①算法分析题3-5 给定n种物品和一背包。物品i的重量是wi，体积是bi，其价值为vi，背包的容量为C，容积为D。问应如何选择装入背包中的物品，使得装入背包中物品的总价值最大？在选择装入背包的物品时，对每种物品i只有两种选择，即装入背包或者不装入背包。不能将物品i装入背包多次，也不能只装入部分的物品i。试设计一个解此问题的动态规划算法，并分析算法的计算复杂性。

解：用m(i,j,k)来表述可选择物品为0，1，2…i在背包容量为j，容积为k时的最优值。得出如下状态转移方程：

m(i,j,k)=max{m(i-1,j,k),m(i-1,j-wi,k-bi)+vi}，j>=wi且k>=bi

m(i,j,k)=m(i-1,j,k)，j<wi或k<bi

代码：

#include <iostream>

using namespace std;

const int maxn=101;

int dp[maxn][maxn][maxn]={0};

int w[maxn]={0};

int b[maxn]={0};

int v[maxn]={0};

int dpFunc(int n,int c,int d)

{

for(int i=1;i<=n;i++)//序号

{

for(int j=1;j<=c;j++)//背包剩余容量

{

for(int k=1;k<=d;k++)//背包剩余容积

{

if(j>=w[i]&&k>=b[i]) dp[i][j][k]=max(dp[i-1][j][k],dp[i-1][j-w[i]][k-b[i]]+v[i]);

else dp[i][j][k]=dp[i-1][j][k];

}

}

}

return dp[n][c][d];

}

int main()

{

int n,c,d;

cin>>n>>c>>d;

for(int i=1;i<=n;i++)

{

cin>>w[i]>>b[i]>>v[i];

}

int res=dpFunc(n,c,d);

cout<<res<<endl;

return 0;

}

②算法实现题3-2 最少硬币问题

问题描述：设有n种不同面值的硬币，各硬币的面值存于数组T[1:n]中。现要用 这些面值的硬币来找钱。可以使用的各种面值的硬币个数存于数组Coins[1:n]中。

编程任务：对于给定的1≤n≤10，硬币面值数组T和可以使用的各种面值的硬币 个数数组Coins，以及钱数m，0≤m≤20001，编程计算找钱m的最少硬币数。

解：代码如下：

#include <iostream>

#include <cstdio>

#include <climits>

#include <cstring>

using namespace std;

const int maxn = 1e4 + 10;

const int INF = INT\_MAX;

int t[maxn], f[maxn], sumt[maxn], sumf[maxn];

int dp[maxn];

int n, S;

int main(){

cin>>n;

cin>>S;

memset(sumt, 0, sizeof(sumt));

memset(sumf, 0, sizeof(sumf));

for(int i=0; i<=n; i++)

dp[i] = INF;

for(int i=1; i<=n; i++)

{

cin>>t[i];

cin>>f[i];

}

for(int i=n; i>0; i--){

sumt[i] = sumt[i+1] + t[i];

sumf[i] = sumf[i+1] + f[i];

}

dp[n+1] = 0;

for(int i=n; i>0; i--){

for(int j=i+1; j<=n+1; j++){

dp[i] = min(dp[i], dp[j]+(S+sumt[i]-sumt[j])\*sumf[i]);

}

}

cout<<dp[1];

return 0;

}