**Задача за раницата**

Дадена е раница с вместимост *M* килограма и *N* предмета, всеки от които се характеризира с две числа - тегло *mi* и стойност *ci*. Да се избере такова множество от предмети, чиято сумарна стойност е максимална, а сумата от теглата не надвишава *M*.

**I начин – пълно изчерпване**

I решение – започваме пресмятането от m докато достигне 0. Предметите се разглеждат последователно, затова освен теглото, във функцията calculate се подава и втори параметър – номера на следващия предмет. Програмата връща само максималната стойност при указаното тегло.

#include <iostream>

using namespace std;

int m[1000], c[1000];

int N, M;

int calculate( int t, int next\_i )

{

int maxst = 0, st;

for ( int i = next\_i; i < N; i++ )

if ( t - m[i] >= 0 )

{

st = calculate( t - m[i], i + 1 ) + c[i];

if (st>maxst) maxst = st;

}

return maxst;

}

int main()

{

cin >> M >> N;

for ( int i = 0; i < N; i++ ) cin >> m[i] >> c[i];

cout << calculate( M, 0 ) << endl;

return 0;

}

I I решение – аналогично на предходния, но започваме пресмятането от 0 докато достигне m. В този случай задаваме стойности на параметрите по подразбиране и не ги задаваме при извикване

#include <iostream>

using namespace std;

int m[1000], c[1000];

int N, M;

int calculate( int t=0, int next\_i=0 )

{

int mx = 0, st;

for ( int i = next\_i; i < N; i++ )

if ( t + m[i] <= M )

{

st = calculate( t + m[i], i + 1 ) + c[i];

if (st>mx) mx = st;

}

return mx;

}

int main()

{

cin >> M >> N;

for ( int i = 0; i < N; i++ ) cin >> m[i] >> c[i];

cout << calculate() << endl;

return 0;

}

**I I начин – динамично оптимиране**

Дефинираме рекурентна целева функция:

*F*(0) = 0;  *F*(*i*) = max { *cj* + *F*(*i - mj*),  *j =* 1, 2, ..., *N*,  *mj ≤ i* }, *i >* 0

Методът на динамичното оптимиране изисква последователно пресмятане на стойностите на *F*(*i*), като за това пресмятане се използват вече пресметнатите стойности за по-малки *i*. Програмата връща само максималната стойност при указаното тегло и предметите, при които се достига тази стойност. Затова използваме масив set, на ред i на който отбелязваме с 1 взетите предмети при тегло i на раницата. Накрая извеждаме на последния ред тези елементи, които са отбелязани с 1.

#include <iostream>

using namespace std;

int N, M, m[1000], c[1000];

int Fn[1000];

bool set[1000][1000];

void calculate()

{ int maxValue, maxIndex, i, j;

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

{

/намираме максималната стойност за тегло i и индекса на взетия предмет

maxValue = maxIndex = 0;

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

if (m[j]<=i && !set[i-m[j]][j])

if (c[j] + Fn[i-m[j]] > maxValue)

{ maxValue = c[j] + Fn[i-m[j]];

maxIndex = j;

}

//предметите, които са били в раницата преди поставянето на последния предмет се намират в ред i-m[maxIndex], затова копираме целия този ред на ред i и добавяме 1 за последния взет предмет

if (maxIndex > 0)

{ Fn[i] = maxValue;

for(int k=0;k<=N-1;k++) set[i][k]=set[i-m[maxIndex]][k];

set[i][maxIndex] = 1;

}

//ако получената максимална стойност за тегло i е по-малка от максималната стойност за тегло i-1, тогава копираме целия ред i-1 на мястото на ред i.

if (Fn[i] < Fn[i-1])

{ Fn[i] = Fn[i-1];

for(int k=0;k<=N-1;k++) set[i][k], set[i-1][k];

}

}

cout << "Max value: " << Fn[M]<<endl;

cout<<"Taken things: ";

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

if (set[M][i]) cout << i << " ";

cout << endl;

}

int main()

{

cin >> M >> N;

for (int i=1; i<=N; i++) cin >> m[i] >> c[i];

calculate();

return 0;

}

/\*

Вход:

7

8

3 5

7 3

6 9

1 1

2 1

4 2

5 5

5 2

Изход: 10

Взети предмети: 3 è 4

\*/

Да разгледаме примера:

N = 8;  
index  1  2  3  4  5  6  7  8  
m[8] 3, 7, 6, 1, 2, 4, 5, 5  
c[8] 5, 3, 9, 1, 1, 2, 5, 2  
M = 8;  
  
Fn[0] = 0;  
Fn[1] = max{c[4]+Fn[0]} = 1;   
 set[1][4]=1; set[1] = {0,0,0,1,0,0,0}  
  
Fn[2] = max{c[4]+Fn[1], c[5]+Fn[0]} = 1  
 set[2][5]=1; set[2] = {0,0,0,0,1,0,0}  
  
Fn[3] = max{c[1]+Fn[0], c[4]+Fn[2], c[5]+Fn[1]} =   
 max{5 +0, 1 +1, 1 +1} = 5  
 set[3][1]=1; set[3] = {1,0,0,0,0,0,0}  
  
Fn[4] = max{c[1]+Fn[1], c[4]+Fn[3], c[5]+Fn[2], c[6]+Fn[0]} =   
 max{5 +1, 1 +5, 1 +1, 2 +0} = 6  
 set[4][1]=1; set[4] = {1,0,0,1,0,0,0}   
  
Fn[5] = max{c[1]+Fn[2],c[4]+Fn[4],c[5]+Fn[3],c[6]+Fn[1],c[7]+Fn[0],c[8]+Fn[0]} =   
 max{5 +1, 1 +6, 1 +5, 2 +1, 5 +0, 2 +0} = 6  
 set[5][1]=1; set[5] = {1,0,0,0,1,0,0}  
  
Fn[6] = max{c[1]+Fn[3],c[3]+Fn[0],c[4]+Fn[5],c[5]+Fn[4],c[6]+Fn[2],c[7]+Fn[2],c[8]+Fn[1]} =   
 max{5 +5, 9 +0, 1 +5, 1 +6, 2 +1, 5 +1, 2 +1} = 9  
 set[6][3]=1; set[6] = {0,0,1,0,0,0,0}  
  
Fn[7] = max{c[1]+Fn[4],c[2]+Fn[0],c[3]+Fn[1],c[4]+Fn[6],c[6]+Fn[5],c[7]+Fn[2],c[8]+Fn[2]} =   
 max{5 +6, 5 +0, 9 +1, 1 +9, 2 +6, 5 +1, 2 +1} = 10  
 set[7][3]=1; set[7] = {0,0,1,1,0,0,0}  
  
Fn[8] = max{c[1]+Fn[5],c[2]+Fn[1],c[3]+Fn[2],c[4]+Fn[7],c[6]+Fn[4],c[7]+Fn[3],c[8]+Fn[3]} =   
 max{5 +6, 5 +1, 9 +1, 1 +10, 2 +6, 5 +1, 2 +1} = 10  
 set[8][3]=1; set[8] = {0,0,1,0,1,0,0}