|  |
| --- |
|  |
| 投资问题 |
|  |

|  |
| --- |
| 计算机1604 王殊 1611640413  2018-11-28 |

# 要求：

求解最大投资问题。

# 思路：

最大投资问题满足最优子结构性质，可以将问题进行分解。但是子问题的求解过程中会有冗余，所以采用动态规划的思想。

用F[ ][ ]数组表示投资当前项目能获得的所有关于项目的最大收益，G[ ][ ]表示单独投资每个项目获得的收益。

核心思想实现：

f = max{f[i - 1][j - k] + g[i][k]}

# 实现：

/\*

问题描述:

设投资总额为 m, 共有 n 个项目 G\_i(X)为向第 i 个项目投资 x 元的收益

问: 如何分配投资以最大化收益

\*/

#include <iostream>

#include <iomanip>

constexpr int N = 3; // 3 个项目

constexpr int M = 9; // 最多 8 元

// 初始化

int G[N][M] = {

{0,5,15,40,80,90,95,98,100},

{0,5,15,40,60,70,73,74,75},

{0,4,26,40,45,50,51,53,53}

};

int F[N][M];

int D[N][M];

void invest(int m, int n, int (\*f)[M], int (\*g)[M], int (\*d)[M])

{

// m 表示投资的额度[0,m]

// n 表示投资的项目

// f 表示要求的在投资序列

// g 项目-预期收益投资关系表

// d 用于最优解, d[i][j] 表示的是在 i 项目投资 j 元 能有最大收益的投资额

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

{

f[0][j] = g[0][j]; // 项目 1 采取投资与 G[1][j] 相同

d[0][j] = j;

}

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

{

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

{

f[i][j] = 0; // 初始化

// 试探投资额

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

{

int s = f[i-1][j-k] + g[i][k]; // 获取在前一个项目采用 j - k 元的投资的最大收益以及项目 i 投资 k 元的收益

if(s > f[i][j])

{

// 比当前的项目的投资的还有好. 前一个项目 j - k 元,当前的项目是 k 元

f[i][j] = s;

d[i][j] = k;

}

}

}

}

}

void print(int (\*p)[M])

{

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

{

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

{

std::cout << std::setw(5) << p[i][j];

}

std::cout << std::endl;

}

}

int main()

{

invest(M-1, N, F,G,D);

std::cout << "F=\n";

print(F);

std::cout << "D=\n";

print(D);

/\*

输出的解释

F=

0 5 15 40 80 90 95 98 100

0 5 15 40 80 90 95 120 140

0 5 26 40 80 90 106 120 140

D=

0 1 2 3 4 5 6 7 8

0 0 0 0 0 0 0 3 4

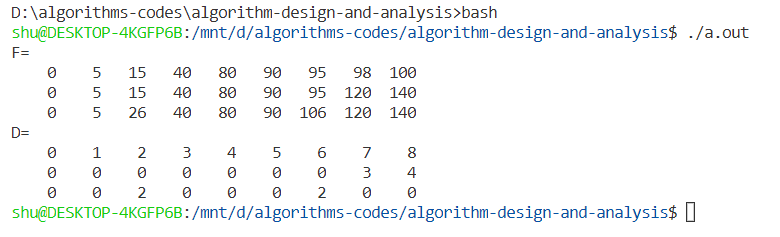
0 0 2 0 0 0 2 0 0

D[i][j] = k 表示在项目 i 值得投入 k 元, 然后从 D[i-1][j - k] 寻找. 例如发现: F[2][8] 最大, 则 G[2][D[2][8]] = 0, G[1][8 - D[2][8]]=G[1][4]= 60,G[0][D[0][8-4]]=G[0][4]= 80 综合 140

\*/

}

# 实验结果：



由F[2][8] = 140 的计算过程，D[2][8] = 0 表示项目3投资0元，D[1][M-0] = D[1][8]=4 表示项2投资4元，D[0][8-4]=D[0][4] = 4 表示项目1投资4元。总计的收益G[0][4] + G[1][4] + G[2][0] = 80 + 60 +0 = 140。