算法设计与分析 (4.29 作业)

智科三班　严中圣　222020335220177

2022 年 4 月 28 日

3.1 用动态规划算法求解下面的组合优化问题，

$$\max \quad g_1(x_1)+g_2(x_2)+g_3(x_3)$$
$$x_1^2+x_2^2+x_3^2 \leqslant 10$$
$$x_1, x_2, x_3 \text{ 为非负整数}$$

其中函数 $g_1(x), g_2(x), g_3(x)$ 的值给在表 3.1 中.

表 3.1 函数值

x	$g_1(x)$	$g_2(x)$	$g_3(x)$	x	$g_1(x)$	$g_2(x)$	$g_3(x)$
0	2	5	8	2	7	16	17
1	4	10	12	3	11	20	22

解.

设 $F_i(y)$ 表示对前 i 个变量求最优解，且满足 $x_1^2+x_2^2+...+x_i^2 \le y$，$x_1, x_2, ..., x_i$ 为非负整数。可以发现对每一个特定的 $g(x)$，函数值都是单调递增的。

故得出递归关系为：

$$F_i(y) = \max_{0 \le x_i \le \lfloor \sqrt{y} \rfloor} F_{i-1}(y - x_i^2) + g_i(x_i)$$
$$F_1(y) = g_1(\lfloor \sqrt{y} \rfloor)$$

迭代实现伪码描述如下：

Algorithm 1 3.1

Require:

Ensure: $\max \quad g_1(x_1)+g_2(x_2)+g_3(x_3)$

```
Initialize F //dp 矩阵，初始全部记为 0
Initialize s //记录最优解下对应变量的取值
for l ← 0 to 10 do
    F[1][l] ← g1(⌊√l⌋)
end for
for i ← 2 to 3 do
    for j ← 0 to 10  do
        maxNum ← 0
        index ← 0
```

for $k \leftarrow 0$ **to** $\lfloor\sqrt{j}\rfloor$ **do**
  if $F[i-1][j-k^2] + g_i(k) > maxNum$ **then**
    $maxNum = F[i-1][j-k^2] + g_i(k)$
    $index = k$
  end if
end for
$F[i][j] = maxNum$
$s[i][j] = index$
end for
end for

根据以上分析编写代码 (见附录) 进行运算，得到备忘录数据和相应最优解数据如下:

F[i][j]	**1**	**2**	**3**
0	2	7	15
1	4	12	20
2	4	14	24
3	4	14	26
4	7	18	26
5	7	20	30
6	7	20	32
7	7	20	32
8	7	23	35
9	11	23	37
10	11	24	37

S[i][j]	**1**	**2**	**3**
0	0	0	0
1	1	1	0
2	1	1	1
3	1	1	1
4	2	2	0
5	2	2	1
6	2	2	1
7	2	2	1
8	2	2	2
9	3	2	2
10	3	3	2

图 1:

根据结果可得：$F_3(10) = 37$，对应的 $x_3 = 2$，故有 $x_1^2 + x_2^2 \leq 6$，又 $F_2(6) = 20, x_2 = 2$，再代入得 $x_1^2 \leq 2$，由 $F_1(2) = 4$，得到 $x_1 = 1$。所以最终问题最优解为 $x_1 = 1, x_2 = 2, x_3 = 2$，此时目标函数最大值为 37.

3.3 有 n 个底面为长方形的货柜需要租用库房存放．如果每个货柜都必须放在地面上，且所有货柜的底面宽度都等于库房的宽度，那么第 i 个货柜占用库房面积大小只需要用它的底面长度 l_i 来表示，$i=1,2,\cdots,n$．设库房总长度是 $D(l_i \leqslant D$ 且 $\sum_{i=1}^{n} l_i > D)$．设第 i 号货柜的仓储收益是 v_i，若要求库房出租的收益达到最大，问如何选择放入库房的货柜？若 $l_1,l_2,\cdots,l_n,D$ 都是正整数，设计一个算法求解这个问题，给出算法的伪码描述并估计算法最坏情况下的时间复杂度．

解.

