背包动规 2

在上一讲中，我们讲解了 01 背包，同时涉及了 01 背包推演出来的满背包问题和二维背包问题，在这一讲中，我们讲讲多重背包和完全背包。

多重背包

有一个容量为 p 的背包，有 n 种物品，每种物品的重量为 w[i]，价值为 v[i]，有 c[i]个，问怎么挑物品放进背包才能使背包的价值最大。

和 01 背包不同的是，这里给出是物品的种类，每种物品有 c[i]个。对于 01 背包问题，我们对每个物品进行决策时，是选与不选，那么在多重背包中，对于当前物品，除了选和不选之外，还可以选 1 个，选 2 个，一直到选 c[i]个。所以，状态转移方程为：

F[i][j]=max(f[i-1][j-k*w[i]]+k*v[i])|(0<=k<=c[i])
k 表示选第 i 种物品的个数，为 0 时表示一个不选

代码如下：

```
#include<iostream>
using namespace std;
int n,p,c[101],w[101],v[101],f[101][10000];
int main(){
    cin>>p>>n;
    for(int i=1;i<=n;i++)cin>>c[i]>>w[i]>>v[i];
    for(int i=1;i<=n;i++){
        for(int j=0;j<=p;j++){
            for(int k=0;k<=c[i];k++){
                if(j>=k*w[i])f[i][j]=max(f[i][j],f[i-1][j-k*w[i]]+k*v[i]);
            }
        }
    }
    cout<<f[n][p];
}
```

时间复杂度为 O（n*p*max(c[i])）。

当然，以上代码有可以优化的地方。对于第 i 种物品，有 c[i]个，并不表示可以选择到 c[i]个，如果背包只放第 i 个物品，最多可以选择 p/w[i]个，所以，对于第 i 种物品，最多可以放置 min（c[i],p/w[i]）个，也就是说 k 取值范围为 0<=k<=min(c[i],p/w[i])。在空间上，可以压缩为一维数组。

代码如下：

```cpp
#include<iostream>
using namespace std;
int n,p,c[101],w[101],v[101],f[10000];
int main(){
    cin>>n>>p;
    for(int i=1;i<=n;i++)cin>>c[i]>>w[i]>>v[i];
    for(int i=1;i<=n;i++){
        for(int j=p;j>=0;j--){  //倒序循环
            for(int k=1;k<=min(c[i],p/w[i]);k++){
                if(j>=k*w[i])f[j]=max(f[j],f[j-k*w[i]]+k*v[i]);
            }
        }
    }
    cout<<f[p];
}
```

完全背包

给出一个背包，容量为 p，有 n 种物品，每种物品个数不限，每种物品有两个属性，重量 w[i]和价值 v[i]，问怎样挑选物品才能使背包价值最大？

显然，这个问题可以转化成多重背包来处理，虽然题目说物品个数不限，但是物品个数再不限，背包容量毕竟是有限的，那怕整个背包全放第 i 种物品，最多也只能放 p/w[i]个，所以，你可以把每种物品的个数看成是 p/w[i]个，这就是多重背包问题了。如果 w[i]为 1，p/w[i]=p，所有时间复杂度为 O（n*p*p）。

那么，这个问题有可以优化的地方吗？我们观察状态转移方程:
F[i][j]=max(f[i-1][j-k*w[i]]+k*v[i])（0<=k<=p/w[i]）
考虑放第 k 件 i 物品时，状态会变成在 j-w[i]处放 k-1 件 i 物品，但其实在求 f[i][j-w[i]]时，已经求出了对于 j-w[i]放置 k 件 i 物品的最优决策，所有，并不需要对 f[i][j-w[i]]再循环 k 次枚举放置 k-1 件 i 物品的最优方案。所以，对于放第种 i 物品时，决策变得很简单，如果不放，f[i][j]=f[i-1][j]，如果放，f[i][j]=f[i][j-w[i]]+v[i]（在前面的基础上再放一件），两者取最大：

f[i][j]=max(f[i-1][j],f[i][j-w[i]]+v[i])

代码如下：

```cpp
#include<iostream>
using namespace std;
int n,p,w[101],v[101],f[101][10001];
int main(){
    cin>>p>>n;
    for(int i=1;i<=n;i++)cin>>w[i]>>v[i];
    for(int i=1;i<=n;i++){
```

```
        for(int j=0;j<=p;j++){
            f[i][j]=f[i-1][j];//不放
            if(j>=w[i]){ //如果能放，再比较放和不放
                f[i][j]=max(f[i][j],f[i][j-w[i]]+v[i]);
            }
        }
    }
    cout<<f[n][p];
}
```

