**01背包**

问题描述

　　给定N个物品,每个物品有一个重量W和一个价值V.你有一个能装M重量的背包.问怎么装使得所装价值最大.每个物品只有一个.

输入格式

　　输入的第一行包含两个整数n, m，分别表示物品的个数和背包能装重量。  
　　以后N行每行两个数Wi和Vi,表示物品的重量和价值

输出格式

　　输出1行，包含一个整数，表示最大价值。

样例输入

3 5  
2 3  
3 5  
4 7

样例输出

8

数据规模和约定

　　1<=N<=200,M<=5000.

C++代码：

#include<iostream>

using namespace std;

int w[205];

int v[205];

int wv[205][5005]={0};

int main()

{

int i,j,n,m;

cin>>n;

cin>>m;

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

{

cin>>w[i];

cin>>v[i];

}

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

for(j=1;j<=m;j++)

{

if(w[i]<=j)

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

else

wv[i][j]=wv[i-1][j];

}

cout<<wv[n][m]<<endl;

return 0;

}

C代码：

#include<stdio.h>

int main()

{

int a[5010]={0};

int m,n,i,j,p,q,k;

scanf("%d%d",&n,&m);

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

{

scanf("%d%d",&p,&q);

for(j=m;j>=p;j--)

{

if(a[j]<a[j-p]+q)

{

a[j]=a[j-p]+q;

}

}

// for(k=1;k<=m;k++)

// {

// printf("%d ",a[k]);

// }

// printf("\n");

}

printf("%d\n",a[m]);

return 0;

}

JAVA代码：

import java.io.\*;

import java.util.\*;

import java.math.\*;

import java.text.\*;

public class Main {

void solve() {

int[] d = new int[5005];

int n = in.nextInt(), m = in.nextInt();

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

int w = in.nextInt(), v = in.nextInt();

for (int j = m; j - w >= 0; j--) {

d[j] = Math.max(d[j], d[j - w] + v);

}

}

out.println(d[m]);

}

FastScanner in;

PrintWriter out;

void run(String s) {

if (s == null) {

in = new FastScanner();

} else {

in = new FastScanner(s);

}

out = new PrintWriter(System.out);

solve();

out.close();

}

public static void main(String[] args) {

new Main().run(null);

}

static class FastScanner {

BufferedReader br;

StringTokenizer st;

FastScanner(String file) {

try {

br = new BufferedReader(new FileReader(file));

} catch (FileNotFoundException e) {

e.printStackTrace();

}

}

FastScanner() {

br = new BufferedReader(new InputStreamReader(System.in));

}

public String next() {

while (st == null || !st.hasMoreTokens()) {

try {

st = new StringTokenizer(br.readLine());

} catch (IOException e) {

e.printStackTrace();

}

}

return st.nextToken();

}

boolean hasMoreTokens() {

while (st == null || !st.hasMoreTokens()) {

try {

String s = br.readLine();

if (s == null)

return false;

st = new StringTokenizer(s);

} catch (IOException e) {

e.printStackTrace();

}

}

return true;

}

public int nextInt() {

return Integer.parseInt(next());

}

public long nextLong() {

return Long.parseLong(next());

}

public double nextDouble() {

return Double.parseDouble(next());

}

}

}