ch10 动态规划

动态规划--Dynamic Programming

- 最优控制问题
- 动态规划--求解决策过程(decision process)最优化的数学方法。20世纪50年代初美国数学家R.E.Bellman等人在研究多阶段决策过程(multistep decision process)的优化问题时，提出了著名的最优化原理(principle of optimality)，把多阶段过程转化为一系列单阶段问题，利用各阶段之间的关系，逐个求解，创立了解决这类过程优化问题的新方法——动态规划
- Dynamic—choices depend on the current state, rather than being decided ahead of time
- Main feature—replace an exponential time computation by a polynomial time computation

算法总体思想

- 动态规划算法与分治法类似，其基本思想也是将待求解问题分解成若干个子问题
- 但是适合于用动态规划求解的问题经分解得到的子问题往往**不是互相独立的**。不同子问题的数目常常很多。在用分治法求解时，有些子问题被重复计算了许多次。
- 如果能够保存已解决的子问题的答案，而在需要时再找出已求得的答案，就可以避免大量重复计算，从而得到多项式时间算法。

算法总体思想

- 最优化原理（最优子结构性质）：不论过去状态和决策如何，对前面的决策所形成的状态而言，余下的诸决策必须构成最优策略。简而言之，一个最优化策略的子策略总是最优的。一个问题满足最优化原理又称其具有最优子结构性质
- 子问题的重叠性 动态规划将原来具有指数级时间复杂度的搜索算法改进成了具有多项式时间复杂度的算法。其中的关键在于解决冗余，这是动态规划算法的根本目的。动态规划实质上是一种以空间换时间的技术，它在实现的过程中，不得不存储产生过程中的各种状态，所以它的空间复杂度要大于其它的算法。

10.3 矩阵连乘积

- 矩阵连乘和最优二分搜索树----介绍动态规划的例子
- 矩阵连乘: n个矩阵连乘，确定最优计算顺序，使得数乘次数最少

10.3 矩阵连乘积

- 两个矩阵相乘： $C_{d1\times d3} = A_{d1\times d2} \times B_{d2\times d3}$

```
For(i=1; i<=d1; i++)
    For(j=1; j<=d3; j++)
    {
        c[i][j]=0;
        for(k=1; k<=d2; k++)
            c[i][j]=c[i][j]+a[i][k]*b[k][j];
    }//数乘次数： O(d1×d2×d3)
```

10.3 矩阵连乘积

- n>2个矩阵连乘----确定计算顺序
  - Example10.4：设有四个矩阵A,B,C,D，它们的维数分别是：
  - **A：30×1, B：1×40, C：40×10, D：10×25**
  - ((AB)C)D=30*1*40+30*40*10+30*10*25=20700
  - A(B(CD))=40*10*25+1*40*25+30*1*25=11750
  - (AB)(CD)=30*1*40+40*10*25+30*40*25=41200
  - A((BC)D)=1*40*10+1*10*25+30*1*25=**1400**
- **矩阵乘法满足结合律**
  - 多个矩阵连乘，不同的计算顺序，数乘次数不同；确定最优的计算顺序，可以明显降低矩阵连乘的时间代价

10.3 矩阵连乘积

- 给定n个矩阵 $\{A_1, A_2, \cdots, A_n\}$ ，其中A_i与A_{i+1}是可乘的，$i=1, 2 \cdots, n-1$。如何确定计算矩阵连乘积的计算次序，使得依此次序计算矩阵连乘积需要的数乘次数最少。
- A_i的行数为d_{i-1}，列数为d_i

- **穷举法**：列举出所有可能的计算次序，并计算出每一种计算次序相应需要的数乘次数，从中找出一种数乘次数最少的计算次序。$\Theta((n-1)!)$

10.3 矩阵连乘积--贪心法

- **贪心法**：每次选计算量最少的2个矩阵乘。
- 设有3个矩阵A：40×10, B：10×20, C：20×50
- AB=8000，BC=10000
- 根据设定的贪心策略，计算次序为：

  (AB)C=8000+40000=48000

- 实际上：A(BC)=20000+10000=30000 计算量更少
- 贪心法不保证能得到最优解

分析最优解的结构

- 分析：若计算$A_i \times A_{i+1} \times ...... \times A_j$的最优次序若首先在$A_k$处断开，则其给出的$A_i \times A_{i+1} \times ...... \times A_k$的计算次序也是$A_i \times A_{i+1} \times ...... \times A_k$的最优计算次序，其给出的$A_{k+1} \times A_{k+1} \times ...... \times A_j$的计算次序也是$A_{k+1} \times A_{k+1} \times ...... \times A_j$的最优计算次序
- 即：计算A[i:j]（即：$A_i \times A_{i+1} \times ...... \times A_j$）的最优次序所包含的计算矩阵子链 A[i:k]和A[k+1:j]的次序也是**最优**的。
- 矩阵连乘计算次序问题的最优解包含着其子问题的最优解。这种性质称为**最优子结构性质**。

