# 多重背包问题

### 题目

有$N$种物品和一个容量为$V$的背包。第$i$种物品最多有$p[i]$件可用，每件费用是$w[i]$，价值是$v[i]$。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

### 基本思路

这题目和完全背包问题很类似。基本的方程只需将完全背包问题的方程略微一改即可，因为对于第$i$种物品有$p[i]+1$种策略：取0件，取1件···取$p[i]$件。令$f[i][j]$表示前$i$种物品恰放入一个容量为$j$的背包的最大权值，则有状态转移方程：

$$f[i][j]=max(f[i-1][j-k*w[i]]+k*v[i])|0<=k<=p[i]$$

复杂度是$O(V\sum p[i])$。

# 转化为01背包问题

另一种好想好写的基本方法是转化为01背包求解：把第$i$种物品换成$p[i]$件01背包中的物品，则得到了物品数为$\sum p[i]$的01背包问题，直接求解，复杂度仍然是$O(V*\sum p[i])$。但是我们期望将它转化为01背包问题之后能够像完全背包一样降低复杂度。仍然考虑二进制的思想，我们考虑把第$i$种物品换成若干件物品，使得原问题中第$i$种物品可取的每种策略——取$0\cdots p[i]$件——均能等价于取若干件代换以后的物品。另外，取超过$p[i]$件的策略必不能出现。

方法是：将第$i$种物品分成若干件物品，其中每件物品有一个系数，这件物品的费用和价值均是原来的费用和价值乘以这个系数。使这些系数分别为$1,2,4,\cdots,2^{k-1},p[i]-2^k+1$，且$k$是满足$p[i]-2^k+1>0$的最大整数。例如，如果$p[i]$为$13$，就将这种物品分成系数分别为$1,2,4,6$的四件物品。分成的这几件物品的系数和为$p[i]$，表明不可能取多于$p[i]$件的第$i$种物品。另外这种方法也能保证对于$0\cdots p[i]$间的每一个整数，均可以用若干个系数的和表示，这个证明可以分$0\cdots 2^k-1$和$2^k\cdots p[i]$两段来分别讨论得出，并不难，希望你自己思考尝试一下。这样就将第$i$种物品分成了$O(log(p[i]))$种物品，将原问题转化为了复杂度为$O(V*\sum \log(p[i]))$的01背包问题，是很大的改进。二进制拆分代码如下：

In [1]:
```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#ifdef __cplusplus  //曾经的C/C++，使用这个宏
extern "C" {
    using namespace std;
    const int maxn = (int)1e2+5;

    int val[maxn],cst[maxn],sze[maxn],N,V;
    int dp[maxn];

    int max(int a, int b){
        return a>=b?a:b;
    }
    void ZeroOnePack (int cost, int value) {   //01背包   逆序
        for (int i = V; i >= cost; i--) {
            dp[i] = max(dp[i], dp[i-cost]+value);
        }
    }

    void CompletePack (int cost, int value) {  //完全背包  顺序
        for (int i = cost; i <= V; i++) {
            dp[i] = max(dp[i], dp[i-cost]+value);
        }
    }

    void MultiplePack (int idx) {
        if (sze[idx]*cst[idx] >= V) {     //如果装不下，那就是完全背包问题
            CompletePack(cst[idx], val[idx]);  //调用完全背包
            return ;
        }
        int x = 1;
        int num = sze[idx];
        while (x <= num) {
            ZeroOnePack(x*cst[idx], x*val[idx]);
            num -= x;     //num-  1,2,4,8,16.....
            x <<= 1;     //x=1,2,4,8,16.....
        }
        if (num > 0) {  //如果还有剩则在单独考虑一次。
            ZeroOnePack(num*cst[idx], num*val[idx]);
        }
    }
    void print_result(){
        freopen("dp04beibao03_01.in","r",stdin);
        scanf("%d %d",&N,&V);
        for (int i = 0; i < N; i++) {
            scanf("%d %d %d",&cst[i],&val[i],&sze[i]);  //费用，价值，数量
        }
        memset(dp, 0, sizeof(dp));
        for (int i = 0; i < N; i++) {
            MultiplePack(i);           //多重背包
        }
        printf("%d",dp[V]);
    }
}
#endif
```

Out[1]:

In [2]:
```
print_result();
```

```
10
```

Out[2]: (void) nullptr

**例题：**

HDU 1059 Dividing (http://acm.hdu.edu.cn/showproblem.php?pid=1059)
Luogu P1776 宝物筛选 (https://www.luogu.org/problemnew/show/P1776)

In [ ]: