混合背包问题简介：

> 如果将前面1、2、3中的三种背包问题混合起来。也就是说，有的物品只可以取一次（01 背包），有的物品可以取无限次（完全背包），有的物品可以取的次数有一个上限（多重背包)。同样问，众多合法方案中的价值总和最大值。

解决思路

- 面对什么类型的物品选择，就采取什么迁移方案就行了。
- 结论是：按照正确的迁移方程，从小到大的迁移过程中，所得到的更小规模的子问题的解都是正确的。意味着，该问题，是所有合法方案物品价值和的一个取max的结果。

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1010;
int f[maxn];
struct node
{
    int w;
    int v;
};
int main()
{
    int n, m;
    cin >> n >> m;
    for (int i = 1; i <= n; i++)
    {
        int w, v, s;
        cin >> w >> v >> s; // wast , value ,sum;
        if (s < 0)          //一样的转移方程。一样的处理
            for (int j = m; j >= w; j--)
                f[j] = max(f[j], f[j - w] + v);
        else if (s == 0)
            for (int j = w; j <= m; j++)
                f[j] = max(f[j], f[j - w] + v);
        else
        {
            vector<node> a;
            for (int k = 1; k < s; k *= 2)
            {
                s -= k;
                a.push_back({k * w, k * v});
            }
            a.push_back({w * s, s * v});
            for (int t = 0; t < a.size(); t++)
                for (int j = m; j >= a[t].w; j--)
                    f[j] = max(f[j], f[j - a[t].w] + a[t].v);
        }
    }
    cout << f[m] << '\n';
}
```