定义 x_i 表示第 i 个货柜是否放置进库房，$x_i = 1$ 表示放入，建立目标函数为

$$W = \max \sum_{i=1}^{n} v_i x_i \quad , \sum_{i=1}^{n} l_i x_i \le D$$

建立 dp 数组 $M[i][j]$，表示装前 i 个货柜，长度限制为 j 时的最大收益，$i = 1, 2, ..., n$，$j = 1, 2, ..., D$，则递推关系如下所示：

$$M[i][j] = \begin{cases} M[i-1][j] & l_i > j \\ \max\{M[i-1][j], M[i-1][j-l_i] + v_i\} & l_i \le j \end{cases} \quad i > 1$$

$$M[1][j] = \begin{cases} v_1 & l_1 \le j \\ 0 & l_1 > j \end{cases}$$

相应伪码如下：

Algorithm 2 3.3 库房出租问题

Require: $D\ L\ V$ 　库房总长度限制，货柜长度数据，货柜收益数据

Ensure: 最大收益以及货柜选择

$Initialize\ M[n][D]$//dp 矩阵，初始化为 0
$Initialize\ S[n][D]$//结果矩阵，存储最优解方案下的货柜选择
for $j \leftarrow 1$ **to** D **do**
  $M[1,j] \leftarrow V[1]$
end for
for $i \leftarrow 2$ **to** n **do**
  for $j \leftarrow 1$ **to** D **do**
    if $L_i \le j$ **then**
      if $M[i-1][j] < M[i-1][j-l_i] + v_i$ **then**
        $M[i][j] = M[i-1][j-l_i] + v_i$
        $S[i][j] = i$
      else
        $M[i][j] = M[i-1][j]$
        $S[i][j] = S[i-1][j]$
      end if
    else

```
        M[i][j] = M[i-1][j]
        S[i][j] = S[i-1][j]
      end if
    end for
  end for
  return M[n][D]
```

算法最坏情况下的时间复杂度为 $O(nD)$。

3.5 设有 n 种不同面值的硬币，第 i 种硬币的币值是 v_k（其中 $v_1=1$），重量是 w_i，$i=1,2,\cdots,n$ 且现在购买某些总价值为 y 的商品，需要用这些硬币付款，如果每种钱币使用的个数不限，那么如何选择付款的方法使得付出钱币的总重量最轻？设计一个求解该问题的算法，给出算法的伪码描述并分析算法的时间复杂度. 假设问题的输入实例是：

$$v_1=1,\quad v_2=4,\quad v_3=6,\quad v_4=8$$
$$w_1=1,\quad w_2=2,\quad w_3=4,\quad w_4=6$$
$$y=12$$

给出算法在该实例上计算的备忘录表和标记函数表，并说明付线的方法.

解.

根据题意定义 x_i 为第 i 种硬币使用的个数，x_i 为非负整数，则目标函数为

$$W=\min\sum_{i=1}^{n}\omega_i x_i \quad ,\sum_{i=1}^{n}v_i x_i=y$$

设定 dp 矩阵 $W[n][y]$，$W[i][j]$ 表示使用前 i 种硬币，总价值为 j 时的最优解重量。结果矩阵 $S[n][y]$，$S[i][j]$ 表示最优解时选择的硬币的最大标号。递推方程如下：

$$
\begin{gathered}
W[i][j]=\min\{W[i-1][j],W[i][j-v_i]+\omega_i\} \quad ,i>1,0<j\le y\\
W[1][j]=\omega_1 j \quad ,0<j\le q\\
W[i][0]=0 \quad ,0<i\le n\\
W[i][j]=+\infty \quad ,j<0\\
S[i][j]=\begin{cases} i \quad ,W[i-1][j]\ge W[i][j-v_i]+\omega_i\\ S[i-1][j] \quad ,else\end{cases} \quad ,i>1,0<j\le y\\
S[1][j]=1 \quad ,0<j\le y\\
S[i][0]=0 \quad ,1\le i\le n
\end{gathered}
$$

