



北京理工大学  
BEIJING INSTITUTE OF TECHNOLOGY

# 算法与计算理论

# 课程内容



## 数据结构

概述

线性表

栈与队列

数组与广义表

串

树

图

查找

内部排序

外部排序



## 算法与计算理论

概述

分治

动态规划

贪心

回溯

.....

.....

.....

.....

计算模型

可计算理论

计算复杂性



动态规划的一般原理

动态规划算法的求解过程

经典问题求解

# 动态规划与分治

分治的过程:

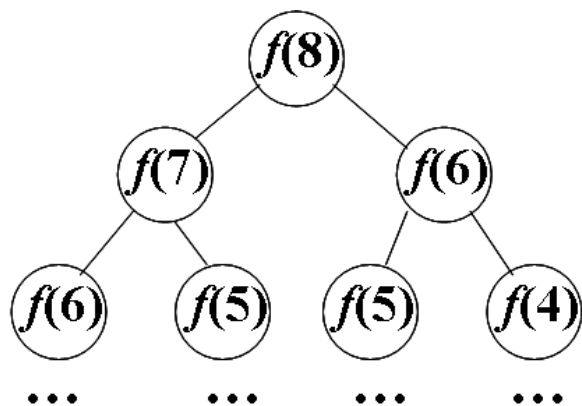
分解—递归解子问题—合并

若有大量重复子问题, 则不宜分治

举例: Fibonacci数的计算

$$f(n) = f(n-1) + f(n-2), f(1)=1, f(0)=0.$$

输入 $n$ , 输出 $f(n)$ .



递归:

```
int f(int n)
{ if(n<2) return(1);
  return(f(n-1)+f(n-2));}
```

$2^{O(n)}$ 时间,  **$O(n)$ 空间**

# 动态规划与分治

分治的过程:

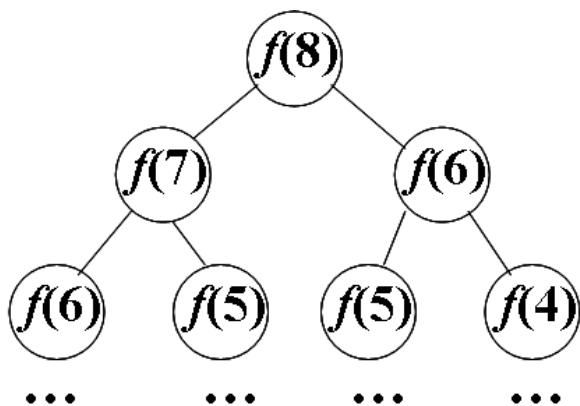
分解—递归解子问题—合并

若有大量重复子问题, 则不宜分治

举例: Fibonacci数的计算

$$f(n) = f(n-1) + f(n-2), f(1)=1, f(0)=0.$$

输入  $n$ , 输出  $f(n)$ .



Programming的含义:  
打表记录中间结果

动态规划:

```
f[1]=1;f[0]=0;
for(i=2,i<n,i++)
    f[i]=f[i-1]+f[i-2];
O(n)时间, O(n)空间
```

```
f=1; b=0;i=1;
while(i++<n){
    temp=f;f=f+b;b=temp;}
O(n)时间, O(1)空间
```

## DP适用条件

最优子结构性质, optimal substructure property

OSP: 最优策略的子策略也是最优.

重叠子结构性质

Programming是指使用表格化的算法.

Bellman, 1955, 奠定DP数学基础.

"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."

It can be summarized simply as follows: "every optimal policy consists only of optimal sub policies."

## DP适用条件-无后效性

动态规划中的无后效性指的是一旦某个阶段的状态确定后，这个状态对于未来决策的影响是固定的，即未来的子问题的解决不会影响到当前状态的决策。在动态规划中，这种无后效性有两个方面的体现：

**状态无后效性：**动态规划将整个决策过程划分为一系列阶段，每个阶段都有一个或多个状态。一旦某个阶段的状态确定，就完全决定了这个阶段以及所有后续阶段的最优决策。

**决策无后效性：**在每个阶段，当决定采取某个决策之后，这个决策如何影响当前阶段和所有后续阶段的最优性是确定的。

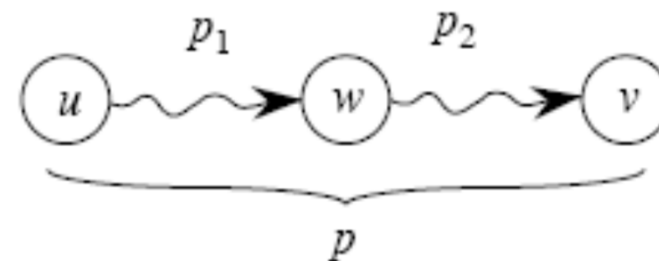
**动态规划的无后效性允许我们使用过去计算的结果（子问题的解）来解决当前的问题，而无需重新计算。**这就是动态规划中“记忆”子问题解的重要性，因为每个子问题的解都是独立的，不会因为解决其他子问题的方式而改变。



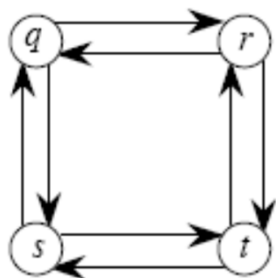
## DP适用条件-无后效性

无权最短路径: 边数最少. 具有最优子结构性质.

无权最长路径: 一定简单.



if we decompose a longest simple path into subpaths ,  
then mustn't  $p_1$  be a longest simple path from  $u$  to  $w$ ,  
and mustn't  $p_2$  be a longest simple path from  $w$  to  $v$ ?  
The answer is no!



Consider  $q \rightarrow r \rightarrow t =$  longest path  $q \rightsquigarrow t$ . Are its subpaths longest paths?

No!

- Subpath  $q \rightsquigarrow r$  is  $q \rightarrow r$ .
- Longest simple path  $q \rightsquigarrow r$  is  $q \rightarrow s \rightarrow t \rightarrow r$ .
- Subpath  $r \rightsquigarrow t$  is  $r \rightarrow t$ .
- Longest simple path  $r \rightsquigarrow t$  is  $r \rightarrow q \rightarrow s \rightarrow t$ .





动态规划的一般原理

动态规划算法的求解过程

经典问题求解

## [王]设计步骤

- 1) 描述最优解的结构
- 2) 递归定义最优解
- 3) 自底向上计算最优值
- 4) 由计算结果构造最优解

## 矩阵连乘问题([王])

输入: 给定矩阵  $A_1, A_2, \dots, A_n$ ,  $A_i$  与  $A_{i+1}$  可乘

输出: 计算量最小的乘法次序

输入样例:  $A_1$ (10×100阶),  $A_2$ (100×5阶),  $A_3$ (5×50阶)

两种计算次序:  $((A_1 \times A_2) \times A_3)$ ,  $(A_1 \times (A_2 \times A_3))$

$A_1 \times A_2$  的计算量:  $10 \times 100 \times 5$  (乘法次数)

$((A_1 \times A_2) \times A_3) : 10 \times 100 \times 5 + 10 \times 5 \times 50 = 7500$

$(A_1 \times (A_2 \times A_3)) : 100 \times 5 \times 50 + 10 \times 100 \times 50 = 75000$

样例输出:  $((A_1 \times A_2) \times A_3)$

## 矩阵连乘问题([王])

取整数序列  $q_0, q_1, \dots, q_n$  设  $A_i$  是  $q_{i-1} \times q_i$  阶矩阵  
n个矩阵连乘不同次序个数:

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

$\Rightarrow p(n) = C(n+1)$  为 *Catalan* 数

$$C(n) = \frac{1}{n-1} \binom{2n-4}{n-2} = \Omega\left(\frac{4^n}{n^{3/2}}\right) \quad \text{呈指数增长}$$

## 分析最优解结构、建立递推关系

假设定好了  $A_1 \dots A_n$  一个乘法次序  $P$

- 用  $A[i,j]$  记连乘积  $A_i \dots A_j$ , 相应计算量  $T[i,j]$
- 设  $P$  最后乘法在  $A_k$  后断开, 即  $A[1:k] \times A[k+1:n]$   
那么  $P$  的计算量为  $T[1,k] + T[k+1,n] + q_0 \times q_k \times q_n$
- 若  $P$  最优, 则  $P$  在  $A[1:k]$  和  $A[k+1:n]$  上也最优

**最优子结构性质**: 最优策略的子策略也是最优.

设  $A[i,j]$  的最小计算量为  $m[i,j]$ , 那么

$$m[i, j] = \begin{cases} 0 & i = j \\ \min_{i \leq k < j} \{m[i, k] + m[k + 1, j] + q_{i-1} q_k q_j\} & i < j \end{cases}$$

## 最优值与最优解的区别

$$m[i, j] = \begin{cases} 0 & i = j \\ \min_{i \leq k < j} \{m[i, k] + m[k + 1, j] + q_{i-1}q_kq_j\} & i < j \end{cases}$$

输入: 给定矩阵  $A_1, A_2, \dots, A_n$  的维数序列:

$q_0, q_1, \dots, q_n$ , 即  $A_i$  是  $q_{i-1} \times q_i$  阶矩阵

输出: 计算量最小的乘法次序

**最优解**是要输出的次序

**最优值**是最优解的计算量

# DP适用条件 and 设计步骤

OSP: 最优策略的子策略也是最优.