当然，和 01 背包类似，这个代码还可以再优化，在空间上优化成一维。求 f[i][j]时，如果不放，f[i][j]=f[i-1][j]，在一维的状态下，f[j]一开始就是表示不放的状态。如果放，f[i][j]=f[i][j-w[i]]+v[i]，在这里，更新 f[i][j]，需要同阶段 i 的左边的数据 j-w[i]来更新，所以我们可以用顺序循环来完成，代码如下：

```
#include<iostream>
using namespace std;
int n,p,w[101],v[101],f[10001];
int main(){
    cin>>p>>n;
    for(int i=1;i<=n;i++)cin>>w[i]>>v[i];
    for(int i=1;i<=n;i++){
        for(int j=w[i];j<=p;j++){
            f[j]=max(f[j],f[j-w[i]]+v[i]);
        }
    }
    cout<<f[p];
}
```

注意观察以上代码，神奇的是，这个代码和 01 背包的代码极其类似，只是在第二重循环时逆序改成了顺序，但是同学们要注意比较区分。

逃亡的准备

【题目描述】
在《Harry Potter and the Deathly Hallows》中，Harry Potter 他们一起逃亡，现在有许多的东西要放到赫敏的包里面，但是包的大小有限，所以我们只能够在里面放入非常重要的物品，现在给出该种物品的数量、体积、价值的数值，希望你能够算出怎样能使背包的价值最大的组合方式，并且输出这个数值，赫敏会非常地感谢你。
【输入】
（1）第一行有 2 个整数，物品种数 n 和背包装载体积 v。n<=2000,v<=500
（2）2 行到 i+1 行每行 3 个整数，为第 i 种物品的数量 m、体积 w、价值 s。m<=5000。
【输出】
包含一个整数，即为能拿到的最大的物品价值总和。

【样例输入】
2 10
3 4 3
2 2 5
【样例输出】
13
【分析】
多重背包模板题。
【代码】

```
#include<iostream>
using namespace std;
int n,v,m[2001],w[2001],s[2001],f[2001][510];
int main(){
    cin>>n>>v;
    for(int i=1;i<=n;i++){
        cin>>m[i]>>w[i]>>s[i];
    }
    for(int i=1;i<=n;i++){
        for(int j=1;j<=v;j++){
            for(int k=0;k<=min(m[i],v/w[i]);k++){
                if(j>=k*w[i])f[i][j]=max(f[i][j],f[i-1][j-k*w[i]]+k*s[i]);
            }
        }
    }
    cout<<f[n][v];
}
```

竞赛总分

【题目描述】
学生在我们 USACO 的竞赛中的得分越多我们越高兴。我们试着设计我们的竞赛以便人们能尽可能的多得分。现在要进行一次竞赛，总时间 T 固定，有若干类型可选择的题目，每种类型题目可选入的数量不限，每种类型题目有一个 si(解答此题所得的分数)和 ti(解答此题所需的时间)，现要选择若干题目，使解这些题的总时间在 T 以内的前提下，所得的总分最大。
输入包括竞赛的时间,M(1 <= M <= 10000)和题目类型数目 N(1 <= N <= 10000)。
后面的每一行将包括两个整数来描述一种"题型":
第一个整数说明解决这种题目能得的分数(1 <= points <= 10000),第二整数说明解决这种题目所需的时间(1 <= minutes <= 10000)。
【输入】
第 1 行: 两个整数：竞赛的时间 M 和题目类型数目 N。 第 2-N+1 行: 两个整数：每种类型题目的分数和耗时。
【输出】
单独的一行，在给定固定时间里得到的最大的分数。

【样例输入】
300 4
100 60
250 120
120 100
35 20
【样例输出】
605
【分析】
完全背包模板题。
【代码】

```
#include<iostream>
using namespace std;
int m,n,s[10001],t[10001],f[10001];

int main(){
    cin>>m>>n;
    for(int i=1;i<=n;i++)cin>>s[i]>>t[i];
    for(int i=1;i<=n;i++){
        for(int j=t[i];j<=m;j++){
            f[j]=max(f[j],f[j-t[i]]+s[i]);
        }
    }
    cout<<f[m];
}
```