由上伪码如下：

Algorithm 3 3.5 硬币选择问题

Require: ω,V,y $\quad\omega$ 为不同面值硬币的重量，V 不同面值的硬币，y 为付款数

Ensure: 总重量最轻的付款的方法

```
initialize  W[n][y] //dp 数组
initialize  S[n][y] //结果数组
```

```
for j ← 1 to y do
    W[1][j] ← ω_1 j
    S[1][j] ← 1
end for
for i ← 2 to n do
    for j ← 1 to y do
        if W[i-1][j] ≥ W[i][j-v_i] + ω_i then
            W[i][j] = W[i][j-v_i] + ω_i
            S[i][j] = i
        else
            W[i][j] = W[i-1][j]
            S[i][j] = S[i-1][j]
        end if
    end for
end for
return W[n][y]
```

将实例代入上述解法代码求解 (附录 2) 得备忘录和结果矩阵如下所示:

W[i][j]	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	4	5	6	7	8	9	10	11	12
2	1	2	3	2	3	4	5	4	5	6	7	6
3	1	2	3	2	3	4	5	4	5	6	7	6
4	1	2	3	2	3	4	5	4	5	6	7	6

S[i][j]	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	2	2	2	2	2	2	2	2	2
3	1	1	1	2	2	3	3	2	2	3	3	2
4	1	1	1	2	2	3	3	2	2	3	3	2

图 2:

由此可得，最轻重量为 $W[4][12]=6$，$S[4][12]=2$，所以 $x_3=x_4=0, x_2\geq 1$，$S[2][8]=2, S[2][4]=2, S[2][0]=0$，所以 $x_2=3, x_1=0$。故最终解为，$x_1=0, x_2=3, x_3=0, x_4=0$，最轻重量为 6.

附录 1

```cpp
#include <iostream>
#include <cmath>
#include <cstring>
#include <vector>
using namespace std;
int main()
{
    int F[3][11];
    int S[3][11];
    int G[3][4] = {{2, 4, 7, 11}, {5, 10, 16, 20}, {8, 12, 17, 22}};
    memset(F, 0, sizeof(F));
    memset(S, 0, sizeof(S));
    for (int i = 0; i < 11; i++)
    {
        F[0][i] = G[0][(int)sqrt(i)];
    }
    for (int i = 1; i < 3; ++i)
    {
        for (int j = 0; j < 11; j++)
        {
            int max = 0;
            int index = 0;
            for (int k = 0; k <= (int)sqrt(j); k++)
            {
                if (F[i - 1][j - k * k] + G[i][k] > max)
                {
                    max = F[i - 1][j - k * k] + G[i][k];
                    index = k;
                }
            }
            F[i][j] = max;
            S[i][j] = index;
        }
    }
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 11; j++)
        {
            cout << F[i][j] << " ";
        }
        cout << endl;
    }
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 11; j++)
```

```
        {
            cout << S[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

附录 2

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    int W[4][13];
    int S[4][13];
    int V[4] = {1, 4, 6, 8};
    int w[4] = {1, 2, 4, 6};
    memset(W, 0, sizeof(W));
    memset(S, 0, sizeof(S));
    for (int j = 1; j <= 12; j++)
    {
        W[0][j] = j * w[0];
        S[0][j] = 1;
    }
    for (int i = 1; i < 4; ++i)
    {
        for (int j = 1; j <= 12; j++)
        {
            if (W[i - 1][j] >= (W[i][j - V[i]] + w[i]))
            {
                W[i][j] = W[i][j - V[i]] + w[i];
                S[i][j] = i + 1;
            }
            else
            {
                W[i][j] = W[i - 1][j];
                S[i][j] = S[i - 1][j];
            }
        }
    }
    for (int i = 0; i < 4; i++)
    {
        for (int j = 1; j <= 12; j++)
        {
            cout << W[i][j] << " ";
```

```
            }
            cout << endl;
        }
        for (int i = 0; i < 4; i++)
        {
            for (int j = 1; j <= 12; j++)
            {
                cout << S[i][j] << " ";
            }
            cout << endl;
        }
    }
```