重叠子问题性质: 记录中间结果.

- 设计步骤
- 1) 描述最优解的结构
  - 2) 递归定义最优解
  - 3) 自底向上计算最优值
  - 4) 由计算结果构造最优解

$$m[i, j] = \begin{cases} 0 & i = j \\ \min_{i \leq k < j} \{m[i, k] + m[k + 1, j] + q_{i-1}q_kq_j\} & i < j \end{cases}$$



## 观察最优值计算

$$m[i, j] = \begin{cases} 0 & i = j \\ \min_{i \leq k < j} \{m[i, k] + m[k + 1, j] + q_{i-1}q_kq_j\} & i < j \end{cases}$$

需要计算的是 $m[1, n]$ , 但没法直接计算

$$m[1, 1] = m[2, 2] = \dots = m[n, n] = 0$$

$$m[1, 2] = ?$$

$$m[1, 2] = q_0 \times q_1 \times q_2, m[2, 3] = q_1 \times q_2 \times q_3, m[3, 4], \dots$$

$$m[1, 3] = ?$$

$$\min\{m[1, 1] + m[2, 3] + q_0 \times q_1 \times q_3, m[1, 2] + m[3, 3] + q_0 \times q_2 \times q_3\}$$

自底向上计算; 表格化方法

# 计算最优值图示

$$m[i, j] = \begin{cases} 0 & i = j \\ \min_{i \leq k < j} \{m[i, k] + m[k + 1, j] + q_{i-1}q_kq_j\} & i < j \end{cases}$$

设维数序列为(30, 35, 15, 5, 10, 20, 25)

m		i					
		1	2	3	4	5	6
j	1						
	2						
	3						
	4						
	5						
	6						

m		i					
		1	2	3	4	5	6
j	1						
	2						
	3						
	4						
	5						
	6						

# 计算最优值图示

$$m[i, j] = \begin{cases} 0 & i = j \\ \min_{i \leq k < j} \{m[i, k] + m[k + 1, j] + q_{i-1}q_kq_j\} & i < j \end{cases}$$

(30, 35, 15, 5, 10, 20, 25)

$$m[1, 2] = q_0 \times q_1 \times q_2 = 30 \times 35 \times 15 = 15750$$

m		i					
		1	2	3	4	5	6
j	1						
	2						
	3						
	4						
	5						
	6						

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

# 计算最优值图示

(30, 35, 15, 5, 10, 20, 25)

$$m[i, j] = \begin{cases} 0 & i = j \\ \min_{i \leq k < j} \{m[i, k] + m[k+1, j] + q_{i-1}q_kq_j\} & i < j \end{cases}$$

$$m[1, 3] = \min \{ m[1, 1] + m[2, 3] + q_0 \times q_1 \times q_3, m[1, 2] + m[3, 3] + q_0 \times q_2 \times q_3 \}$$

$$= \min \{ 0 + 2625 + 30 \times 35 \times 5, 15750 + 0 + 30 \times 15 \times 5 \} = \min \{ 18000, 7875 \}$$

m		i					
		1	2	3	4	5	6
j	1						
	2						
	3						
	4						
	5						
	6						

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

# 计算最优值图示

(30, 35, 15, 5, 10, 20, 25)

$$m[i, j] = \begin{cases} 0 & i = j \\ \min_{i \leq k < j} \{m[i, k] + m[k + 1, j] + q_{i-1}q_kq_j\} & i < j \end{cases}$$

$$\begin{aligned} m[1, 4] &= \min \{m[1, 1] + m[2, 4] + q_0 \times q_1 \times q_4, \\ &\quad m[1, 2] + m[3, 4] + q_0 \times q_2 \times q_4, \\ &\quad m[1, 3] + m[4, 4] + q_0 \times q_3 \times q_4\} \\ &= \min \{0 + 4375 + 30 \times 35 \times 10, \\ &\quad 15750 + 750 + 30 \times 15 \times 10, \\ &\quad 7875 + 0 + 30 \times 5 \times 10\} \\ &= \min \{9375, 21000, 14875\} \end{aligned}$$

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

# 计算最优值图示

(30, 35, 15, 5, 10, 20, 25)

$$m[i, j] = \begin{cases} 0 & i = j \\ \min_{i \leq k < j} \{m[i, k] + m[k+1, j] + q_{i-1}q_kq_j\} & i < j \end{cases}$$

$$\begin{aligned} m[1, 5] &= \min \{ 9375 + 0 + 30 \times 10 \times 20, 7875 + 1000 + 30 \times 5 \times 20, \\ &\quad 15750 + 2500 + 30 \times 15 \times 20, 0 + 7125 + 30 \times 35 \times 20 \} \\ &= \min \{ 15750, 11875, 27250, 28125 \} \end{aligned}$$

m		i					
		1	2	3	4	5	6
j	1						
	2						
	3						
	4						
	5						
	6						

m		i					
		1	2	3	4	5	6
j	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

## 计算最优值算法

$$m[i, j] = \begin{cases} 0 & i = j \\ \min_{i \leq k < j} \{m[i, k] + m[k+1, j] + q_{i-1}q_kq_j\} & i < j \end{cases}$$

输入:  $n$ 和维数序列  $(q_0, q_1, \dots, q_n)$

1. 对  $i = 1$  到  $n$ ,  $m[i, i] = 0$ ,
2. 对  $r = 1$  到  $n-1$
3.   对  $i = 1$  到  $n-r$
4.        $j = i + r$ ;  $m[i, j] = \text{INF}$ ;
5.       对  $k = i$  到  $j-1$
6.            $t = m[i, k] + m[k+1, j] + q_{i-1} \times q_k \times q_j$ ,
7.           若  $m[i, j] > t$ , 则  $m[i, j] = t$
8. 输出:  $m[1, n]$

INF如何处理? 如何构造最优解?



## 计算最优值同时标记分断点

$$m[i, j] = \begin{cases} 0 & i = j \\ \min_{i \leq k < j} \{m[i, k] + m[k+1, j] + q_{i-1}q_kq_j\} & i < j \end{cases}$$

输入:  $n$ 和维数序列  $(q_0, q_1, \dots, q_n)$

1. 对  $i = 1$  到  $n$ ,  $m[i, i] = 0$ ,
2. 对  $r = 1$  到  $n-1$
3.   对  $i = 1$  到  $n-r$
4.        $j = i + r$ ;  $s[i, j] = i$ ;
5.        $m[i, j] = m[i, i] + m[i+1, j] + q_{i-1} \times q_i \times q_j$ ;
6.       对  $k = i + 1$  到  $j-1$
7.            $t = m[i, k] + m[k+1, j] + q_{i-1} \times q_k \times q_j$ ,
8.           若  $m[i, j] > t$ , 则  $m[i, j] = t$ ;  $s[i, j] = k$ ;

输出:  $s$  //  $s[i, j]$ 是计算  $[i, j]$ 段的分断点

## 构造最优解

Traceback(1, n, s) //输出最优解, s[i,j]是[i:j]的最优分断点

Traceback(i, j, s) //[C]

1. 若  $i == j$ , 打印 "A" , i

2. 否则 打印 "("

3.           Traceback(i, s[i,j], s)

4.           Traceback(s[i,j]+1,j,s)

5.           打印 ")"

10, 100, 5, 50

s[1,3]=2

输出样例:

( (A1 A2) A3 )

Traceback2(i,j,s) //[王], 书中解释不匹配

1. 若  $i == j$  返回

2. Traceback2(i, s[i,j],s)

3. Traceback2(s[i,j]+1,j,s)

4. 打印 "A" , i, "," , s[i,j], "× A" , s[i,j]+1, "," , j

输出样例

A 1,1 × A 2,2

A 1,2 × A 3,3

## DP适用条件 and 设计步骤

OSP: 最优策略的子策略也是最优.

重叠子问题性质: 记录中间结果.