Coin 金银岛

【题目描述】
在金银岛上，人们使用的货币的值都为完全平方数，例如 1,4,9……289.对于要支付十元的话就有下列四种办法。 1：十个一元的钱。 2：一个四元的，六个一元的。 3：二个四元的，二个一元的。 4：一个九元的，一个一元的。 你的任务在于对于给定的钱数(设其值少于300)，给出有多少种支付的方法.
【输入】
输入共有 n+1 行数(n 未知)，但是输入文件以数字 0 结束，每行为一个自然数 t(1<=t<=300)。
【输出】
输出共有 n 行，每行表示输入文件所对应行自然数 t，可有多少种完全自然数组成数字 t 的方案总数。
【样例输入】
2
10
30
0

【样例输出】
1
4
27
【分析】
其实完全背包的模板，可以理解成是一种递推的方案。设 f[i][j]表示用前 i 种货币能支付 j 钱的方案数，对于 f[i][j]，不使用第 i 种货币，f[i][j]=f[i-1][j]，考虑第 i 中货币 v[i]，在 j-v[i]的基础上放上 v[i]可以达到这个状态，f[i][j]=f[i][j-v[i]]。所以 f[i][j]=f[i-1][j]+f[i][j-v[i]]。
【代码】

```
#include<iostream>
using namespace std;
int f[301],x,v[20];
int main() {
    for(int i=1; i<=17; i++) {
        v[i]=i*i;
    }
    f[0]=1;
    for(int i=1; i<=17; i++) {
        for(int j=v[i]; j<=300; j++) {
            f[j]+=f[j-v[i]];
        }
    }
    while(cin>>x) {
        if(x==0)break;
        cout<<f[x]<<endl;
    }
}
```

质数和分解

【题目描述】
任何大于 1 的自然数 N，都可以写成若干个大于等于 2 且小于等于 N 的质数之和表达式(包括只有一个数构成的和表达式的情况)，并且可能有不止一种质数和的形式。例如 9 的质数和表达式就有四种本质不同的形式：9 = 2+5+2 = 2+3+2+2 = 3+3+3 = 2+7 。
这里所谓两个本质相同的表达式是指可以通过交换其中一个表达式中参加和运算的各个数的位置而直接得到另一个表达式。试编程求解自然数 N 可以写成多少种本质不同的质数和表达式。
【输入】
文件中的每一行存放一个自然数 N，2≤N≤200。
【输出】
依次输出每一个自然数 N 的本质不同的质数和表达式的数目。
【样例输入】
样例输入一：2

样例输入二：200
【样例输出】
样例输出一：1
样例输出二：9845164
【分析】
先预求出 n 以内的质数个数，再用完全背包递推求解。
【代码】

```
#include<iostream>
using namespace std;
int n,a[201],f[201],p[201];
int getprime(int n){
    int len=0;
    for(int i=2;i<=n;i++){
        if(p[i]==0){
            a[++len]=i;
            for(int j=2;j<=n/i;j++){
                p[i*j]=1;
            }
        }
    }
    return len;
}
int main(){
    cin>>n;
    int len=getprime(n);
    f[0]=1;
    for(int i=1;i<=len;i++){
        for(int j=a[i];j<=n;j++){
            f[j]+=f[j-a[i]];
        }
    }
    cout<<f[n];
}
```

僵尸的进攻

【题目描述】
植物大战僵尸这款游戏中，还有一个特别的玩儿法：玩家操纵僵尸进攻植物。
首先，僵尸有 m(m<=100)种（每种僵尸都是无限多的），玩家可以选择合适的僵尸来进攻。使用第 i 种僵尸需要花费 Wi 资源，可以得到 Pi 的攻击效果。在这里，我们认为多个僵尸总的攻击效果就是他们每个攻击效果的代数和。
地图共有 n(n<=200000)行，对于第 i 行，最左端有若干植物，这些植物需要至少 Qi 的攻击才能被全部消灭。若一行上的植物全部被消灭，我们称这一行被攻破。