矩阵连乘积----动态规划

- 给定$\boldsymbol{n}$个矩阵 $\{\boldsymbol{A_1, A_2, \ldots, A_n}\}$ ，其中$\boldsymbol{A_i}$与$\boldsymbol{A_{i+1}}$是可乘的，$\boldsymbol{i}$**=1, 2…,** $\boldsymbol{n}$**-1**。
- $\boldsymbol{d_0, d_1, d_2, \ldots, d_n}$**----(0,**$\boldsymbol{n}$**)**
- 考察计算A[i:j]（即：$A_i \times A_{i+1} \times \ldots\ldots \times A_j$）的最优计算次序。设这个计算次序在矩阵$A_k$和$A_{k+1}$之间将矩阵链断开，$i \leqslant k < j$，则其相应$\boldsymbol{(A_iA_{i+1}\ldots A_k)(A_{k+1}\ldots A_j)}$ **----维数：** $\boldsymbol{(d_{i-1}, d_k), (d_k, d_j)}$
- k的位置

last[i,j]为$A_i \times A_{i+1} \times \ldots\ldots \times A_j$的最优计算顺序最外层的括号所加的位置

矩阵连乘积----动态规划

- 设计算子问题i:j，1≤i≤j≤n，所需要的最少数乘次数cost[i,j]($A_i \times A_{i+1} \times \ldots\ldots \times A_j$的最优计算顺序对应的矩阵元素的数乘次数)，则原问题的最优值为cost[1,n]
- 当i=j时，last[i:j]=-1，因此，cost[i,i]=0，i=1,2,···,n
- 当i<j时，cost[i,j]=cost[i,k]+cost[k+1,j]+$d_{i-1}d_kd_j$
- 如何确定k:

$$\cos t[i,j] = \begin{cases} 0 & i = j \\ \min\limits_{i \le k < j}\{\cos t[i,k] + \cos t[k+1,j] + d_{i-1}d_kd_j\} & i < j \end{cases}$$

- k的位置只有j-i种可能

A1	A2	A3	A4	A5	A6
30×35	35×15	15×5	5×10	10×20	20×25

用动态规划法求最优解

```
void MatrixChain(int *d， int n， int **cost， int **last)
{      for (i = 1; i <= n; i++)  {cost[i][i] = 0; last[i][i]=-1; }
       for (L = 2; L<= n; L++)
         for (i = 1; i <= n - L+1; i++) {
            j=i+L-1;   cost[i][j] = cost[i+1][j]+ d[i-1]*d[i]*d[j];    last[i][j] = i;
           for (k = i+1; k < j; k++) {
                 t=cost[i][k] +cost[k+1][j] + d[i-1]*d[k]*d[j];
                if (t< cost[i][j]) {cost[i][j] =t ;   last[i][j] = k;}
            }
         }
}
```

算法复杂度分析：
算法的计算时间上界为$O(n^3)$。算法所占用的空间显然为$O(n^2)$。

A1	A2	A3	A4	A5	A6
30×35	35×15	15×5	5×10	10×20	20×25

	1	2	3	4	5	6
1	0					
2		0				
3			0			
4				0		
5					0	
6						0

```
for (int i = 1; i <= n; i++) last[i][i] = -1;cost[i][i]=0
```

