箱子的最大容量
物品的价值：每个物品的体积
物品的重量：每个物品的体积

看出来了吗？这道题其实就是在求如何组合能达到箱子的最大容量，因为物品的体积也是价值，所以无需进行额外的判断，直接套用01背包的代码即可。

编码

二维数组代码

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

using namespace std;

int bagV, n;

int w[31];     //最多30个物品的体积
int v[31];     //最多30个物品的价值

int dp[31][20001] = {{0}}; //30个物品  20000容量

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];
        //体积就是价值
        v[i] = w[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 << bagV - dp[n][bagV];
    return 0;
}
```

一维优化

```
#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 dp[20001] = {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] = w[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 = bagV; j >= 1; 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[j] = dp[j];
            else
                dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
        }
    }

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

    return 0;
}
```