动态规划

Jun Wu

wujun@yzu.edu.cn

March 28, 2018

公共子序列

- 给定字符序列$x[1\cdots m]$和$y[1\cdots n]$，若字符序列$z[1\cdots k]$既是x的子序列也是y的子序列，则称z是x和y的公共子序列。

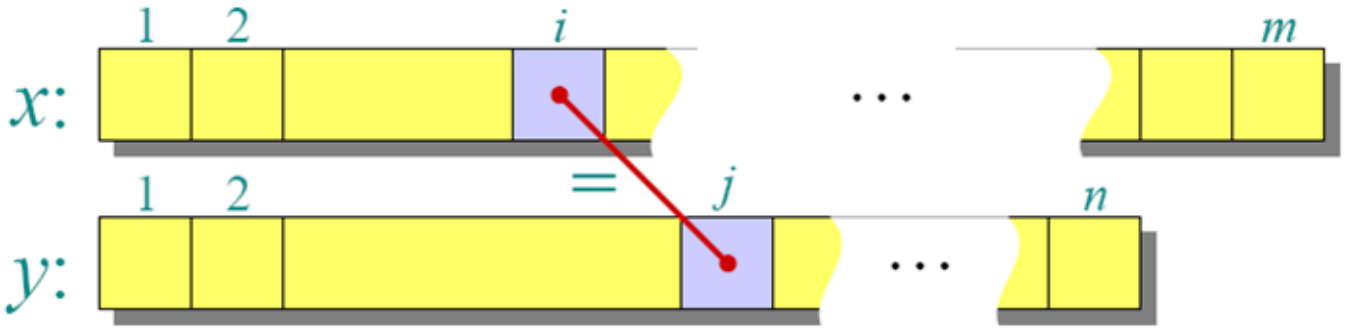
Problem (最长公共子序列)

实例： 输入字符序列$x[1\cdots m]$和$y[1\cdots n]$。

问题： 寻找公共子序列$z[1\cdots k]$使得k最大。

计算最长公共子序列长度的递归关系

- 定义$c[i,j]$为$x[1\cdots i]$和$y[1\cdots j]$的最长公共子序列的长度。



$$c[i,j]=\begin{cases}0, & if\ \ i=0\ \ or\ \ j=0\\ c[i-1,j-1]+1, & if\ \ i,j>0 \text{ and } x[i]=y[j]\\ \max\{c[i-1,j],c[i,j-1]\}, & if\ \ i,j>0 \text{ and } x[i]\neq y[j]\end{cases}$$