	1	2	3	4	5	6
1	-1					
2		-1				
3			-1			
4				-1		
5					-1	
6						-1

A1	A2	A3	A4	A5	A6
30×35	35×15	15×5	5×10	10×20	20×25

	1	2	3	4	5	6
1	0	15750				
2		0				
3			0			
4				0		
5					0	
6						0

$$cost[1][2]=cost[1][1]+d_0*d_1*d_2=30*35*15=15750$$

	1	2	3	4	5	6
1	-1	1				
2		-1				
3			-1			
4				-1		
5					-1	
6						-1

A1	A2	A3	A4	A5	A6
30×35	35×15	15×5	5×10	10×20	20×25

	1	2	3	4	5	6
1	0	15750				
2		0	2625			
3			0			
4				0		
5					0	
6						0

$cost[2][3]=cost[2][2]+d_1*d_2*d_3=35*15*5=2625$

	1	2	3	4	5	6
1	-1	1				
2		-1	2			
3			-1			
4				-1		
5					-1	
6						-1

A1	A2	A3	A4	A5	A6
30×35	35×15	15×5	5×10	10×20	20×25

	1	2	3	4	5	6
1	0	15750				
2		0	2625			
3			0	750		
4				0		
5					0	
6						0

$$cost[3][4]=cost[3][3]+d_2*d_3*d_4=15*5*10=750$$

	1	2	3	4	5	6
1	-1	1				
2		-1	2			
3			-1	3		
4				-1		
5					-1	
6						-1

A1	A2	A3	A4	A5	A6
30×35	35×15	15×5	5×10	10×20	20×25

	1	2	3	4	5	6
1	0	15750				
2		0	2625			
3			0	750		
4				0	1000	
5					0	
6						0

$$cost[4][5]=cost[4][4]+d_3*d_4*d_5=5*10*20=1000$$

	1	2	3	4	5	6
1	-1	1				
2		-1	2			
3			-1	3		
4				-1	4	
5					-1	
6						-1

A1	A2	A3	A4	A5	A6
30×35	35×15	15×5	5×10	10×20	20×25

	1	2	3	4	5	6
1	0	15750				
2		0	2625			
3			0	750		
4				0	1000	
5					0	5000
6						0

$$cost[5][6]=cost[5][5]+d_4*d_5*d_6=10*20*25=5000$$

	1	2	3	4	5	6
1	-1	1				
2		-1	2			
3			-1	3		
4				-1	4	
5					-1	5
6						-1

A1	A2	A3	A4	A5	A6
30×35	35×15	15×5	5×10	10×20	20×25

	1	2	3	4	5	6
1	0	15750	7875			
2		0	2625			
3			0	750		
4				0	1000	
5					0	5000
6						0

$$\text{cost}[1][3] = \min \begin{cases} \text{cost}[2][3] + d_0 * d_1 * d_3 = 2625 + 30 * 35 * 5 = 7875 \\ \cos t[1][2] + \cos t[3][3] + d_0 * d_2 * d_3 = 15750 + 0 + 30 * 15 * 5 = 18000 \end{cases}$$

	1	2	3	4	5	6
1	-1	1	1			
2		-1	2			
3			-1	3		
4				-1	4	
5					-1	5
6						-1

A1	A2	A3	A4	A5	A6
30×35	35×15	15×5	5×10	10×20	20×25

	1	2	3	4	5	6
1	0	15750	7875			
2		0	2625	4375		
3			0	750		
4				0	1000	
5					0	5000
6						0

$$\text{cost}[2][4] = \begin{cases} \text{cost}[3][4] + d_1 * d_2 * d_4 = 750 + 35*15*10 = 6000, \\ \text{cost}[2][3] + \text{cost}[4][4] + d_1 * d_3 * d_4 = 2625 + 35*5*10 = 4375 \end{cases}$$

	1	2	3	4	5	6
1	-1	1	1			
2		-1	2	3		
3			-1	3		
4				-1	4	
5					-1	5
6						-1

A1	A2	A3	A4	A5	A6
30×35	35×15	15×5	5×10	10×20	20×25

	1	2	3	4	5	6
1	0	15750	7875			
2		0	2625	4375		
3			0	750	2500	
4				0	1000	
5					0	5000
6						0

$$\text{cost}[3][5] = \begin{cases} \text{cost}[4][5] + d_2 * d_3 * d_5 = 1000 + 15*5*20 = 2500, \\ \text{cost}[3][4] + \text{cost}[5][5] + d_2 * d_4 * d_5 = 750 + 15*10*20 = 3750 \end{cases}$$

	1	2	3	4	5	6
1	-1	1	1			
2		-1	2	3		
3			-1	3	3	
4				-1	4	
5					-1	5
6						-1

A1	A2	A3	A4	A5	A6
30×35	35×15	15×5	5×10	10×20	20×25

	1	2	3	4	5	6
1	0	15750	7875			
2		0	2625	4375		
3			0	750	2500	
4				0	1000	3500
5					0	5000
6						0

$$\mathrm{cost}[4][6] = \begin{cases} \mathrm{cost}[5][6] + d_3 * d_4 * d_6 = 5000 + 5*10*25 = 6250, \\ \mathrm{cost}[4][5] + \mathrm{cost}[6][6] + d_3 * d_5 * d_6 = 1000 + 5*20*25 = 3500 \end{cases}$$