由于资源紧张，我们希望能够算出攻破所有行总共需要的最少的资源值 K，你能算出 K 值来吗？

【输入】

第一行三个非负整数：m、n；

第二行 m 个正整数，第 i 个数表示 Wi；

第三行 m 个正整数，第 i 个数表示 Pi；

第四行 n 个非负整数，第 i 个数表示 Qi。

【输出】

一个正整数 K。

【样例输入】

3 4

5 2 11

3 1 7

10 3 2 4

【样例输出】

32

（样例说明：需要的最小代价是 16+5+4+7=32。）

【分析】

本题将完全背包反过来用，不是求最大价值，而是求最小价值。

【代码】

```
#include<iostream>
#include<cstring>
using namespace std;
int m,n,w[101],p[101],q[200001],mxn,ans,f[2000001];
int main(){
    cin>>m>>n;
    for(int i=1;i<=m;i++)cin>>w[i];
    for(int i=1;i<=m;i++)cin>>p[i];
    for(int i=1;i<=n;i++){
        cin>>q[i];
        mxn=max(mxn,q[i]);  //最大的攻击数，以最大攻击数为背包容量
    }
    memset(f,127,sizeof(f));
    f[0]=0;
    for(int i=1;i<=m;i++){
        for(int j=1;j<=mxn;j++){
            if(j-p[i]<0)f[j]=min(f[j],w[i]);  //p[i]小于 j，也需要 w[i]的资源
            else f[j]=min(f[j],f[j-p[i]]+w[i]);
        }
    }
    for(int i=1;i<=n;i++){  //循环求出每个植物需要耗费的资源
        ans+=f[q[i]];
    }
    cout<<ans;
```

```
25. }
```

图书问题

【题目描述】
WM：真的和 Mr Zhang 说的一样，一楼是图书馆……
哇塞，有好多书哦。善于观察的 WM 童鞋发现一共有 M 种图书，每种图书都有 N 本，并且每本书都有一个编号(同一种类的图书编号相同，不同种类的图书编号也可能相同，这是什么情况?)。于是
WM 童鞋就开始思考，如果把这些书的编号进行组合，能组合出好多数字。问题也随之而来，用总数不超过 N 本书的编号进行组合，在组合出来的数字中可以连续出现的数字中最多的有多少个?
【输入】
第 1 行为 M 和 N 的值，中间用一个空格隔开；
第 2 行 M 个整数，每两个整数之间用一个空格隔开，第 i 个整数 Num[i]代表第 i 种图书的编号；
【输出】
输出共一行一个整数，在不超过 N 本书的情况下，所组合出来的连续的数字中最多的有多少个数；
【样例输入】
3 4
1 2 4
【样例输出】
14
【样例解释】
{下表为图书编号组合可能情况中的一种}

数字 1	1	数字 9	4+4+1
数字 2	2	数字 10	4+4+2
数字 3	2+1	数字 11	4+4+2+1
数字 4	4	数字 12	4+4+4
数字 5	4+1	数字 13	4+4+4+1
数字 6	4+2	数字 14	4+4+4+2
数字 7	4+2+1	数字 15	无法用≤4 本组合出来
数字 8	4+4	数字 16	4+4+4+4

这里显示数字 15 无法用≤4 本的图书编号组合出来，因此最大的连续集是[1..14]，但是没有理由相信这个连续集非得从 1 开始。
【数据规模】
对于 30%的数据： 1≤M≤20；1≤N≤20；
对于 50%的数据： 1≤M≤50；1≤N≤50；
对于 70%的数据： 1≤M≤100；1≤N≤100；
对于 100%的数据：1≤M≤200；1≤N≤200；1≤Num[i]≤255；
【分析】
本题需要先求出 n 个数能组合成的数，然后在这些组合数中求最长连续区间。

【代码】

```
#include<iostream>
#include<algorithm>
#include<cstring>
using namespace std;
int n,m,a[201],f[60001],ans,sum;
int main(){
    cin>>n>>m;
    for(int i=1;i<=n;i++)cin>>a[i];
    sort(a+1,a+n+1);
    int maxN=m*a[n];
    memset(f,127,sizeof(f));
    f[0]=0;  //f[j]表示达到 j 这个数最少需要几本书
    for(int i=1;i<=n;i++){  //完全背包模板
        for(int j=a[i];j<=maxN;j++){  //放上第 a[i]本，用书量会否更少
            if(f[j-a[i]]+1<=m)f[j]=min(f[j],f[j-a[i]]+1);
        }
    }
    for(int i=1;i<=maxN;i++){  //求最长连续区间
        if(f[i]!=f[60000])sum++;
        else{
            ans=max(ans,sum);
            sum=0;
        }
    }
    ans=max(ans,sum);
    cout<<ans;
}
```