递归关系的正确性

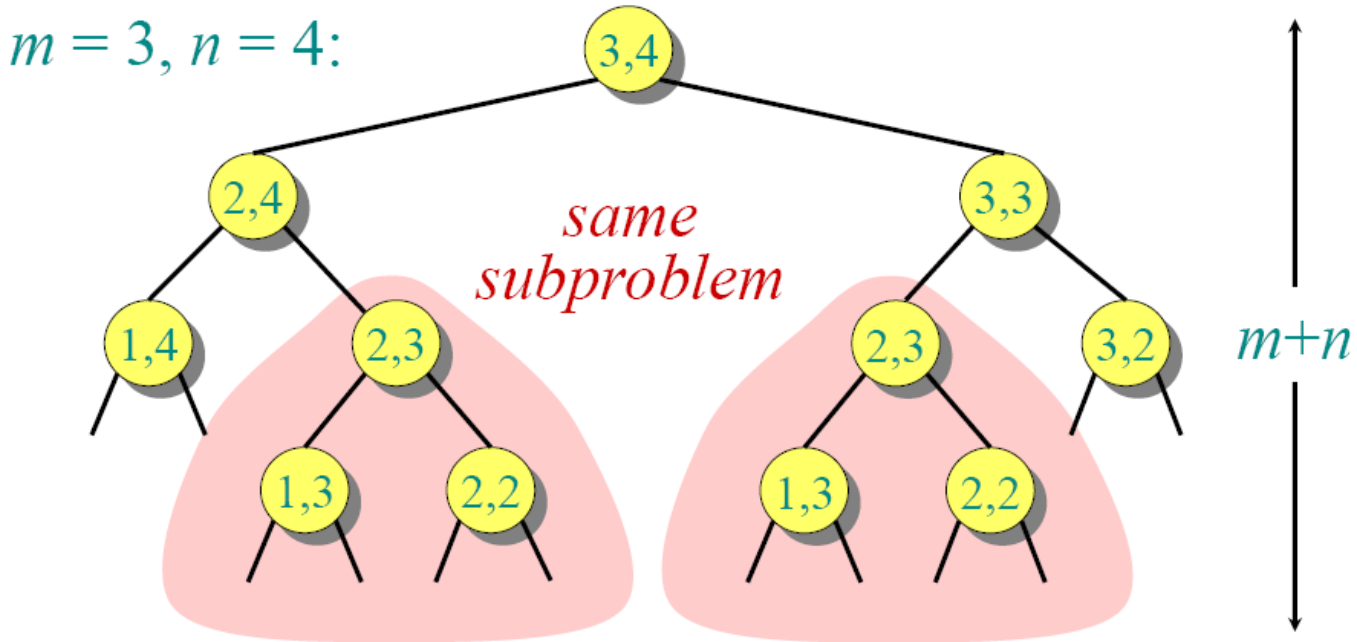
Lemma 1

设$z[1\cdots k]$为$x[1\cdots m]$和$y[1\cdots n]$的最长公共子序列。则，

1. 若$x[m]=y[n]$，那么$x[m]=y[n]=z[k]$且$z[1\cdots k-1]$是$x[1\cdots m-1]$和$y[1\cdots n-1]$的最长公共子序列；
2. 若$x[m]\neq y[n]$，且$z[k]\neq x[m]$，则$z[1\cdots k]$是$x[1\cdots m-1]$和$y[1\cdots n]$的最长公共子序列；
3. 若$x[m]\neq y[n]$，且$z[k]\neq y[n]$，则$z[1\cdots k]$是$x[1\cdots m]$和$y[1\cdots n-1]$的最长公共子序列。

- 证明思路：(剪切-黏贴法，以(1)为例。)
- 假设$z[1\cdots k-1]$不是$x[1\cdots m-1]$和$y[1\cdots n-1]$的最长公共子序列。
- 那么，有$x[1\cdots m-1]$和$y[1\cdots n-1]$的最长公共子序列$z'[1\cdots k'-1]$，$k'>k$。
- 而$z'\cdot x[m]$是x和y的最长公共子序列，这与z是最长公共子序列矛盾。

递归算法的复杂性



- 递归树深度$O(m+n)$。
- 但不同子问题的数量$O(mn)$。

计算$c[i,j]$

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0						
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

计算$c[i,j]$

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0 ↑	0 ↑	0 ↑	1=	1←	1=
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

计算$c[i,j]$

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0 ↑	0 ↑	0 ↑	1=	1←	1=
2	B	0	1=	1 ←	1 ←	1 ↑	2=	2 ←
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

计算$c[i,j]$

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0 ↑	0 ↑	0 ↑	1=	1←	1=
2	B	0	1=	1 ←	1 ←	1 ↑	2=	2 ←
3	C	0	1 ↑	1 ↑	2=	2 ←	2 ↑	2 ↑
4	B	0	1=	1 ↑	2 ↑	2 ↑	3=	3 ←
5	D	0	1 ↑	2=	2 ↑	2 ↑	3 ↑	3 ↑
6	A	0	1 ↑	2 ↑	2 ↑	3=	3 ↑	4=
7	B	0	1=	2 ↑	2 ↑	3 ↑	4=	4 ↑

LCS-LENGTH(X, Y)

```
1:  m ← length[X]
2:  n ← length[Y]
3:  for i ← 1 to m do
4:      c[i, 0] ← 0
5:  end for
6:  for j ← 0 to n do
7:      c[0, j] ← 0
8:  end for
9:  for i ← 1 to m do
10:     for j ← 1 to n do
11:         if x_i = y_j then
12:             c[i, j] ← c[i − 1, j − 1] + 1
13:             b[i, j] ← " = "
14:         else
15:             if c[i − 1, j] ≥ c[i, j − 1]
                then
16:                 c[i, j] ← c[i − 1, j]
17:                 b[i, j] ← " ↑ "
18:             else
19:                 c[i, j] ← c[i, j − 1]
20:                 b[i, j] ← " ← "
21:             end if
22:         end if
23:     end for
24: end for
```

根据$b[]$计算LCS

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0 ↑	0 ↑	0 ↑	1=	1←	1=
2	B	0	1=	1 ←	1 ←	1 ↑	2=	2 ←
3	C	0	1 ↑	1 ↑	2=	2 ←	2 ↑	2 ↑
4	B	0	1=	1 ↑	2 ↑	2 ↑	3=	3 ←
5	D	0	1 ↑	2=	2 ↑	2 ↑	3 ↑	3 ↑
6	A	0	1 ↑	2 ↑	2 ↑	3=	3 ↑	4=
7	B	0	1=	2 ↑	2 ↑	3 ↑	4=	4 ↑

