**01:**

**f[i][v]=max{f[i-1][v],f[i-1][v-c[i]]+w[i]}**

**for i=1..N**

**for v=V..0**

**f[v]=max{f[v],f[v-c[i]]+w[i]};**

**2 wanquan:**

**f[i][v]=max{f[i-1][v-k\*c[i]]+k\*w[i]|0<=k\*c[i]<=v}**

**for i=1..N**

**for v=0..V**

**f[v]=max{f[v],f[v-cost]+weight}**

**3 duochong:**

**有N种物品和一个容量为V的背包。第i种物品最多有n[i]件可用，每件费用是c[i]，价值是w[i]。**

**f[i][v]=max{f[i-1][v-k\*c[i]]+k\*w[i]|0<=k<=n[i]}**

**4 hunhe:**

如果将[P01](http://love-oriented.com/pack/P01.html)、[P02](http://love-oriented.com/pack/P02.html)、[P03](http://love-oriented.com/pack/P03.html)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）。

for i=1..N

if 第i件物品属于01背包

for v=V..0

f[v]=max{f[v],f[v-c[i]]+w[i]};

else if 第i件物品属于完全背包

for v=0..V

f[v]=max{f[v],f[v-c[i]]+w[i]};

or…

for i=1..N

if 第i件物品属于01背包

ZeroOnePack(c[i],w[i])

else if 第i件物品属于完全背包

CompletePack(c[i],w[i])

else if 第i件物品属于多重背包

MultiplePack(c[i],w[i],n[i])

5 er wei fei yong:

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代价；对于每种代价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。设这两种代价分别为代价1和代价2，第i件物品所需的两种代价分别为a[i]和b[i]。两种代价可付出的最大值（两种背包容量）分别为V和U。物品的价值为w[i]。

**算法**

费用加了一维，只需状态也加一维即可。设f[i][v][u]表示前i件物品付出两种代价分别为v和u时可获得的最大价值。状态转移方程就是：

f[i][v][u]=max{f[i-1][v][u],f[i-1][v-a[i]][u-b[i]]+w[i]}

如前述方法，可以只使用二维的数组：当每件物品只可以取一次时变量v和u采用逆序的循环，当物品有如完全背包问题时采用顺序的循环。当物品有如多重背包问题时拆分物品。这里就不再给出伪代码了，相信有了前面的基础，你能够自己实现出这个问题的程序。

6 :