设计步骤 1) 描述最优解的结构

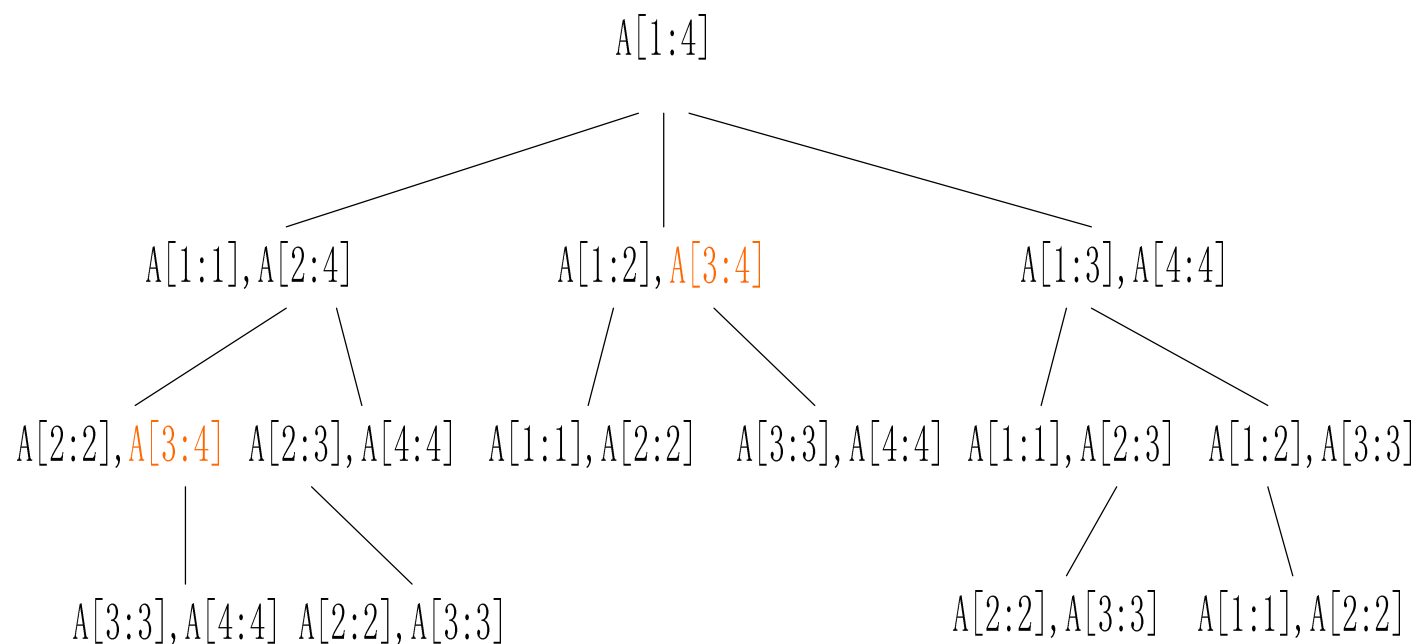
2) 递归定义最优解

3) 自底向上计算最优值 → 自顶向下

4) 由计算结果构造最优解

$$m[i, j] = \begin{cases} 0 & i = j \\ \min_{i \leq k < j} \{m[i, k] + m[k + 1, j] + q_{i-1}q_kq_j\} & i < j \end{cases}$$

自顶向下计算 $m[1][4]$ 过程如下:



设置备忘录

## 备忘录方法: 自顶向下

- MEMOIZED-MATRIX-CHAIN

1. 对所有  $i, j$ ,  $m[i, j] = 0$
2.  $LU(1, n)$       //LookUp

- $LU(i, j)$

1. 若  $m[i, j] > 0$ , 返回  $m[i, j]$
2. 若  $i = j$ , 返回 0
3.  $s[i, j] = i$ ;  $m[i, j] = LU(i, i) + LU(i+1, j) + q_{i-1} \times q_i \times q_j$
4. 对  $k = i + 1$  到  $j-1$
5.       $t = LU(i, k) + LU(k+1, j) + q_{i-1} \times q_k \times q_j$
6.      若  $m[i, j] > t$ , 则  $m[i, j] = t$ ;  $s[i, j] = k$ ;
7. 返回  $m[i][j]$

## 自底向上与自顶向下的比较

动态规划方法采用自底向上,通常采用备忘录来“记忆”所有子问题的解

备忘录技术有效地将动态规划问题的时间复杂度从指数级降低到了多项式级,尤其是在解决具有大量重叠子问题的情况时。

在实际编程中,备忘录技术可以显著减少不必要的计算,提高程序的性能。

在一个铁路沿线顺序存放着 $n$ 堆装满货物的集装箱。货物储运公司要将集装箱有次序地集中成一堆。规定每次只能选相邻的2堆集装箱合并成新的一堆，所需的运输费用与新的一堆中集装箱数成正比。给定各堆的集装箱数，试制定一个运输方案，使总运输费用最少。

输入：5, 3, 4, 2, 3, 4, 3, 4

为了简化，只算4, 2, 3, 4



# 练习

$$m[i, j] = \begin{cases} 0 & i = j \\ \min_{i < k \leq j} \{m[i, k-1] + m[k, j] + \sum_{t=i}^j a[t]\} & i < j \end{cases}$$

	$j = 1$	2	3	4
$i = 1$				
2				
3				
4				

4, 2, 3, 4

在一个圆形操场的四周顺序存放着 $n$ 堆装满货物的集装箱。货物储运公司要将集装箱有次序地集中成一堆。规定每次只能选相邻的2堆集装箱合并成新的一堆，所需的运输费用与新的一堆中集装箱数成正比。给定各堆的集装箱数，试制定一个运输方案，使总运输费用最少。

5, 3, 4, 2, 3, 4, 3, 4

为了简化，只算4, 2, 3, 4



# Contents

## 本章内容

动态规划的一般原理

动态规划算法的求解过程

经典问题求解

## 最长公共子序列(LCS, [王,M,C])

输入: 字符串 $X = x_1 x_2 \dots x_n$ ,  $Y = y_1 y_2 \dots y_m$

输出: X和Y最长的公共子序列(LCS)

样例:  $X = \text{ACTLE}$ ,  $Y = \text{WARE}$ , 输出: AE

若  $x_i = y_j$ , 则,  $(X_i Y_j)$  有LCS含  $(x_i, y_j)$  配对;

若  $x_i \neq y_j$ , 则,  $(X_i Y_j)$  的LCS是  $(X_{i-1}, Y_j)$  或  $(X_i Y_{j-1})$  的LCS.

例如:

ISIST

BIT

## 最长公共子序列(LCS, [王,M,C])

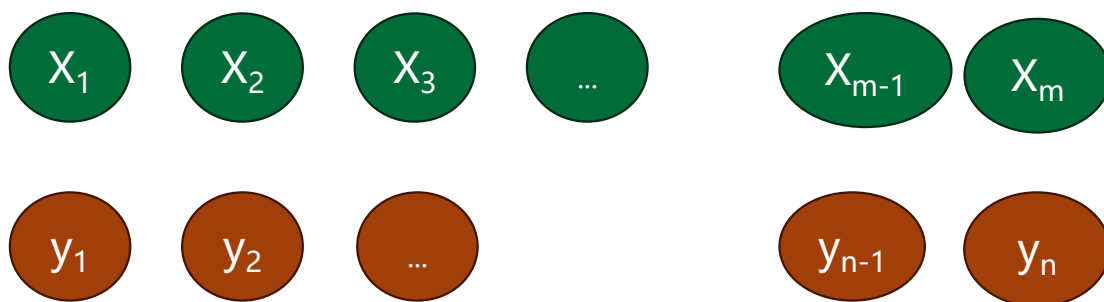
设序列 $X=\{x_1, x_2, \dots, x_m\}$ 和 $Y=\{y_1, y_2, \dots, y_n\}$ 的最长公共子序列为 $Z=\{z_1, z_2, \dots, z_k\}$ ，则

(1)若 $x_m = y_n$ ，则 $z_k = x_m = y_n$ ，且 $z_{k-1}$ 是 $x_{m-1}$ 和 $y_{n-1}$ 的最长公共子序列。

(2)若 $x_m \neq y_n$ 且 $z_k \neq x_m$ ，则 $Z$ 是 $x_{m-1}$ 和 $Y$ 的最长公共子序列。

(3)若 $x_m \neq y_n$ 且 $z_k \neq y_n$ ，则 $Z$ 是 $X$ 和 $y_{n-1}$ 的最长公共子序列。

由此可见，2个序列的最长公共子序列包含了这2个序列的前缀的最长公共子序列。  
因此，最长公共子序列问题具有最优子结构性质。



## LCS: 递归定义最优解

输入:  $X = x_1 x_2 \dots x_n$ ,  $Y = y_1 y_2 \dots y_m$ , 输出: X和Y的LCS

定义:  $c[i][j]$  =  $X_i Y_j$ 的LCS长度

$$c[i][j] = \begin{cases} 0 & i = 0 \text{ 或 } j = 0 \\ c[i-1][j-1] + 1 & i, j > 0 \text{ 且 } x_i = y_j \\ \max\{c[i][j-1], c[i-1][j]\} & i, j > 0 \text{ 且 } x_i \neq y_j \end{cases}$$

LCSlength( $n, m, x, y, c$ )

1. 初值  $c[0][1:m] = 0$ ,  $c[1:n][0] = 0$

2. 对  $i = 1:n$ ,  $j = 1:m$

3. 若  $x[i] = y[j]$ , 则  $c[i][j] = c[i-1][j-1] + 1$

4. 否则 若  $c[i][j-1] \geq c[i-1][j]$ , 则  $c[i][j] = c[i][j-1]$

5. 否则  $c[i][j] = c[i-1][j]$

输出  $c[n][m]$ .

