《算法设计与分析》

第五章 动态规划

马丙鹏

2024年11月10日

中国科学院大学
University of Chinese Academy of Sciences

第五章 动态规划

- 5.1 一般方法
- 5.2 多段图问题
- 5.3 每对结点之间的最短路径
- 5.4 最优二分检索树
- 5.5 矩阵连乘问题
- 5.6 0/1背包问题
- 5.7 可靠性设计
- 5.8 货郎担问题
- 5.9 流水线调度问题

5.5 矩阵连乘问题

- **两个矩阵的乘积：**
  - 已知A为p×r的矩阵，B为r×q的矩阵，则A与B的乘积是一个p×q的矩阵，记为C：
    $\mathbf{C = A_{p\times r} \times B_{r\times q} = (c_{ij})_{p\times q}}$，
  - 其中，$c_{ij} = \sum_{1\le k\le r} a_{ik} b_{kj}$， $i = 1,2,..p,\quad j = 1,2,\ldots,\quad q$
  - 每个c_{ij}的计算需要r次乘法（另有r-1次加法，这里仅考虑元素的标量乘法），则计算C共需要pqr次标量乘法运算。

5.5 矩阵连乘问题

■ 问题描述

- 给定n个矩阵$\{A_1, A_2, \ldots, A_n\}$，其中$A_i$与$A_{i+1}$是可乘的，$i = 1, 2, \ldots, n-1$。
- 考察这n个矩阵的连乘积$A_1A_2 \ldots A_n$
- 由于矩阵乘法满足结合律，所以计算矩阵的连乘可以有许多不同的计算次序。这种计算次序可以用加括号的方式来确定。
- 若一个矩阵连乘积的计算次序完全确定，也就是说该连乘积已完全加括号，则可以依此次序反复调用2个矩阵相乘的标准算法计算出矩阵连乘积。

5.5 矩阵连乘问题

■ 问题描述

□设有四个矩阵A_1, A_2, A_3, A_4, 它们的维数分别是,

$A_1 = 10\times100$ $A_2 = 100\times5$ $A_3 = 5\times50$ $A_4 = 50\times30$

□乘积$A_1A_2A_3A_4$总共有五种不同 的加括号方式完成：

$(A_1(A_2(A_3A_4)))$ $(A_1((A_2A_3)A_4))$

$((A_1A_2)(A_3A_4))$ $((A_1(A_2A_3))A_4)$

$(((A_1A_2)A_3)A_4)$

5.5 矩阵连乘问题

■ 问题描述

□ $A_1A_2A_3A_4$不同的加括号方式及其对应计算量分别下:

$A_1 = 10 \times 100$ $A_2 = 100 \times 5$ $A_3 = 5 \times 50$ $A_4 = 50 \times 30$

① $(A_1(A_2(A_3A_4)))$: $5 \times 50 \times 30 + 100 \times 5 \times 30 + 10 \times 100 \times 30 = 52500$

② $(A_1((A_2A_3)A_4))$: $100 \times 5 \times 50 + 100 \times 50 \times 30 + 10 \times 100 \times 30 = 205000$

③ $((A_1A_2)(A_3A_4))$: $10 \times 100 \times 5 + 5 \times 50 \times 30 + 10 \times 5 \times 30 = 14000$

④ $((A_1(A_2A_3))A_4)$: $100 \times 5 \times 50 + 10 \times 100 \times 50 + 10 \times 50 \times 30 = 90000$

⑤ $(((A_1A_2)A_3)A_4)$: $10 \times 100 \times 5 + 10 \times 5 \times 50 + 10 \times 50 \times 30 = 22500$

□ 最优解 == 以最少的数乘次数计算出矩阵连乘的乘积

5.5 矩阵连乘问题

■ 问题描述

- 矩阵连乘问题定义
  - 给定n个矩阵$\{A_1, A_2, \ldots, A_n\}$，其中$A_i$与$A_{i+1}$是可乘的，$i=1, 2\ldots, n-1$。
  - 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。

5.5 矩阵连乘问题

■ 穷举法求解

- □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。
- □算法复杂度分析：
  - ➢对于**n**个矩阵的连乘积，设其不同的计算次序为**P(n)**。
  - ➢由于每种加括号方式都可以分解为两个子矩阵的加括号问题：$(A_1 \ldots A_k)(A_{k+1} \ldots A_n)$可以得到关于**P(n)**的递推式如下：

$$P(n) = \begin{cases} 1, & n = 1 \\ \sum_{k=1}^{n-1} P(k)P(n-k), & n > 1 \end{cases} \quad \Rightarrow \quad P(n) = \Omega(4^n / n^{3/2})$$

5.5 矩阵连乘问题

- 贪心策略1: 计算量小的优先
  - 计算n个矩阵的连乘积共有n-1次乘法计算。
  - 首先在n-1次乘法计算中选择乘法计算量最小的两个矩阵优先计算;
  - 然后再在剩下的n-2次乘法计算中选择计算量最小的两个矩阵进行计算，依此往下。
  - 该解决策略在某些情况下可以得到最优解，但是在很多情况下得不到最优解。
  - 如上例的第(5)种完全加括号方式就是采用该策略，但它得到的显然不是最优解。

5.5 矩阵连乘问题

- 贪心策略2:矩阵维数大的优先计算
  - 在n个矩阵的n+1个维数序列 $p_0, p_1, p_2,\ldots, p_n$中选择维数的最大值(设为$p_i$)，并把相邻两个含有维数$p_i$的矩阵优先进行计算:
  - 然后在剩下的n个维数序列中再次选择维数的最大值，进行相应矩阵的乘法运算;依此往下。
  - 该解决策略与上一种策略相似，在有些情况下可以得到最优解，但是在有些情况下得不到最优解。
  - 如上例的第(3)种完全加括号方式就是采用该策略显然它得到的是最优解。但当4个矩阵A_1, A_2, A_3, A_4的维数改为50×10，10×40，40×30和30×5时，可以验证，采用该策略得到的就不是最优解。

5.5 矩阵连乘问题

- 分治法
  - 设n个矩阵连乘积 A_1, A_2, A_3, A_4的计算次序在矩阵A_k和A_{k+1}之间将矩阵链断开，$1\le k<n$，则其相应的完全加括号方式为$((A_1 \ldots A_k)(A_{k+1} \ldots A_n))$由于完全加括号方式是一个递归定义，因此可以递归地计算A[1:k]和A[k+1:n]，然后将计算结果相乘得到A[1:n]。
  - 据此可设计如下递归算法recurmatrixChain:

5.5 矩阵连乘问题

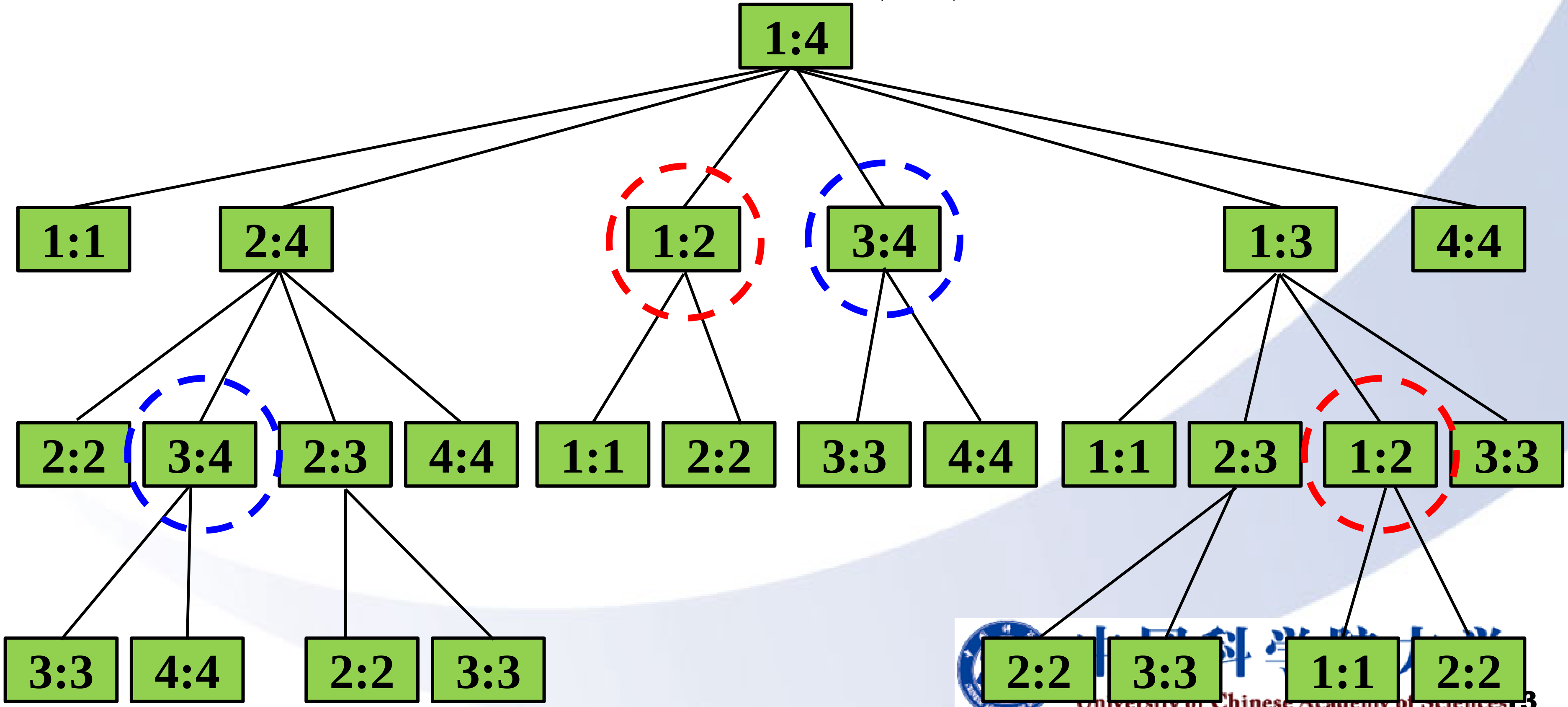
■ 分治法

```
int recurMatrixChain(int i, int j)
{
    if(i == j)  return 0;
    int u = recurMatrixChain(i, i)+recurMatrixChain(i+1,j)
             +p[i-1]*p[i]*p[j];
    s[i][j]=i;
    for (int k = i+1; k <j; k++){
       int t = recurMatrixChain(i, k) + recurMatrixChain(k+1, j)
               + p[i-1]*p[k]*p[j];
       if( t < u){ u = t;    s[i][j] = k; }
     }
    return u;
}
```

5.5 矩阵连乘问题

从图可以看出，许多子问题被重复计算，这是分治法效率低下的根本原因。

- ■ 分治法
  - □为何分治法的效率如此低下?
    - ➢算法recurmatrixChain(1,4)计算A[1:4]的递归树



5.5 矩阵连乘问题

- 分治法
  - 为何分治法的效率如此低下?
    - 该该递归算法的计算时间T(n)可递归定义如下:

$$T(n) \geq \begin{cases} O(1) & n = 1 \\ 1 + \sum_{k=1}^{n-1} (T(k) + T(n-k) + 1) & n > 1 \end{cases}$$

    - 当n>1时

$$T(n) \geq 1 + \sum_{k=1}^{n-1} (T(k) + T(n-k) + 1) = n + 2\sum_{k=1}^{n-1} T(k)$$

    - 以上递归式可用数学归纳法证明 $T(n) \geq 2^{n-1} = \Omega(2^n)$
    - 该算法的计算时间T(n)有指数下界。
    - 对该问题而言，分治法是一个可行方法，但不是一个有效算法。

5.5 矩阵连乘问题

- 动态规划法
  - 矩阵连乘问题满足最优性原理
    - 记$A_{i,j}$表示对乘积$A_iA_{i+1}\ldots A_j$求值的计算模式，其中$i\le j$。
    - 且如果$i<j$，则对于任意k，$i\le k<j$，乘积$A_iA_{i+1}\ldots A_j$都可以通过加括号的方式在A_k与A_{k+1}之间分开，即可以首先计算$A_{i,k}$和$A_{k+1,j}$，然后把它们相乘得到最终乘积$A_{i,j}$。
    - 计算$A_{i,j}$的代价就是计算$A_{i,k}$和$A_{k+1,j}$的代价之和，再加上两者相乘的代价。

中国科学院大学 University of Chinese Academy of Sciences

5.5 矩阵连乘问题

- 动态规划法
  - 最优子结构的证明
    - 假设$A_{i,j}$是$A_iA_{i+1}\ldots A_j$通过加括号后得到的一个最优计算模式、且该计算模式下恰好在A_k与A_{k+1}之间分开。
    - 则其中的$A_{i,k}$对“前缀”子链$A_iA_{i+1}\ldots A_k$必是最优加括号计算子模式。
    - 如若不然，设$A'_{i,k}$是<$A_i,A_{i+1},\ldots A_k$>一个代价更小的计算模式，则由$A'_{i,k}$和$A_{k+1,j}$构造计算过程$A'_{i,j}$，代价将比$A_{i,j}$小，这与$A_{i,j}$是最优连乘模式相矛盾。
    - 对$A_{k+1,j}$亦然。
    - 所以，矩阵连乘问题满足最优性原理，具有最优子结构性。

5.5 矩阵连乘问题

■ 矩阵连乘问题

□递推关系式

- ➢设计算A[i:j]，1≤i≤j≤n，所需要的最少数乘次数m[i,j]，则原问题的最优值为m[1,n]
- ➢当i=j时，A[i:j]=A_i，因此，m[i,i]=0，i=1,2,…,n
- ➢当i<j时，
  - ✓对任意的k，i≤k<j分开的矩阵链乘<A_i,A_{i+1},…A_j>，子乘积$A_{i,k}$是一个p_{i-1}×p_k的矩阵，$A_{k+1,j}$是一个p_k×p_j的矩阵。则：
  - ✓$m[i, j] = m[i,k] + m[k+1, j] + p_{i-1}p_k\, p_j$
  - ✓而这样的k有j-i可能性，取其中和值最小者。

5.5 矩阵连乘问题

- 矩阵连乘问题
  - 递推关系式
    - 由最优子结构性质，可以递归地定义**m[i,j]**为：

$$m(i,j)=\begin{cases}0, & i=j\\ \min\limits_{i\le k<j}\{m(i,k)+m(k+1,j)+p_{i-1}p_kp_j\}, & i<j\end{cases}$$

      - ✓**k**的位置只有**j-i**种可能
    - 再设**s[i,j]**，记录使**m[i,j]**取最小值的**k**，则可以依靠**s**求出最优链乘模式。

5.5 矩阵连乘问题

- 动态规划法
  - 递推关系式
    - 下述过程MATRIX-CHAIN-ORDER计算n个矩阵链乘的最优模式。
    - 其中，输入序列p=<p_0, p_1, ..., p_n>是n个矩阵的维数表示，矩阵A_i的维数是$p_{i-1}\times p_i$, i=1,2,...,n。
    - 使用辅助表m[1..n, 1..n]保存m[i,j]的代价，使用s[1..n, 1..n]记录计算m[i, j]时取得最优代价的k值。
    - 该过程自底向上完成m[i, j]的计算：在m[i,i]=0的基础上，求出所有m[i,i+k]。最后算出m[1,n]。

```
MATRIX-CHAIN-ORDER(p)
   n←length[p]-1                    //length[p]=n+1
   for i←1 to n do m[i,i]=0 repeat   //m初始化
   for h←1 to n-1 do            //h是当前需要计算的链的长度
      for i←1 to n-h do
         j←i+h           //[i,j]是当前需要计算的链区间
         m[i,j]←∞
         for k←i to j-1 do      //i≤k<j
            q←m[i,k] + m[k+1,j] + P_{i-1}P_kP_j
            if q<m[i,j] then
               m[i,j]←q;  s[i,j]←k;
            endif
         repeat
      repeat
   repeat
   return m and s
END MATRIX-CHAIN-ORDER
```

5.5 矩阵连乘问题

- 动态规划法
  - 计算最优值

A_1	A_2	A_3	A_4	A_5	A_6
30×35	35×15	15×5	5×10	10×20	20×25

① 计算A_1, A_2, A_3, A_4, A_5, A_6

② 计算(A_1A_2), (A_2A_3), (A_3A_4), (A_4A_5), (A_5A_6)

③ 计算$(A_1A_2A_3)$, …, $(A_4A_5A_6)$

④ 计算A[1: 4], A[2: 5], A[3: 6]

⑤ 计算A[1: 5], A[2: 6]

⑥ 计算A[1: 6]

5.5 矩阵连乘问题

$$m(i,j)=\begin{cases}0, & i=j\\ \min\limits_{i\le k<j}\{m(i,k)+m(k+1,j)+p_{i-1}p_kp_j\}, & i<j\end{cases}$$

- ■ 动态规划法
  - □计算最优值

p_0	p_1	p_2	p_3	p_4	p_5	p_6
30	35	15	5	10	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

m[i,i]=0, i=1,2,…,n

$m[1,2]=m[1,1]+m[2,2]+p_0p_1p_2$
$=0+0+30\times35\times15=15750$

$m[2,3]=m[2,2]+m[3,3]+p_1p_2p_3$
$=0+0+35\times15\times5=2625$

$m[3,4]=m[3,3]+m[4,5]+p_2p_3p_4$
$=0+0+15\times5\times10=750$

$m[4,5]=m[4,4]+m[5,5]+p_3p_4p_5$
$=0+0+5\times10\times20=1000$

$m[5,6]=m[5,5]+m[6,6]+p_4p_5p_6=0+0+10\times20\times25=5000$

5.5 矩阵连乘问题

$$m(i,j)=\begin{cases}0, & i=j\\ \min\limits_{i\le k<j}\{m(i,k)+m(k+1,j)+p_{i-1}p_kp_j\}, & i<j\end{cases}$$

- ■ 动态规划法
  - □计算最优值

p_0	p_1	p_2	p_3	p_4	p_5	p_6
30	35	15	5	10	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

$$m[1,3]=\min\begin{cases}m[1,1]+m[2,3]+p_0p_1p_3=0+2625+30\times35\times5=7875\\ m[1,2]+m[3,3]+p_0p_2p_3=15750+0+30\times15\times5=18000\end{cases}$$

$$m[2,4]=\min\begin{cases}m[2,2]+m[3,4]+p_1p_2p_4=0+750+35\times15\times10=6000\\ m[2,3]+m[3,4]+p_1p_3p_4=2625+0+35\times5\times10=4375\end{cases}$$

5.5 矩阵连乘问题

$$m(i,j)=\begin{cases}0, & i=j\\ \min\limits_{i\le k<j}\{m(i,k)+m(k+1,j)+p_{i-1}p_kp_j\}, & i<j\end{cases}$$

■ 动态规划法

□计算最优值

p_0	p_1	p_2	p_3	p_4	p_5	p_6
30	35	15	5	10	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

$$m[1,4]=\min\begin{cases}m[1,1]+m[2,4]+p_0p_1p_4=0+4375+30\times35\times10=14875\\ m[1,2]+m[3,4]+p_0p_2p_4=15750+750+30\times15\times10=21000\\ m[1,3]+m[4,4]+p_0p_3p_4=7875+0+30\times5\times10=9375\end{cases}$$

$$m[2,5]=\min\begin{cases}m[2,2]+m[3,5]+p_1p_2p_5=0+2500+35\times15\times20=10500\\ m[2,3]+m[4,5]+p_1p_3p_5=2625+1000+35\times5\times20=7125\\ m[2,4]+m[5,5]+p_1p_4p_5=4375+0+35\times10\times20=11375\end{cases}$$

5.5 矩阵连乘问题

$$m(i,j)=\begin{cases}0, & i=j\\ \min\limits_{i\le k<j}\{m(i,k)+m(k+1,j)+p_{i-1}p_kp_j\}, & i<j\end{cases}$$

■ 动态规划法

□计算最优值

p_0	p_1	p_2	p_3	p_4	p_5	p_6
30	35	15	5	10	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

$$m[1,5]=\min\begin{cases}m[1,1]+m[2,5]+p_0p_1p_5=0+7125+30\times35\times20=28125\\ m[1,2]+m[3,5]+p_0p_2p_5=15750+2500+30\times15\times20=27250\\ m[1,3]+m[4,5]+p_0p_3p_5=7875+1000+30\times5\times20=11875\\ m[1,4]+m[5,5]+p_0p_4p_5=9375+0+30\times10\times20=15375\end{cases}$$

5.5 矩阵连乘问题

$$m(i,j)=\begin{cases}0, & i=j\\ \min\limits_{i\le k<j}\{m(i,k)+m(k+1,j)+p_{i-1}p_kp_j\}, & i<j\end{cases}$$

- 动态规划法
  - 计算最优值

p_0	p_1	p_2	p_3	p_4	p_5	p_6
30	35	15	5	10	20	25

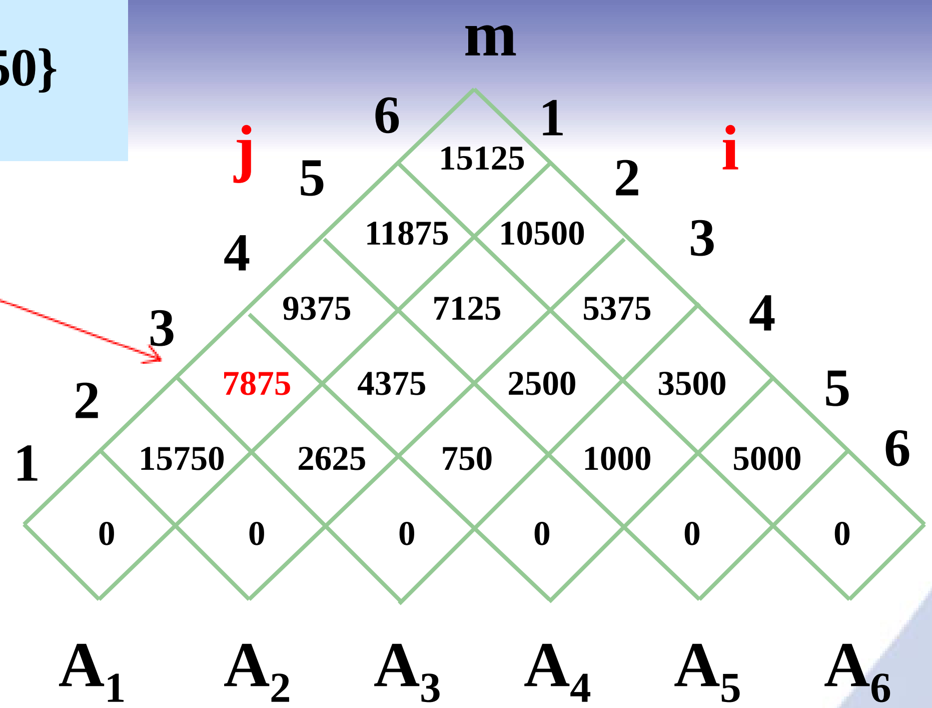
	1	2	3	4	5	6
1	0	15750	7875	9375	11875	15125
2		0	2625	4375	7125	10500
3			0	750	2500	5375
4				0	1000	3500
5					0	5000
6						0

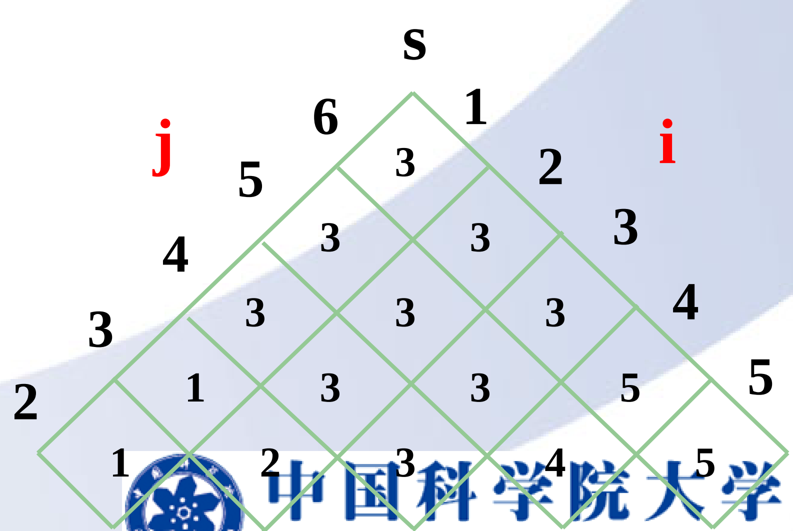
$$m[1,6]=\min\begin{cases}m[1,1]+m[2,6]+p_0p_1p_6=0+10500+30\times35\times25=36750\\ m[1,2]+m[3,6]+p_0p_2p_6=15750+5375+30\times15\times25=32375\\ m[1,3]+m[4,6]+p_0p_3p_6=7875+3500+30\times5\times25=15125\\ m[1,4]+m[5,6]+p_0p_4p_6=9375+5000+30\times10\times25=21875\\ m[1,5]+m[6,6]+p_0p_5p_6=11875+0+30\times20\times25=26875\end{cases}$$

$$\begin{aligned} m[1,3]=\min\{&m[1,1]+m[2,3]+30\times35\times5,\\ &m[1,2]+m[3,3]+30\times15\times5\}\\ =\min\{&0+2625+5250,\ 15750+0+2250\}\\ =7875& \end{aligned}$$

■ 例，

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





$$m(i,j)=\begin{cases}0, & i=j\\ \min\limits_{i\le k<j}\{m(i,k)+m(k+1,j)+p_{i-1}p_kp_j\}, & i<j\end{cases}$$

5.5 矩阵连乘问题

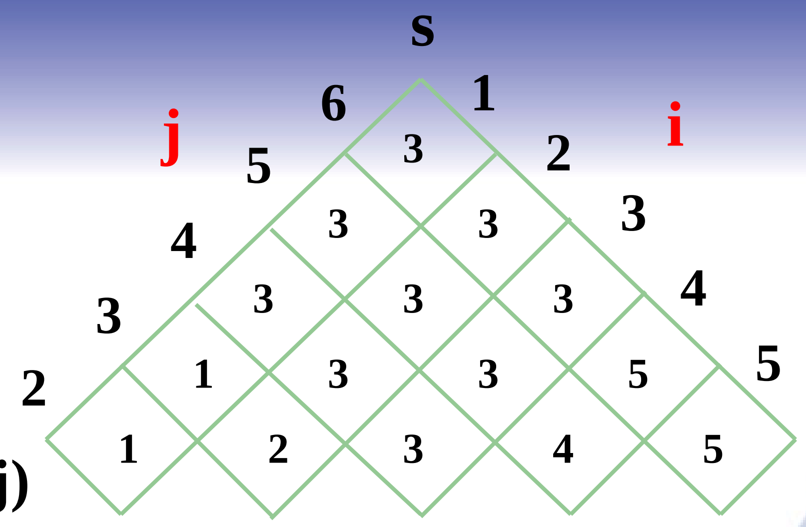
- 动态规划法
  - 构造最优解
    - 如何根据S求出矩阵链乘的最优计算模式?
    - S[i,j]记录了$A_iA_{i+1}\ldots A_j$的最优括号化方案的“最后一个”分割点k。
    - $A_{1\ldots n}$的最优方案中最后一次矩阵乘运算是$A_{1\ldots s[1,n]}A_{s[1,n]+1\ldots n}$。
    - 用递归的方法求出 $A_{1\ldots s[1,n]}$、$A_{s[1,n]+1\ldots n}$等其他所有子问题的最优括号化方案。

中国科学院大学 University of Chinese Academy of Sciences

5.5 矩阵连乘问题

■ 动态规划法

□构造最优解




```
PRINT-OPTIMAL-PARENS(s,i,j)
  if  i == j  print "A"
  else
     print "("
     PRINT-OPTIMAL-PARENS(s, i, s[i,j])
     PRINT-OPTIMAL-PARENS(s, s[i,j+1], j)
    print ")"
```

➢例： PRINT-OPTIMAL-PARENS(s,1,6) ⟹

((A1 (A2A3)) ((A4A5) A6))

5.5 矩阵连乘问题

■ 动态规划法

□时间复杂度分析

➢算法的主体由一个三层循环构成。外循环执行n-1次，内层循环至多执行n-1次，所以MATRIX-CHAIN-ORDER的算法复杂度是$O(n^3)$。

```
for h←1 to n-1  do          //h是当前需要计算的链的长度
    for i←1 to n-h do
    …
        for k←i to j-1 do       //i≤k<j
        …
        repeat
    repeat
repeat
```

5.5 矩阵连乘问题

- 动态规划法
  - 时间复杂度分析
    - 对于$1 \le i \le j \le n$不同的有序对(i,j)对应于不同的子问题。因此，不同子问题的个数最多只有

$$\binom{n}{2} + n = \Theta(n^2)$$

    - 由此可见，在递归计算时，许多子问题被重复计算多次。这也是该问题可用动态规划算法求解的又一显著特征。

5.5 矩阵连乘问题

- 动态规划法
  - 时间复杂度分析
    - 用动态规划算法解此问题，可依据其递归式以自底向上的方式进行计算。
    - 在计算过程中，保存已解决的子问题答案。
    - 每个子问题只计算一次，而在后面需要时只要简单查一下，从而避免大量的重复计算，最终得到多项式时间的算法。

5.5 矩阵连乘问题

- 动态规划法
  - 时间复杂度分析
    - 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。
    - 在分析问题的最优子结构性质时，所用的方法具有普遍性：首先假设由问题的最优解导出的子问题的解不是最优的，然后再设法说明在这个假设下可构造出比原问题最优解更好的解，从而导致矛盾。
    - 利用问题的最优子结构性质，以自底向上的方式递归地从子问题的最优解逐步构造出整个问题的最优解。最优子结构是问题能用动态规划算法求解的前提。

中国科学院大学
University of Chinese Academy of Sciences

5.5 矩阵连乘问题

- 动态规划法
  - 时间复杂度分析
    - 递归算法求解问题时，每次产生的子问题并不总是新问题，有些子问题被反复计算多次。这种性质称为子问题的重叠性质。
    - 动态规划算法，对每一个子问题只解一次，而后将其解保存在一个表格中，当再次需要解此子问题时，只是简单地用常数时间查看一下结果。
    - 通常不同的子问题个数随问题的大小呈多项式增长。
    - 因此用动态规划算法只需要多项式时间，从而获得较高的解题效率。

End