**1、动态规划解决 0-1背包 问题**

#include<iostream>

#include<cstdlib>

#include<ctime>

using namespace std;

const int N = 20; // N件物品

const int C = 40; // C是背包的总容量

int main()

{

int value[N + 1] = {0,4,5,1,7,9,7,11,19,23,12,11,43,6,21,16,17,3,8,20,21}; // 价值

int weight[N + 1] = {0,5,7,2,3,2,3,7,13,17,6,4,27,2,11,9,8,1,4,8,9}; // 重量

int m[N + 1][C + 1] = {0}; // m[i][j]表示在背包容量为j的情况下， 前i件物品的最大价值

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

{

for(int j = 1; j <= C; j++)

{

// 递推关系式

if(j < weight[i])

{

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

}

else

{

int x = m[i - 1][j];

int y = m[i - 1][j - weight[i]] + value[i];

m[i][j] = x < y ? y : x;

}

}

}

for(int i = N; i >= 1; i--)

{

for(int j = 1; j <= C; j++)

{

printf("%4d",m[i][j]);//自动对齐

}

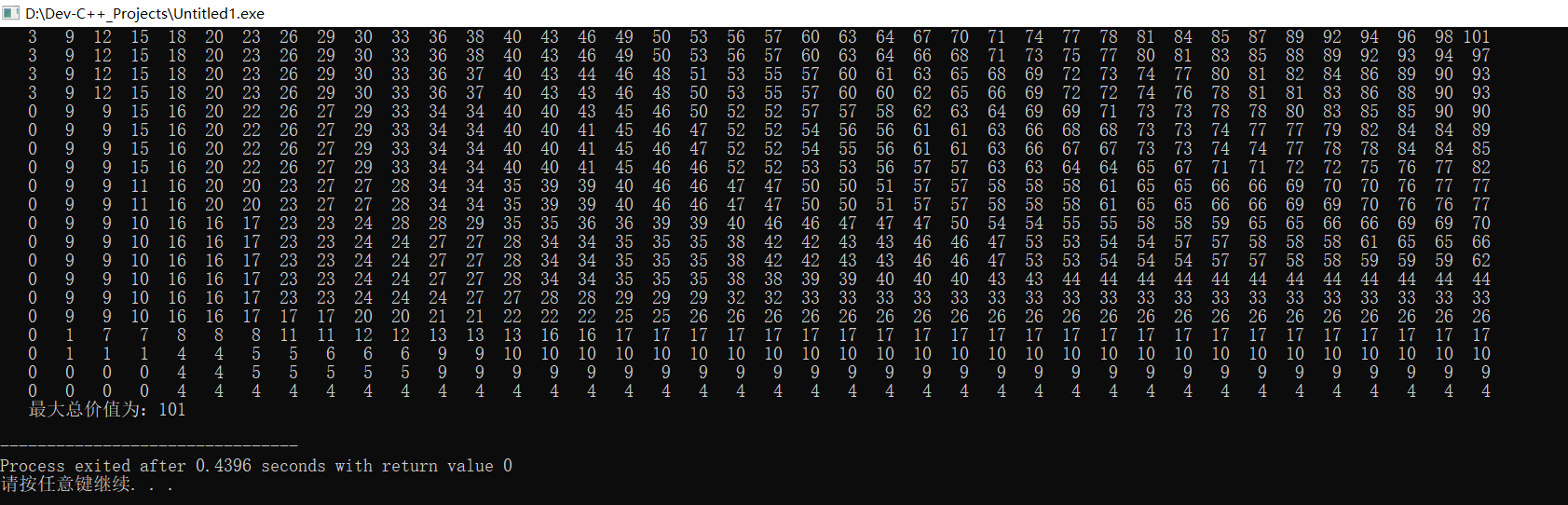
cout << endl;

}

cout<<" 最大总价值为："<<m[20][40]<<endl;

return 0;

}



**2、最大字段和**

#include<iostream>

using namespace std;

//最大字段和

int MaxSum(int a[][20],int n){

int sum = 0,b = 0;

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

if(b > 0){

b += a[0][i];

a[1][i] = b;

}

else{

b = a[0][i];

a[1][i] = b;

}

if(b > sum){

sum = b;

}

a[2][i] = sum;

}

return sum;

}

int main(){

int A[3][20] = {2,11,-4,13,-5,-2,11,-13,25,-32,-12,15,16,13,-24,15,35,-4,-12,15};

int n = 20;

int sum = MaxSum(A,n);

cout<<"a = ";

for(int i = 0;i < 3;i++){

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

printf("%4d",A[i][j]);

cout<<endl;

if(i == 0)

cout<<"b = ";

else if(i == 1)

cout<<"sum = ";

}

cout<<"----------------------------"<<endl;

cout<<"SUM = "<<sum<<endl;

// cout<<MaxSum(n,A)<<endl;

}