## LC: 构造最优解

LCS( )

...

```

1: if (x[i]==y[j]) {
2:   c[i][j]=c[i-1][j-1]+1; ↗
3:   b[i][j]=1;}
4: else if (c[i-1][j]>=c[i][j-1]) {
5:   c[i][j]=c[i-1][j];      ↑
6:   b[i][j]=2;}
7: else {
8:   c[i][j]=c[i][j-1]; ←
9:   b[i][j]=3;}
    
```

		j	0	1	2	3	4	5	6
			y <sub>j</sub>						
				B	D	C	A	B	A
i	x <sub>i</sub>								
0		0	0	0	0	0	0	0	0
1	A	0	↑	↑	↑	↖	←	←	↖
2	B	0	↖	1	←	←	↑	↖	←
3	C	0	↑	↑	↑	2	←	↑	↑
4	B	0	↖	1	↑	↑	↑	↖	←
5	D	0	↑	↖	2	↑	↑	↑	↑
6	A	0	↑	↑	↑	↑	↖	↑	↖
7	B	0	↖	↑	↑	↑	↑	↖	↑



# 最长公共子序列

输入: 字符串 $X = x_1 x_2 \dots x_n$ ,  $Y = y_1 y_2 \dots y_m$

输出: X和Y最长的公共子序列(LCS)

例如:

$X = \text{"ABCB DAB"}$

$Y = \text{"BDCABA"}$

## 最大子段和

给定整数序列 $a_1, a_2, \dots, a_n$ , 求形如  $\sum_{k=i}^j a_k$  的子段和的最大值。  
规定子段和为负整数时, 定义其最大子段和为0, 即

$$\max \left\{ 0, \max_{1 \leq i \leq j \leq n} \sum_{k=i}^j a_k \right\}$$

例如,  $(a_1, a_2, a_3, a_4, a_5, a_6) = (-2, 11, -4, 13, -5, -2)$  最大子段和为

$$\sum_{k=2}^4 a_k = 20$$

## 最大子段和-动规

$$b[j] = \max\{b[j-1] + a_j, 0\} \quad 1 \leq j \leq n$$

```
int MaxSubSum3(int n, int a[])
{
    int sum=0, dp[0]=0; //sum存储当前最大
    for(int j=1; j<=n; j++) {
        if (dp[j-1]>=0)
            dp[j] += a[j];
        if(dp[j]<0) dp[j]=0; //若和为负，则从下一个位置累计
    }
    return sum;
}
```

例: (-1, 5, -3, 6, 4, 2, -7, 8)

运行时间:  $T(n)=O(n)$

## 最大子段和-动规

$$b[j] = \max\{b[j-1] + a_j, 0\} \quad 1 \leq j \leq n$$

```
int MaxSubSum3(int n, int a[])
{
    int sum=0, tms=0; //sum存储当前最大
    for(int j=1; j<=n; j++) {
        tms += a[j];
        if(tms<0) tms=0; //若和为负，则从下一个位置累和
        if(tms>sum) sum=tms;
    }
    return sum;
}
```

例: (-1, 5, -3, 6, 4, 2, -7, 8)

运行时间:  $T(n)=O(n)$

# 发射导弹

某国为了防御敌国的导弹袭击，发展出一种导弹拦截系统。但是这种导弹拦截系统有一个缺陷：虽然它的第一发炮弹能够达到任意的高度，但是以后每一发炮弹都不能高于前一发的高度。某天，雷达捕捉到敌国的导弹来袭。由于该系统还在试用阶段，所以只有一套系统，因此有可能不能拦截所有的导弹。

- ⑩  $D[i]$ :表示第  $i$  枚导弹被拦截以后, 还能最多拦截的导弹数 (包括被拦截的第  $i$  枚), 则

$$D[i] = \begin{cases} 1 & i = n \quad \text{or} \quad (i < j \leq n \text{ and } x(j) > x(i)) \\ \max \{D[j] + 1\} & i < n \quad (i < j \leq n \text{ and } x(j) \leq x(i)) \end{cases}$$

例如, 60, 40, 45, 70, 42

$i$	5	4	3	2	1
$D[i]$	1	2	2	1	3

## 最长递增子序列(LIS[M])

输入: 实数序列( $x_1, x_2, \dots, x_n$ )

输出:  $x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_k}$  其中k最大且  $i_1 < i_2 < \dots < i_k$

输入样例: ( 2,8,9,4,6,1,3,7,5,10)

输出样例: ( 2,4,6,7,10)

归纳尝试一: 已知[1:i]的1个LIS, 求[1:i+1]的LIS?

假设已知 (2,8,9,4,6,1,3) 的1个LIS (2,8,9), 加入7, 7不能加长(2,8,9), 但可加长(2,4,6). 启示,调整?

启示:  $x_{i+1}$ 可能可以使得其它LIS变长

调整: 记录尾巴最小的LIS: MTLIS

## 添加修改递归量

输入: 实数序列 $(x_1, x_2, \dots, x_n)$

输出:  $x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_k}$  其中k最大且  $i_1 < i_2 < \dots < i_k$

尾巴最小的LIS: MTLIS

归纳尝试二: 知 $[1:i]$ 的MTLIS, 求 $[1:i+1]$ 的MTLIS?

假设已知 $(2, 8, 9, 4)$  的MTLIS  $(2, 8, 9)$ , 加入6, 6不能加长 $(2, 8, 9)$ , 但 $(2, 4, 6)$ 是新的MTLIS. 启示调整?

启示:  $x_{i+1}$ 可能可以构成新的MTLIS

调整: 需要记录最长和次长的MTIS

调整: 需要记录长为k的MTIS: **MTIS(k)**,  $k=1, 2, \dots$

归纳尝试三: 知 $[1:i]$ 的MTIS(k), 求 $[1:i+1]$ 的MTIS(k)



# MTIS计算举例

输入: 实数序列( $x_1, x_2, \dots, x_n$ )

输出:  $x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_k}$  其中k最大且  $i_1 < i_2 < \dots < i_k$

长为k, 尾巴最小的IS: MTIS(k),  $k=1, 2, \dots$

归纳三: 知[1:i]的MTIS(k), 求[1:i+1]的MTIS(k)

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
MTIS(1)	2									
MTIS(2)	2	8								
MTIS(3)	2	8	9							

加入4?

能否得到MTIS(4)

能否改变MTIS(3)

能否改变MTIS(2)

能否改变MTIS(1)

# MTIS计算举例

输入: 实数序列( $x_1, x_2, \dots, x_n$ )

输出:  $x_{i1} \leq x_{i2} \leq \dots \leq x_{ik}$ , 其中 $k$ 最大且 $i_1 < i_2 < \dots < i_k$ .

长为 $k$ , 尾巴最小的IS: MTIS( $k$ ),  $k=1, 2, \dots$

归纳三: 知 $[1:i]$ 的MTIS( $k$ ), 求 $[1:i+1]$ 的MTIS( $k$ )

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
MTIS(1)	2									
MTIS(2)	2			4						
MTIS(3)	2	8	9							

加入6?

能否得到MTIS(4)

能否改变MTIS(3)

能否改变MTIS(2)

能否改变MTIS(1)

# MTIS计算举例

输入: 实数序列( $x_1, x_2, \dots, x_n$ )

输出:  $x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_k}$ , 其中 $k$ 最大且 $i_1 < i_2 < \dots < i_k$ .

长为 $k$ , 尾巴最小的IS: MTIS( $k$ ),  $k=1, 2, \dots$

归纳三: 知 $[1:i]$ 的MTIS( $k$ ), 求 $[1:i+1]$ 的MTIS( $k$ )

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
MTIS(1)	2									
MTIS(2)	2			4						
MTIS(3)	2			4	6					

加入1?

# MTIS计算举例

输入: 实数序列( $x_1, x_2, \dots, x_n$ )

输出:  $x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_k}$ , 其中 $k$ 最大且 $i_1 < i_2 < \dots < i_k$ .

长为 $k$ , 尾巴最小的IS: MTIS( $k$ ),  $k=1, 2, \dots$

归纳三: 知 $[1:i]$ 的MTIS( $k$ ), 求 $[1:i+1]$ 的MTIS( $k$ )

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
MTIS(1)						1				
MTIS(2)	2			4						
MTIS(3)	2			4	6					

# MTIS计算举例

输入: 实数序列( $x_1, x_2, \dots, x_n$ )

输出:  $x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_k}$ , 其中 $k$ 最大且 $i_1 < i_2 < \dots < i_k$ .

长为 $k$ , 尾巴最小的IS: MTIS( $k$ ),  $k=1, 2, \dots$

归纳三: 知 $[1:i]$ 的MTIS( $k$ ), 求 $[1:i+1]$ 的MTIS( $k$ )

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
MTIS(1)						1				
MTIS(2)						1	3			
MTIS(3)	2			4	6					

# MTIS计算举例

输入: 实数序列( $x_1, x_2, \dots, x_n$ )

输出:  $x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_k}$ , 其中 $k$ 最大且 $i_1 < i_2 < \dots < i_k$ .