PRINT-LCS(b, X, i, j)

```
1: if i = 0 ∪ j = 0 then
2:     RETURN
3: end if
4: if b[i, j] = "=" then
5:     PRINT-LCS(b, X, i − 1, j − 1)
6:     print x[i]
7: else
8:     if b[i, j] = "↑" then
9:         PRINT-LCS(b, X, i − 1, j)
10:    else
11:        PRINT-LCS(b, X, i, j − 1)
12:    end if
13: end if
```

- 复杂度?

问题定义

- 例：

$$A_1 \times A_2 \times A_3$$

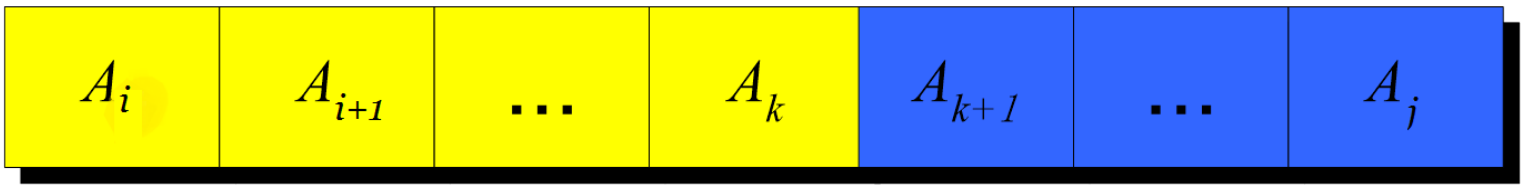
- 其中，A_1, A_2, A_3 分别为$10 \times 100, 100 \times 5, 5 \times 50$的矩阵
- 按$(A_1 \times A_2) \times A_3$计算，需要7500次基本乘法运算
- 按$A_1 \times (A_2 \times A_3)$计算，需要75000次乘法运算

Problem (矩阵连乘)

实例： 给定n个矩阵$\{A_1, A_2, \cdots, A_n\}$，其中$A_i$与$_{i+1}$是可乘的，$i = 1, 2, ,, n-1$。即输入为$n$个矩阵的下标$p_0, p_1, \cdots, p_n$。

问题： 寻找计算矩阵乘积的次序，使得依此次序计算矩阵连乘积需要的乘法次数最少。

递归关系



- 假设最后一步乘法的位置在k。
- 那么将有两个子问题$A_i \times \cdots \times A_k$和$A_{k+1} \times \cdots \times A_j$。
- 这样，

$$m[i,j] = m[i,k] + m[k+1,j] + p_{i-1}p_k p_j$$

- 但是，我们并不知道最优顺序的最后一步乘的位置！？

递归算法

- 需要搜索所有可能的k，从中找到最优的。

$$m[i,j] = \begin{cases} \Theta(1), & i = j \\ \min_{i \le k < j}\{m[i,k] + m[k+1,j] + p_{i-1}p_k p_j\}, & i < j \end{cases}$$

- 递归关系的正确性？（剪切-黏贴法）
- 递归算法的复杂性：

$$P(n) = \begin{cases} 1, & n = 1 \\ \sum_{k=1}^{n-1} P(k)P(n-k), & n > 1 \end{cases}$$

- $P(n)$称作Catalan数，是$\Omega(4^n/n^{\frac{3}{2}})$。

动态规划法–矩阵连乘实例

- 矩阵连乘实例：$A_1 \times A_2 \times \cdots \times A_6$

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

- 即矩阵的下标数组为：

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

- 如何设计表格？

动态规划法–矩阵连乘实例

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		15750				
2			2625			
3				750		
4					1000	
5						5000
6						

动态规划法–矩阵连乘实例

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		15750	7875			
2			2625			
3				750		
4					1000	
5						5000
6						

动态规划法–矩阵连乘实例

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

动态规划法–矩阵连乘实例

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

动态规划法–矩阵连乘实例

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

MATRIX-CHAIN-ORDER(p)

```
1   n ← length[p] - 1
2   for i ← 1 to n
3       do m[i, i] ← 0
4   for l ← 2 to n ▷ l is the chain length.
5       do for i ← 1 to n - l + 1
6                  do j ← i + l - 1
7              m[i, j] ← ∞
8              for k ← i to j - 1
9                do q ← m[i, k] + m[k + 1, j] + p_{i-1} p_k p_j
10                        if q < m[i, j]
11                           then m[i, j] ← q
12                                s[i, j] ← k
13  return m and s
```