	1	2	3	4	5	6
1	-1	1	1			
2		-1	2	3		
3			-1	3	3	
4				-1	4	5
5					-1	5
6						-1

A1	A2	A3	A4	A5	A6
30×35	35×15	15×5	5×10	10×20	20×25

	1	2	3	4	5	6
1	0	15750	7875	9375		
2		0	2625	4375		
3			0	750	2500	
4				0	1000	3500
5					0	5000
6						0

$$\text{cost}[1][4]=\begin{cases}\text{cost}[2][4]+d_0*d_1*d_4=4375+30*35*10=14875\\ \text{cost}[1][2]+\text{cost}[3][4]+d_0*d_2*d_4=15750+750+30*15*10=21000\\ \text{cost}[1][3]+\text{cost}[4][4]+d_0*d_3*d_4=7875+30*5*10=9375\end{cases}$$

	1	2	3	4	5	6
1	-1	1	1	3		
2		-1	2	3		
3			-1	3	3	
4				-1	4	5
5					-1	5
6						-1

A1	A2	A3	A4	A5	A6
30×35	35×15	15×5	5×10	10×20	20×25

	1	2	3	4	5	6
1	0	15750	7875	9375		
2		0	2625	4375	7125	
3			0	750	2500	
4				0	1000	3500
5					0	5000
6						0

$$\text{cost}[2][5] = \begin{cases} \text{cost}[3][5] + d_1 * d_2 * d_5 = 2500 + 35*15*20 = 13000 \\ \text{cost}[2][3] + \text{cost}[4][5] + d_1 * d_3 * d_5 = 2625 + 1000 + 35*5*20 = 7125 \\ \text{cost}[2][4] + \text{cost}[5][5] + d_1 * d_4 * d_5 = 4375 + 0 + 35*10*20 = 11375 \end{cases}$$

	1	2	3	4	5	6
1	-1	1	1	3		
2		-1	2	3	3	
3			-1	3	3	
4				-1	4	5
5					-1	5
6						-1

A1	A2	A3	A4	A5	A6
30×35	35×15	15×5	5×10	10×20	20×25

	1	2	3	4	5	6
1	0	15750	7875	9375		
2		0	2625	4375	7125	
3			0	750	2500	5375
4				0	1000	3500
5					0	5000
6						0

$$\mathrm{cost}[3][6] = \begin{cases} \mathrm{cost}[4][6] + d_2 * d_3 * d_6 = 3500 + 15*5*25 = 5375 \\ \mathrm{cost}[3][4] + \mathrm{cost}[5][6] + d_2 * d_4 * d_6 = 750 + 5000 + 15*10*25 = \\ \mathrm{cost}[3][5] + \mathrm{cost}[6][6] + d_2 * d_5 * d_6 = 2500 + 0 + 15*20*25 = 10000 \end{cases}$$

	1	2	3	4	5	6
1	-1	1	1	3		
2		-1	2	3	3	
3			-1	3	3	3
4				-1	4	5
5					-1	5
6						-1

A1	A2	A3	A4	A5	A6
30×35	35×15	15×5	5×10	10×20	20×25

	1	2	3	4	5	6
1	0	15750	7875	9375	11875	
2		0	2625	4375	7125	
3			0	750	2500	5375
4				0	1000	3500
5					0	5000
6						0

	1	2	3	4	5	6
1	-1	1	1	3	3	
2		-1	2	3	3	
3			-1	3	3	3
4				-1	4	5
5					-1	5
6						-1

A1	A2	A3	A4	A5	A6
30×35	35×15	15×5	5×10	10×20	20×25

	1	2	3	4	5	6
1	0	15750	7875	9375	11875	
2		0	2625	4375	7125	10500
3			0	750	2500	5375
4				0	1000	3500
5					0	5000
6						0

	1	2	3	4	5	6
1	-1	1	1	3	3	
2		-1	2	3	3	3
3			-1	3	3	3
4				-1	4	5
5					-1	5
6						-1

A1	A2	A3	A4	A5	A6
30×35	35×15	15×5	5×10	10×20	20×25

	1	2	3	4	5	6
1	0	15750	7875	9375	11875	15125
2		0	2625	4375	7125	10500
3			0	750	2500	5375
4				0	1000	3500
5					0	5000
6						0

最优计算顺序：$(A_1(A_2A_3))((A_4,A_5)A_6)$

	1	2	3	4	5	6
1	-1	1	1	3	3	3
2		-1	2	3	3	3
3			-1	3	3	3
4				-1	4	5
5					-1	5
6						-1