长为 $k$ , 尾巴最小的IS: MTIS( $k$ ),  $k=1, 2, \dots$

归纳三: 知 $[1:i]$ 的MTIS( $k$ ), 求 $[1:i+1]$ 的MTIS( $k$ )

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
MTIS(1)						1				
MTIS(2)						1	3			
MTIS(3)	2			4	6					
MTIS(4)	2			4	6			7		

# MTIS计算举例

输入: 实数序列( $x_1, x_2, \dots, x_n$ )

输出:  $x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_k}$ , 其中 $k$ 最大且 $i_1 < i_2 < \dots < i_k$ .

长为 $k$ , 尾巴最小的IS: MTIS( $k$ ),  $k=1, 2, \dots$

归纳三: 知 $[1:i]$ 的MTIS( $k$ ), 求 $[1:i+1]$ 的MTIS( $k$ )

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
MTIS(1)						1				
MTIS(2)						1	3			
MTIS(3)						1	3		5	
MTIS(4)	2			4	6			7		

# MTIS计算举例

输入: 实数序列( $x_1, x_2, \dots, x_n$ )

输出:  $x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_k}$ , 其中 $k$ 最大且 $i_1 < i_2 < \dots < i_k$ .

长为 $k$ , 尾巴最小的IS: MTIS( $k$ ),  $k=1, 2, \dots$

归纳三: 知 $[1:i]$ 的MTIS( $k$ ), 求 $[1:i+1]$ 的MTIS( $k$ )

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
MTIS(1)						1				
MTIS(2)						1	3			
MTIS(3)						1	3		5	
MTIS(4)	2			4	6			7		
MTIS(5)	2			4	6			7		10

时间复杂度?

$O(n^3)$ ?

只记录尾巴?

$O(n^2)$ ?



# MTIS计算举例

输入: 实数序列( $x_1, x_2, \dots, x_n$ )

输出:  $x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_k}$ , 其中 $k$ 最大且 $i_1 < i_2 < \dots < i_k$ .

长为 $k$ , 尾巴最小的IS: MTIS( $k$ ),  $k=1, 2, \dots$

归纳三: 知 $[1:i]$ 的MTIS( $k$ ), 求 $[1:i+1]$ 的MTIS( $k$ )

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
MTIS(1)						1				
MTIS(2)						1	3			
MTIS(3)						1	3		5	
MTIS(4)	2			4	6			7		

尾巴递增?

有且仅有一个尾巴改变?

MTIS( $k$ ).last

len: 最大长度

- $x_{i+1} < \text{MTIS}(1).\text{last}$
- $x_{i+1} \geq \text{MTIS}(\text{len}).\text{last}$
- $x_{i+1} \geq \text{MTIS}(s-1).\text{last}$   
 $x_{i+1} < \text{MTIS}(s).\text{last}$

# MTIS计算举例

输入: 实数序列( $x_1, x_2, \dots, x_n$ )

输出:  $x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_k}$ , 其中 $k$ 最大且 $i_1 < i_2 < \dots < i_k$ .

长为 $k$ , 尾巴最小的IS: MTIS( $k$ ),  $k=1, 2, \dots$

归纳三: 知 $[1:i]$ 的MTIS( $k$ ), 求 $[1:i+1]$ 的MTIS( $k$ )

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
MTIS(1)	2									
MTIS(2)	2			4						
MTIS(3)	2			4	6					

尾巴递增?

有且仅有一个尾巴改变?

MTIS( $k$ ).last

len: 最大长度

- $x_{i+1} < \text{MTIS}(1).\text{last}$
- $x_{i+1} \geq \text{MTIS}(\text{len}).\text{last}$
- $x_{i+1} \geq \text{MTIS}(s-1).\text{last}$   
 $x_{i+1} < \text{MTIS}(s).\text{last}$

# MTIS计算举例

输入: 实数序列( $x_1, x_2, \dots, x_n$ )

输出:  $x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_k}$ , 其中 $k$ 最大且 $i_1 < i_2 < \dots < i_k$ .

长为 $k$ , 尾巴最小的IS: MTIS( $k$ ),  $k=1, 2, \dots$

归纳三: 知 $[1:i]$ 的MTIS( $k$ ), 求 $[1:i+1]$ 的MTIS( $k$ )

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
MTIS(1)						1				
MTIS(2)						1	3			
MTIS(3)	2			4	6					
MTIS(4)	2			4	6			7		

尾巴递增?

有且仅有一个尾巴改变?

MTIS( $k$ ).last

len: 最大长度

- $x_{i+1} < \text{MTIS}(1).\text{last}$
- $x_{i+1} \geq \text{MTIS}(\text{len}).\text{last}$
- $x_{i+1} \geq \text{MTIS}(s-1).\text{last}$   
 $x_{i+1} < \text{MTIS}(s).\text{last}$

## 只记录尾巴, 增加父亲标记

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
父亲	0									
MTIS(1).last	2									
MTIS(2).last										
MTIS(3).last										
MTIS(4).last										
MTIS(5).last										

# 只记录尾巴, 增加父亲标记

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
父亲	0	1								
MTIS(1).last	2									
MTIS(2).last		8								
MTIS(3).last										
MTIS(4).last										
MTIS(5).last										

## 只记录尾巴, 增加父亲标记

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
父亲	0	1	2							
MTIS(1).last	2									
MTIS(2).last		8								
MTIS(3).last			9							
MTIS(4).last										
MTIS(5).last										

## 只记录尾巴, 增加父亲标记

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
父亲	0	1	2	1						
MTIS(1).last	2									
MTIS(2).last				4						
MTIS(3).last			9							
MTIS(4).last										
MTIS(5).last										

# 只记录尾巴, 增加父亲标记

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
父亲	0	1	2	1	4					
MTIS(1).last	2									
MTIS(2).last				4						
MTIS(3).last					6					
MTIS(4).last										
MTIS(5).last										



# 只记录尾巴, 增加父亲标记

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
父亲	0	1	2	1	4	0				
MTIS(1).last						1				
MTIS(2).last				4						
MTIS(3).last					6					
MTIS(4).last										
MTIS(5).last										

## 只记录尾巴, 增加父亲标记

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
父亲	0	1	2	1	4	0	6			
MTIS(1).last						1				
MTIS(2).last							3			
MTIS(3).last					6					
MTIS(4).last										
MTIS(5).last										

## 只记录尾巴, 增加父亲标记

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
父亲	0	1	2	1	4	0	6	5		
MTIS(1).last						1				
MTIS(2).last							3			
MTIS(3).last					6					
MTIS(4).last								7		
MTIS(5).last										

## 只记录尾巴, 增加父亲标记

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
父亲	0	1	2	1	4	0	6	5	7	
MTIS(1).last						1				
MTIS(2).last							3			
MTIS(3).last									5	
MTIS(4).last								7		
MTIS(5).last										

# 只记录尾巴, 增加父亲标记

序号	1	2	3	4	5	6	7	8	9	10
序列	2	8	9	4	6	1	3	7	5	10
父亲	0	1	2	1	4	0	6	5	7	8
MTIS(1).last						1				
MTIS(2).last							3			
MTIS(3).last									5	
MTIS(4).last								7		
MTIS(5).last										10

时间复杂度:  $O(n^2) \rightarrow O(n \log n)$  ?

## MTIS计算举例

输入: 实数序列 $(x_1, x_2, \dots, x_n)$

输出:  $x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_k}$ , 其中k最大且 $i_1 < i_2 < \dots < i_k$

归纳三: 知 $[1:i]$ 的MTIS(k), 求 $[1:i+1]$ 的MTIS(k)

为什么每添加一数只有一个MTIS(k)会改变?

性质:  $\text{MTIS}(1).\text{last} \leq \text{MTIS}(2).\text{last} \leq \dots$

证明: 若 $\text{MTIS}(i).\text{last} > \text{MTIS}(i+1).\text{last}$ , 矛盾.

$x_{i+1}$ 改变MTIS(s):  $\text{MTIS}(s-1).\text{last} \leq x_{i+1} < \text{MTIS}(s).\text{last}$

怎么找修改位置?

采用二分搜索找s, 时间 $O(\log n) \times n$ .

[王]第三章习题1,2

# LIS算法

- 输入: 实数序列( $x_1, x_2, \dots, x_n$ )
- 输出:  $x_{i_1} \leq x_{i_2} \leq \dots \leq x_{i_k}$  其中k最大且  $i_1 < i_2 < \dots < i_k$
- 1. 数组X, L, F //X是输入, F[i]: 记录X[i]的父亲  
    //L[i]: X[L[i]]=MTIS(i).last, 即MTIS(i).last的位置
- 2. L[i] = 0, F[i] = 0; L[1] = 1, F[1] = 0, len = 1 //len:最大长度
- 3. 对于 i = 2 : n, //  $X[L[1]] \leq X[L[2]] \leq \dots \leq X[L[len]]$ ; X[i]?
- 4. 若  $X[L[1]] > X[i]$ , 则 F[i] = 0; L[1] = i.
- 5. 若  $X[L[len]] \leq X[i]$ , 则 F[i] = L[len]; len++; L[len] = i.
- 6. 否则 二分搜索L[1:len], 求s使得  $X[L[s-1]] \leq X[i] < X[L[s]]$
- 7. F[i]=L[s-1]; L[s]=i.
- 8. pt = L[len];
- 9. 对 i = len:1, D[i] = A[pt]; pt = F[pt]. 输出D.

## 0-1背包问题([王]p71)

输入:  $n$ 物品重 $W[1:n]$ , 价值 $V[1:n]$ , 背包容量 $C$

输出: 装包使得价值最大.

例:  $W:\{2,2,6,5,4\}$

$V:\{6,3,5,4,6\}$

$C=10$



## 0-1背包问题OSP

$$\begin{aligned} & \max \sum_{i=l}^n v_i x_i \\ & \begin{cases} \sum_{i=l}^n w_i x_i \leq c \\ x_i \in \{0,1\} \quad l \leq i \leq n \end{cases} \end{aligned} \quad \xRightarrow{\text{记}} \quad \text{Knap}(l, n, c)$$

- 证明：设 $(y_1, y_2, \dots, y_n)$ 是 $\text{Knap}(1, n, c)$ 的一个最优解，则
- $(y_2, \dots, y_n)$ 是 $\text{Knap}(2, n, c - w_1 y_1)$ 子问题的一个最优解。
- 若不然，设 $(z_2, \dots, z_n)$ 是 $\text{Knap}(2, n, c - w_1 y_1)$ 的最优解，因此有

$$\begin{aligned} & \sum_{i=2}^n v_i z_i > \sum_{i=2}^n v_i y_i \quad \text{且} \quad \sum_{i=2}^n w_i z_i \leq c - w_1 y_1 \\ & \Rightarrow \quad v_1 y_1 + \sum_{i=2}^n v_i z_i > \sum_{i=1}^n v_i y_i \quad \text{又有} \quad w_1 y_1 + \sum_{i=2}^n w_i z_i \leq c \end{aligned}$$

说明 $(y_1, z_2, \dots, z_n)$ 是 $\text{Knap}(1, n, c)$ 的一个更优解，矛盾。

## 0-1背包问题及(分数)背包OSP

背包的形式化描述是：给定 $c > 0, w_i > 0, v_i > 0, 1 \leq i \leq n$ ，要求找出一个 $n$ 元向量  $(x_1, x_2, x_3, \dots, x_n)$ ， $0 \leq x_i \leq 1, 1 \leq i \leq n$ ，使得

$$\sum_{i=1}^n w_i x_i \leq c \text{ 且 } \sum_{i=1}^n v_i x_i \quad \text{达到最大}$$

这两类问题都具有最优子结构性质。对于0-1背包问题，设 $A$ 是能够装入容量为 $c$ 的背包的具有最大价值的物品集合，则 $A_j = A - \{j\}$ 是 $n-1$ 个物品可装入容量为 $c - w_j$ 的背包的最大价值的物品集合。对于背包问题，类似的，若它的一个最优解包含物品 $j$ ，则从该最优解中拿出所含的物品 $j$ 的那部分重量 $w$ ，剩余的将是一个 $n-1$ 个原重物品 $1, 2, \dots, j-1, j+1, \dots, n$ 以及重为 $w_j - w$ 的物品 $j$ 中可装入容量 $c - w$ 的背包且具有最大价值的物品。

## 0-1背包问题

例：有5个物体，重量2,2,6,5,4，价值6,3,5,4,6，背包的载重量为10，求装入背包的物体及其总价值

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	6	6	<u>6</u>	6	6	6	6	6	6
2	0	0	6	6	9	9	<u>9</u>	9	9	9	9
3	0	0	6	6	9	9	9	9	11	11	14
4	0	0	6	6	9	9	9	10	11	13	14
5	0	0	6	6	9	9	12	12	15	15	<u>15</u>

$$dp_i(k) = \max\{dp_{i-1}(k), dp_{i-1}(k-w_i) + v_i\}$$

## 0-1背包: 编程

输入:  $n$ 物品重 $W[1:n]$ , 价值 $V[1:n]$ , 背包容量 $C$

输出: 装包使得价值最大 (物品重量为整数).

$[1:i]$ ,  $dp[i,k]$  = 由 $[1:i]$ 组合出重量 $\leq k$ 的最大价值

$dp[i,k] = \max\{ dp[i-1,k], dp[i-1,k-W[i]] + V[i] \}$  (OSP)

最优值:  $dp[n,C]$

1. 初始  $dp[0,0:C] = 0$ ,
2. 对  $i = 1:n$ , 对  $k = 1:C$ ,
3.  $dp[i,k] = dp[i-1,k]$
4. 若  $k \geq W[i]$ ,  $pt = dp[i-1,k-W[i]] + V[i]$ ;
5. 若  $pt > dp[i,k]$ ,  $dp[i,k] = pt$
6. 输出  $dp[n,C]$  //时间 $O(nC)$ .

## 0-1背包问题推广 找零钱问题

一出纳员支付一定数量的现金. 假设他手中有几种面值的硬币, 要求他用最少的硬币数支付规定的现金.

例如:

现有3种硬币: 它们的面值分别为1元、4元和6元. 要支付8元.

## 0-1背包问题推广 找零钱问题

$p_i(j)$ : 前  $i$  种硬币支付金额  $j$  所需硬币的最少个数。

$$p_i(j) = \begin{cases} p_{i-1}(j) & j < v_i \\ \min\{p_{i-1}(j), p_i(j-v_i) + 1\} & j \geq v_i \end{cases}$$

不用第  $i$  种硬币      用第  $i$  种硬币

$$p_i(0) = 0$$

$$p_0(j) = \infty$$

# 0-1背包问题推广 找零钱问题

$$p_i(j) = \begin{cases} p_{i-1}(j) & j < v_i \\ \min\{p_{i-1}(j), p_{i-1}(j-v_i) + 1\} & j \geq v_i \end{cases}$$

$V_i$	$i$	0	1	2	3	4	5	6	7	8
0	0	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	1									
4	2									
6	3									

现有3种硬币：它们的面值分别为1元、4元和6元. 要支付8元

# 0-1背包问题推广 找零钱问题

$$p_i(j) = \begin{cases} p_{i-1}(j) & j < v_i \\ \min\{p_{i-1}(j), p_{\textcolor{red}{i}}(j - v_i) + \textcolor{red}{1}\} & j \geq v_i \end{cases}$$

$V_i$	$i$	0	1	2	3	4	5	6	7	8
0	0	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	1	0	1	2	3	4	5	6	7	8
4	2	0	1	2	3	1	2	3	4	2
6	3	0	1	2	3	1	2	1	2	2

现有3种硬币：它们的面值分别为1元、4元和6元. 要支付8元



## (分数)背包问题(knapsack Prob)

### 0-1背包问题

输入:  $n$ 物品重 $W[1:n]$ ,  
价值 $V[1:n]$ ,  
背包容量 $C$

物品重量为整数.

输出: 装包使得价值最大.  
每件物品只能取或不取

$$\max \sum_{i=1}^n V_i x_i$$
$$\sum_{i=1}^n W_i x_i \leq C$$

$$x_i \in \{0,1\}, 1 \leq i \leq n$$

### (分数)背包问题

输入: 物品重 $W[1:n]$ , 价值 $V[1:n]$

输出: 装包使得价值最大.

物品 $i$ 可以取重量 $0 \sim W[i]$

$$\max \sum_{i=1}^n V_i x_i$$
$$\sum_{i=1}^n W_i x_i \leq C$$

$$0 \leq x_i \leq 1, 1 \leq i \leq n$$

## (分数)背包问题

(分数)背包问题

输入: 物品重 $W[1:n]$ , 价值 $V[1:n]$

输出: 装包使得价值最大.

物品 $i$ 可以取重量 $0 \sim W[i]$

$$\max \sum_{i=1}^n V_i x_i$$

$$\sum_{i=1}^n W_i x_i \leq C$$

$$0 \leq x_i \leq 1, 1 \leq i \leq n$$

贪心算法:

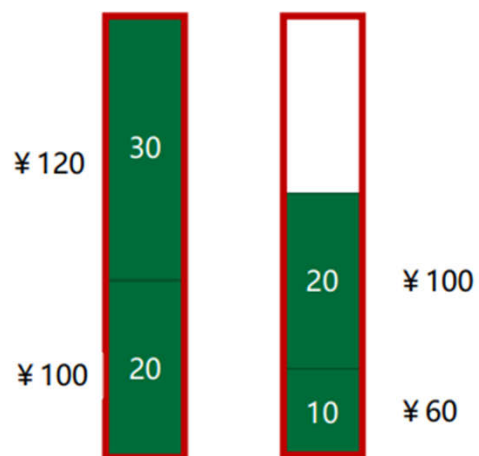
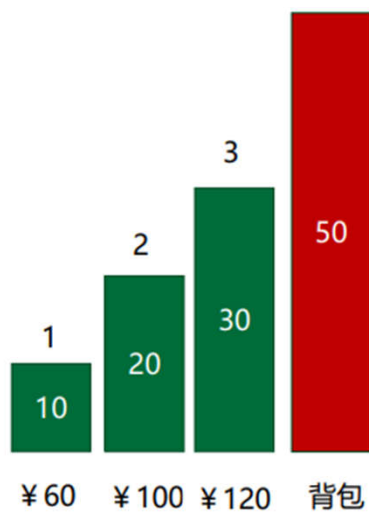
1. 对单位重量价值  $\{ V[i]/W[i] \}_{i=1}^n$  排序
2. 优先放入单位重量价值大的物品, 直到填满

$O(n \log n)$ ,

0-1背包能贪心吗?

