二维费用背包问题

问题简介：

二维费用的背包问题是指：对于每件物品，具有两种不同的费用，选择这件物品必须同时付出这两种费用。对于每种费用都有一个可付出的最大值（背包容量）。问怎样选择物品可以得到最大的价值。
设第 i 件物品所需的两种费用分别为 Ci 和 Di。两种费用可付出的最大值（也即两种背包容量）分别为 V 和 U。物品的价值为Wi。

$first$ 三维数组解法。

```
#include <bits/stdc++.h>
using namespace std;
const int max_v = 110, maxn = 1010;
int f[maxn][max_v][max_v];
int main()
{
    int M, V, n;
    cin >> n >> V >> M;
    for (int i = 1; i <= n; i++)
    {
        int v, m, w;
        cin >> v >> m >> w; //体积,重量,价值。
        //状态怎么卡迁移?
        //只是用二维数组的情况下，为了防止数据丢失必须注意数据迁移的顺序性。
        for (int j = 0; j <= M; j++)
            for (int k = 0; k <= V; k++)
            {
                f[i][j][k] = f[i - 1][j][k];
                if (j >= m && k >= v)
                    f[i][j][k] = max(f[i][j][k], f[i - 1][j - m][k - v] + w);
            }
    }
    cout << f[n][M][V] << '\n';
}
```

优化成二维数组的关键

- 枚举过程中f[i][j],第一层i:M->m。改变的都是较大M值得dp也就是说。这个顺序下去，后续计算中得数据不会丧失。

```
#include <bits/stdc++.h>
using namespace std;
const int max_v = 110;
int f[max_v][max_v];
int main()
{
    int M, V, n;
    cin >> n >> V >> M;
```

```
    for (int i = 1; i <= n; i++)
    {
        int v, m, w;
        cin >> v >> m >> w; //体积,重量,价值。
        //状态怎么卡迁移?
        //只是用二维数组的情况下，为了防止数据丢失必须注意数据迁移的顺序性。
        for (int j = M; j >= m; j--)
            for (int k = V; k >= v; k--)
                f[j][k] = max(f[j][k], f[j - m][k - v] + w);
    }
    cout << f[M][V] << '\n';
}
```

可能出现得困惑：

- 怀疑答案得正确性：
    - 这个疑惑前面就存在了。并做出一定得解答。看最基础得几个背包问题即可。