通过s找到最优顺序

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

PRINT-OPTIMAL-PARENS(s, i, j)

```
1 if i = j
2       then print "A_i"
3            else print "("
4                 PRINT-OPTIMAL-PARENS(s, i, s[i, j])
5                 PRINT-OPTIMAL-PARENS(s, s[i, j] + 1, j)
6 print ")"
```

- 复杂度分析：
- 计算m的每个表项$O(n)$
- 共有$O(n^2)$个表项，因此，计算m和s的开销是$O(n^3)$
- PRINT-OPTIMAL-PARENS的复杂度？

动态规划的基本思路

- 本质上仍然是“大事化小”策略
- 所用的技巧：保存已求出小问题的解备用
- 什么问题可以采用动态规划求解？

最优子结构性质

大问题的最优解包含了小问题的最优解。

- 最优子结构性质又称做“最优化原理”或“马尔科夫性”
- 如何分析问题是否具备最优子结构性质？
- 通常可以采用“剪切-黏贴”法证明。

备忘录法

动态规划的基本步骤：

1. 找出最优解的性质，并刻划其结构特征。
2. 递归地定义最优值。
3. 以规划的方式计算出最优值。
4. 根据计算最优值时得到的信息，构造最优解。

- 在计算最优解值时，可以采用自底向上的填表法
- 也可以采用自顶向下的备忘录法
- 备忘录法保留了递归结构，算法流程清晰但开销稍大
- 当所有的子问题都需要求解时，自底向上的方法效率较高，否则可以采用备忘录方法

备忘录法示例

LOOKUP-CHAIN(p, i, j)

```
1 if m[i, j] < ∞
2   then return m[i, j]
3 if i = j
4   then m[i, j] ← 0
5    else for k ← i to j - 1
6         do q ← LOOKUP-CHAIN(p, i, k) + LOOKUP-
                                CHAIN(p,k + 1, j) + p_{i-1} p_k p_j
7            if q < m[i, j]
8                 then m[i, j] ← q
9 return m[i, j]
```

备忘录法示例

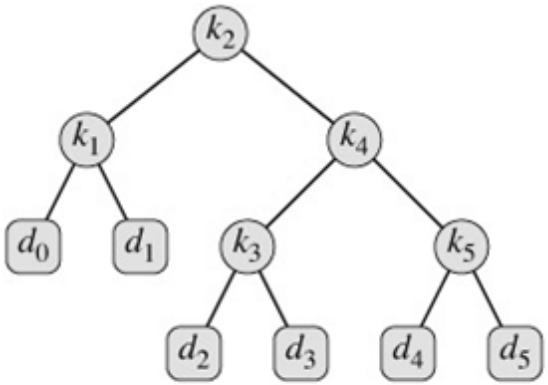
MEMOIZED-MATRIX-CHAIN(p)

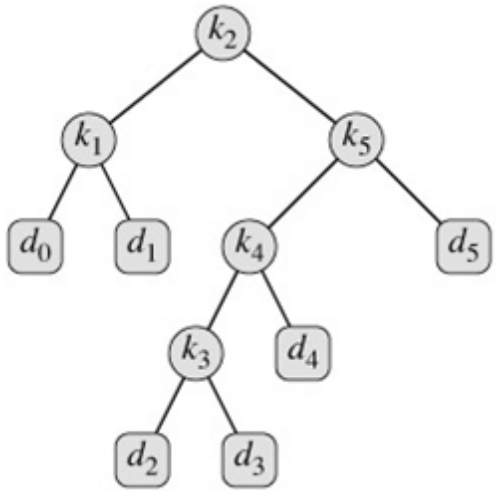
```
1 n ← length[p] - 1
2 for i ← 1 to n
3         do for j ← i to n
4                 do m[i, j] ← ∞
5 return LOOKUP-CHAIN(p, 1, n)
```

- 复杂性分析：
- 只要$m[i,j] < \infty$，lookup chain 将在第二行返回
- 因此总共递归调用的次数$O(n^2)$
- 算法的复杂度仍然是$O(n^3)$

二叉查找树

- 给定n个键值$k_1 \le k_2 \le \cdots \le k_n$，以及相应的查找频率
- $p_1, \cdots, p_n$和$q_0, \cdots, q_n$: $\sum_{i=1}^{n} p_i + \sum_{j=0}^{n} q_j = 1$.





对于查找树T，定义平均查找长度：

$$\mathbb{E}[\text{search cost in T}] = \sum_{i=1}^{n}(d_T(k_i)+1)p_i + \sum_{j=0}^{n}(d_T(d_j)+1)q_j$$

$$= 1 + \sum_{i=1}^{n} p_i d_T(k_i) + \sum_{j=0}^{n} d_T(d_j)$$

最优二叉查找树

node	depth	probability	contribution
k_1	1	0.15	0.30
k_2	0	0.10	0.10
k_3	2	0.05	0.15
k_4	1	0.10	0.20
k_5	2	0.20	0.60
d_0	2	0.05	0.15
d_1	2	0.10	0.30
d_2	3	0.05	0.20
d_3	3	0.05	0.20
d_4	3	0.05	0.20
d_5	3	0.10	0.40
Total			2.80

Problem (最优二叉查找树)

实例： n个键值$k_1 \leq k_2 \leq \cdots \leq k_n$，以及相应的查找频率$p_1, \cdots, p_n$和$q_0, \cdots, q_n$。

问题： 寻找二叉查找树，使得平均查找长度最短。

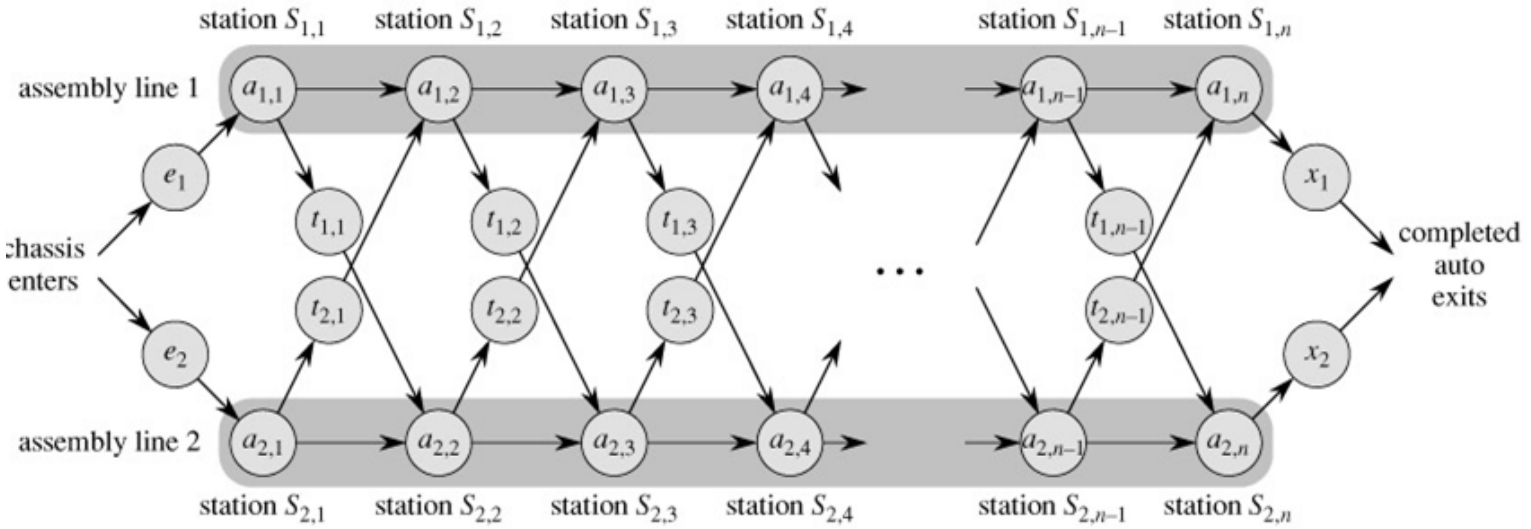
递归关系

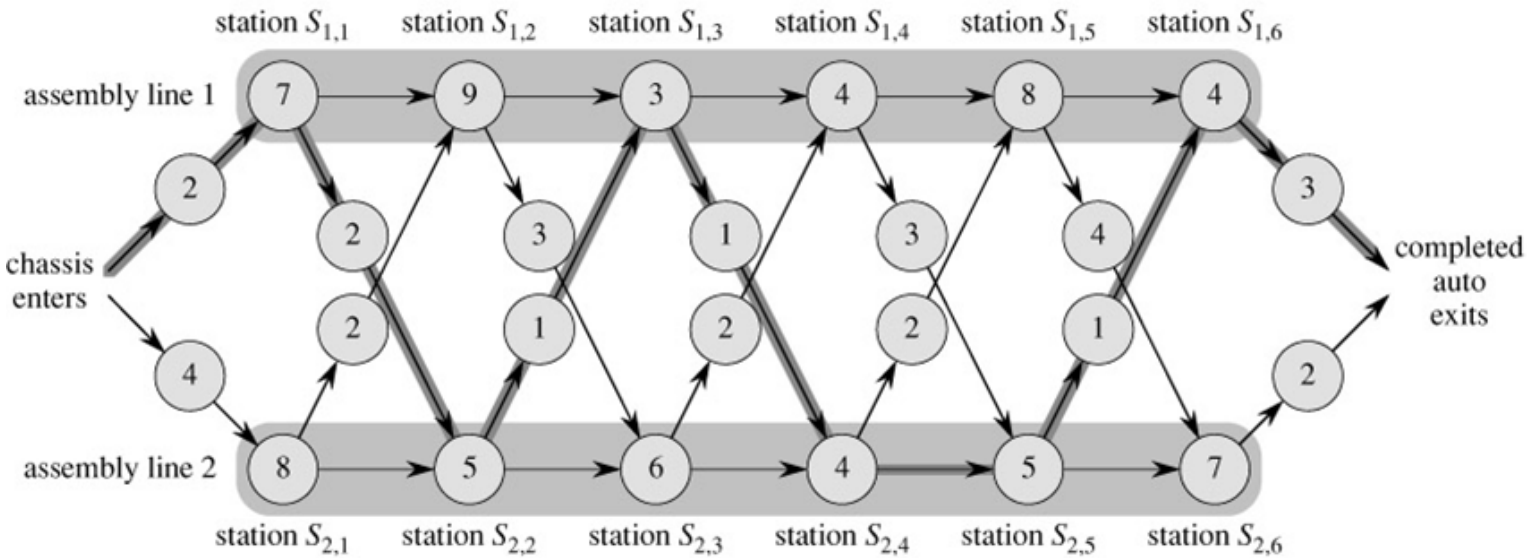
$$e[i,j] = \begin{cases} q_i, \text{if } j = i-1 \\ \\ \min\limits_{i \le r \le j} \{e[i,r-1] + e[r+1,j] + w[i,j]\}, \text{if } j \ge i \end{cases}$$

where,

$$w[i,j] = \sum_{l=i}^{j} p_l + \sum_{\ell=i-1}^{j} q_\ell$$

流水线调度


station $S_{1,1}$
station $S_{1,2}$
station $S_{1,3}$
station $S_{1,4}$
station $S_{1,n-1}$