# (分数)背包问题

0-1背包能贪心吗?



## 最优装载([王]p95)

### 0-1背包问题

输入:  $n$ 物品重 $W[1:n]$ ,  
价值 $V[1:n]$ ,  
背包容量 $C$

物品重量为整数.

输出: 装包使得价值最大.

每件物品只能取或不取

$$\max \sum_{i=1}^n V_i x_i$$

$$\sum_{i=1}^n W_i x_i \leq C$$

$$x_i \in \{0,1\}, 1 \leq i \leq n$$

---

### 最优装载

输入:  $n$ 物品重 $W[1:n]$ ,  
背包容量 $C$

输出: 装包使得件数最多.

每件物品只能取或不取

$$\max \sum_{i=1}^n x_i$$

$$\sum_{i=1}^n W_i x_i \leq C$$

$$x_i \in \{0,1\}, 1 \leq i \leq n$$

最优装载

输入:  $n$  物品重  $W[1:n]$ ,  
背包容量  $C$

输出: 装包使得件数最多.  
每件物品只能取或不取

$$\max \sum_{i=1}^n x_i$$

$$\sum_{i=1}^n W_i x_i \leq C$$

$$x_i \in \{0,1\}, 1 \leq i \leq n$$

---

贪心算法( $O(n \log n)$ ):

1. 对重量  $W[1:n]$  升序排列
2. 优先添加重量小的物品, 直到不能再添加

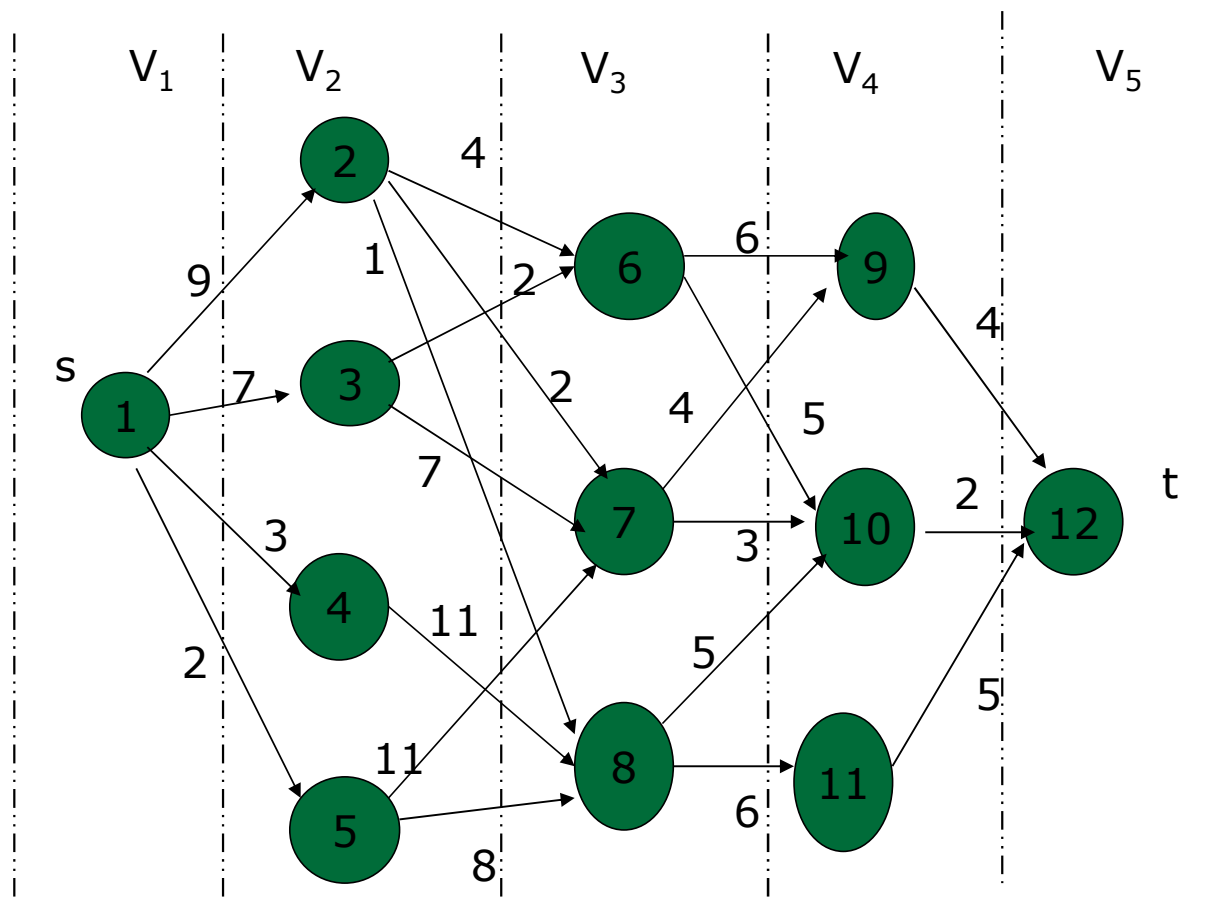
**贪心选择性质:** 最优解可以包含重量最小的

**OSP:** 最优解( $[1:n], C$ )去掉重量最小( $[2:n], C - W[1]$ )仍是最优解

## 多段图的最短路问题

多段图：设 $G=\langle V,E \rangle$ 是一个有向连通图，其中 $|V|=n$ ,  $|E|=m$ ,  $V$ 有划分 $\{V_1, V_2, \dots, V_k\}$ ，这里 $V_1 = \{s\}$ ,  $s$ 称为源点， $V_k = \{t\}$ ,  $t$ 称为终点，其中 $k \geq 2$ 。对于每条有向边 $\langle u, v \rangle \in E$ 都存在 $V_i \in V$ ，使得 $u \in V_i$ 和 $v \in V_{i+1}$ ，其中 $1 \leq i < k$ 且每条边 $\langle u, v \rangle$ 均附有代价 $C(u, v)$ ，则称 $G$ 是一个 $k$ 段图。

# 多段图的最短路问题



## 多段图的最短路问题

假设 $s, v_2, v_3, \dots, v_{k-1}, t$ 是一条从 $s$ 到 $t$ 的最短路径，还假定从源点 $s$ （初始状态）开始，已做出了到结点 $v_2$ 的决策（初始决策），因此 $v_2$ 就是初始决策所产生的状态。如果把 $v_2$ 看成是原问题的一个子问题的初始状态，解这个子问题就是找出一条由 $v_2$ 到 $t$ 的最短路径。这条路径显然是 $v_2, v_3, \dots, v_{k-1}, t$ ，否则设 $v_2, q_3, \dots, q_{k-1}, t$ 是一条由 $v_2$ 到 $t$ 的更短路径，则 $s, v_2, q_3, \dots, q_{k-1}, t$ 是一条比路径 $s, v_2, v_3, \dots, v_{k-1}, t$ 更短的由 $s$ 到 $t$ 的路径，与假设矛盾，故最优化原理成立。



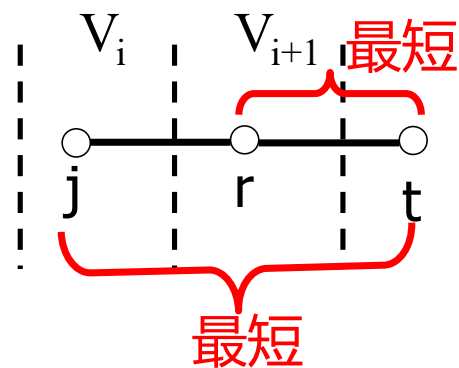
## 多段图的最短路问题

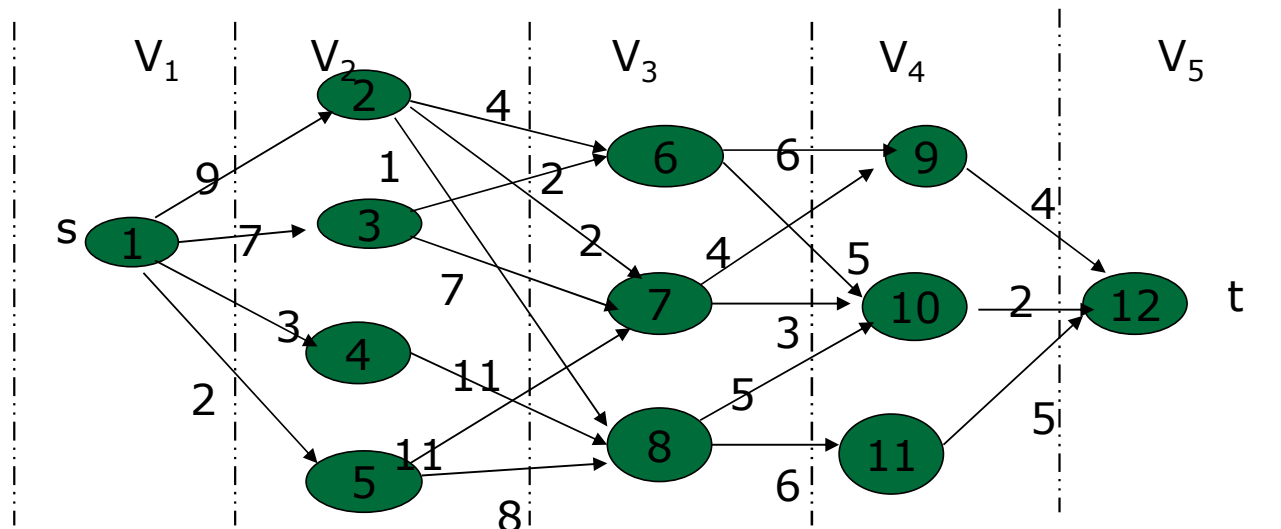
向前处理法：设 $P(i,j)$ 是从 $V_i$ 中的点 $j$ 到 $t$ 的一条最短路， $\text{cost}(i,j)$ 是这条路线的耗费。由后向前推算，得到

