#include<bits/stdc++.h>

#define llINF 9223372036854775807

#define pi 3.141592653589793//23846264338327950254

#define endl "\n"

#define mp make\_pair

#define pb push\_back

#define ll long long

#define int long long

#define IOS ios::sync\_with\_stdio(0); cin.tie(0); cout.tie(0);

**using** **namespace** std;

**const** ll maxn=1e3+7;

**const** **double** eps=1e-10;

**const** ll mod=1e9+7;

//挺裸的一个多重背包dp

ll w[maxn],v[maxn],num[maxn];

//w[i]代表制作一份第i种糕点需要的面粉数量，v[i]为售出一份第i种糕点获得的收益

//num[i]为第i种糕点从馅料角度看能制作的最大份数，用ai/bi即可得到

int32\_t main()

{

    IOS;

    ll n,m,c0,d0;

    cin>>n>>m>>c0>>d0;

    ll a,b;

**for**(ll i=1;i<=m;i++)

    {

        cin>>a>>b;

        cin>>w[i]>>v[i];

        num[i]=a/b;

    }

    w[m+1]=c0;v[m+1]=d0;num[m+1]=llINF; //增加第m+1种糕点，也就是无馅料的糕点，制作上限为无穷个

    ll dp[13][maxn];    //dp[i][j]代表制作前i种糕点，使用j份面团获得的最大收益

    memset(dp,0,**sizeof**(dp));

**for**(ll i=1;i<=m+1;i++)  //i是当前制作的糕点种类

    {

**for**(ll j=0;j<=n;j++)        //j是当前的初始面团份数

        {

**for**(ll k=0;j-k\*w[i]>=0&&k<=num[i];k++)      //k是当前糕点制作多少份，j-k\*w[i]>=0保证了面团够用，k<=num[i]保证了馅料够用

            {

                dp[i][j]=max(dp[i][j],dp[i-1][j-k\*w[i]]+k\*v[i]);    //多重背包的状态转移方程自己去看书或者别人博客吧.

            }

        }

    }

    cout<<dp[m+1][n]<<endl;

}