动态规划-01背包：

https://www.nowcoder.com/practice/fd55637d3f24484e96dad9e992d3f62e?tpId=230&tqId=38964&ru=/exam/oj

```cpp
#include<iostream>
#include<cstring>
using namespace std;
const int N = 1010;
int n, V, v[N], w[N];
int dp[N][N];

int main() {
    cin >> n >> V;
    for (int i = 1; i <= n; i++) {
        cin >> v[i] >> w[i];
    }
     for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= V; j++) {
            dp[i][j] = dp[i - 1][j]; // 不选当前物品
            if (j >= v[i]) { // 确保背包容量足够容纳当前物品
                dp[i][j] = max(dp[i][j], w[i] + dp[i - 1][j - v[i]]);
            }
        }
    }
    cout << dp[n][V] << endl;


    //2.
    memset(dp, 0, sizeof dp);  // 将 dp 数组全部初始化为 0
    for (int j = 1; j <= V; j++) dp[0][j] = -1; // 初始化第一行
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= V; j++) {
            dp[i][j] = dp[i - 1][j]; // 不选当前物品
            if (j >= v[i] && dp[i - 1][j - v[i]] != -1) { // 确保 j - v[i] 不越界且状态有效
                dp[i][j] = max(dp[i][j], w[i] + dp[i - 1][j - v[i]]);
            }
        }
    }
   if(dp[n][V] == -1) {
        cout << 0 << endl; // 表示无法装入有效的物品组合
    } else {
        cout << dp[n][V] << endl; // 输出最大价值
    }
```

```
40
41      return 0;
42  }
```