动态规划 2

多重背包优化

给定一个容量为 m 的背包和 n 种物品。每种物品价值为 v_i，重量为 w_i，数量为 c_i，求最多可以得到的价值。

二进制优化

暴力方法为将每种物品转化为 c_i 个独立物品考虑，然后就是一个 01 背包问题。

不难发现任意一个不超过 c_i 的正整数一定用 $1,2,4\cdots 2^n,(c_i-2^{n+1}+1)$ 表示。

于是可以将每种物品拆成 $1,2,4\cdots 2^n,(c_i-2^{n+1}+1)$ 个物品构成的包考虑。

物品总数优化为 $O\left(\sum_{i=1}^n \log {c_i}\right)$，总时间复杂度为 $O\left(m\sum_{i=1}^n \log {c_i}\right)$。

```
const int MAXM=4e4+5,MAXC=2005;
int dp[MAXM],v[MAXC],w[MAXC];
int main()
{
    int n=read_int(),m=read_int(),cnt=0;
    _for(i,0,n){
        int a=read_int(),b=read_int(),c=read_int();
        for(int j=1;j<c;j<<=1){
            v[cnt]=j*a,w[cnt++]=j*b;
            c-=j;
        }
        v[cnt]=c*a,w[cnt++]=c*b;
    }
    _for(i,0,cnt)
    for(int j=m;j>=w[i];j--)
    dp[j]=max(dp[j],dp[j-w[i]]+v[i]);
    enter(dp[m]);
    return 0;
}
```

单调队列优化

考虑滚动背包，有 $\text{dp}_i=\max_{k\le c}(\text{dp}_{i-kw}+kv)$，设 $i=jw+r$，有

$$ \text{dp}_{jw+r}=\max_{k\le c}(\text{dp}_{jw+r-kw}+kv)=\max_{k\le c}(\text{dp}_{(j-k)w+r}-(j-k)v)+jv $$

枚举余数 r，对每个余数 r 维护 $\text{dp}_{aw+r}-av$ 的单调队列即可。时间复杂度 $O(nm)$。

```
const int MAXM=4e4+5,MAXC=2005;
int dp[MAXM];
pair<int,int> que[MAXM];
int main()
{
    int n=read_int(),m=read_int();
    _for(i,0,n){
        int v=read_int(),w=read_int(),c=read_int();
        _for(j,0,w){
            if(m<j)continue;
            int pos1=(m-j)/w,pos2=pos1,head=0,tail=1;
            for(int k=0;pos2>=0&&k<=c;pos2--,k++){
                int t=dp[pos2*w+j]-pos2*v;
                while(head>=tail&&que[head].second<=t)head--;
                que[++head]=make_pair(pos2,t);
            }
            for(;pos1>=0;pos1--){
                while(que[tail].first>pos1)tail++;
                dp[pos1*w+j]=max(dp[pos1*w+j],que[tail].second+pos1*v);
                if(pos2<0)continue;
                int t=dp[pos2*w+j]-pos2*v;
                while(head>=tail&&que[head].second<=t)head--;
                que[++head]=make_pair(pos2,t);
                pos2--;
            }
        }
    }
    enter(dp[m]);
    return 0;
}
```