station $S_{1,n}$
assembly line 1
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,4}$ $a_{1,n-1}$ $a_{1,n}$
e_1
$t_{1,1}$ $t_{1,2}$ $t_{1,3}$ $t_{1,n-1}$
x_1
chassis enters
completed auto exits
$t_{2,1}$ $t_{2,2}$ $t_{2,3}$ $t_{2,n-1}$
e_2
x_2
assembly line 2
$a_{2,1}$ $a_{2,2}$ $a_{2,3}$ $a_{2,4}$ $a_{2,n-1}$ $a_{2,n}$
station $S_{2,1}$
station $S_{2,2}$
station $S_{2,3}$
station $S_{2,4}$
station $S_{2,n-1}$
station $S_{2,n}$


station $S_{1,1}$
station $S_{1,2}$
station $S_{1,3}$
station $S_{1,4}$
station $S_{1,5}$
station $S_{1,6}$
assembly line 1
7 9 3 4 8 4
2
3 1 3 4
3
chassis enters
completed auto exits
2 2 1 2 2 1
4
2
assembly line 2
8 5 6 4 5 7
station $S_{2,1}$
station $S_{2,2}$
station $S_{2,3}$
station $S_{2,4}$
station $S_{2,5}$
station $S_{2,6}$

练习2

- 设有一个长度为N的数字串，要求选手使用K个乘号将它分成$K+1$个部分，找出一种分法，使得这$K+1$个部分的乘积能够为最大。
- 有一个数字串：312，当$N=3, K=1$时会有以下两种分法：
  1. $3 \times 12 = 36$
  2. $31 \times 2 = 62$
- 这时，符合题目要求的结果是：$31 \times 2 = 62$