$$\text{cost}(i,j) = \min \{ C(j,r) + \text{cost}(i+1,r) \}$$

$$r \in V_{i+1}$$

$$\langle j,r \rangle \in E$$



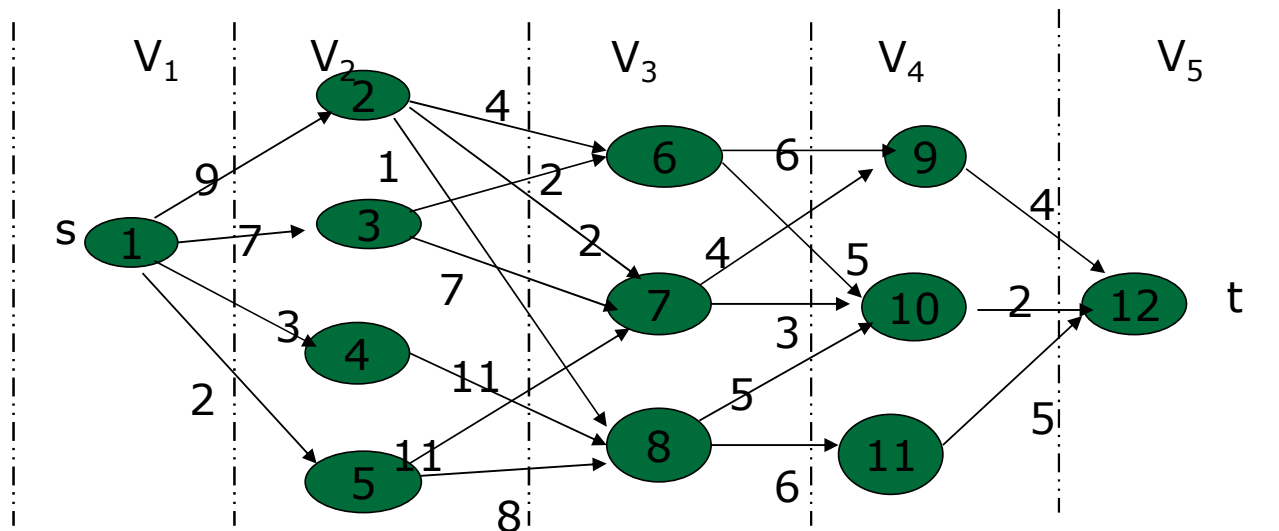


$$\text{cost}(i,j)=\min\{C(j,r)+\text{cost}(i+1,r)\}$$

$$\text{cost}(4,9)=4 \quad \text{cost}(4,10)=2 \quad \text{cost}(4,11)=5$$

$$\begin{aligned} \text{cost}(3,6) &= \min\{6+\text{cost}(4,9), 5+\text{cost}(4,10)\} \\ &= \min\{6+4, 5+2\} = 7 \end{aligned}$$

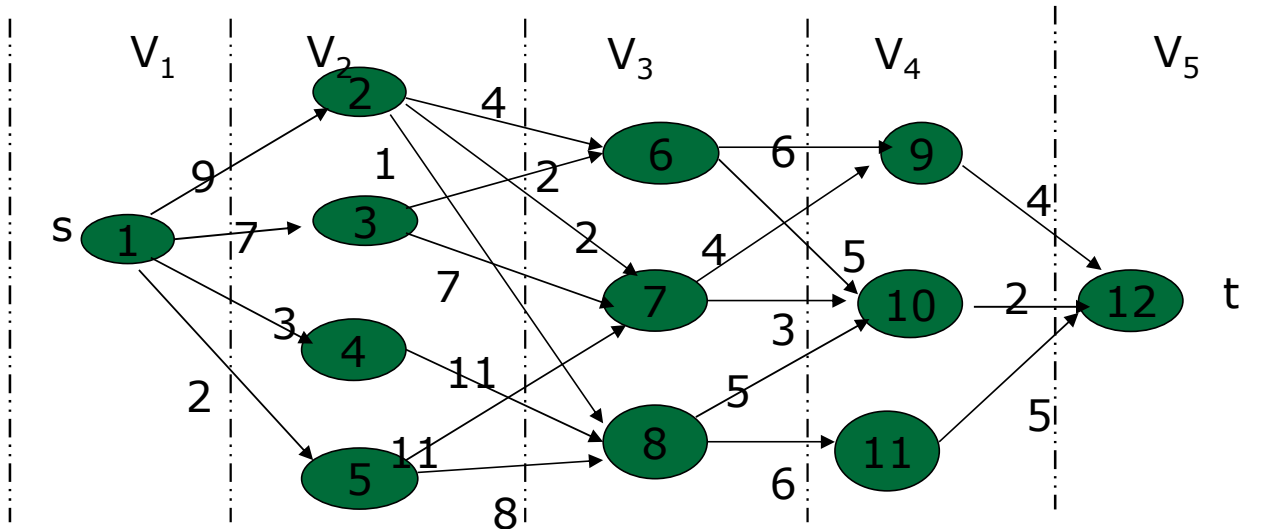
从第3段的顶点6到t的最短路径是6-10-t



$$\begin{aligned} \text{cost}(3,7) &= \min \{4 + \text{cost}(4,9), 3 + \text{cost}(4,10)\} \\ &= \min \{4 + 4, 3 + 2\} = 5 \end{aligned}$$

•从第3段的顶点7到t的最短路径是7-10-t

$\text{cost}(3,8)=7$   
•从第3段的顶点8到t的最短路径是8-10-t



$\text{cost}(2,2)=7$  从第2段顶点2到t的最短路是2-7-10-t

$\text{cost}(2,3)=9$  从第2段顶点3到t的最短路是3-6-10-t

$\text{cost}(2,4)=18$  从第2段顶点4到t的最短路是4-8-10-t

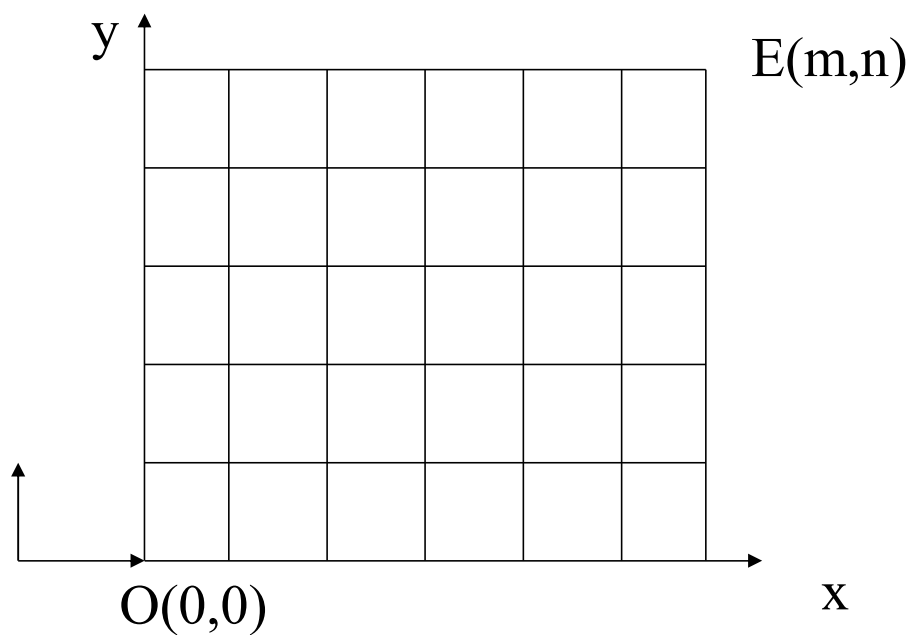
$\text{cost}(2,5)=15$  从第2段顶点5到t的最短路是5-8-10-t

$\text{cost}(1,1)=16$  从第1段顶点1到t的最短路径由两条：

1-2-7-10-t和1-3-6-10-t

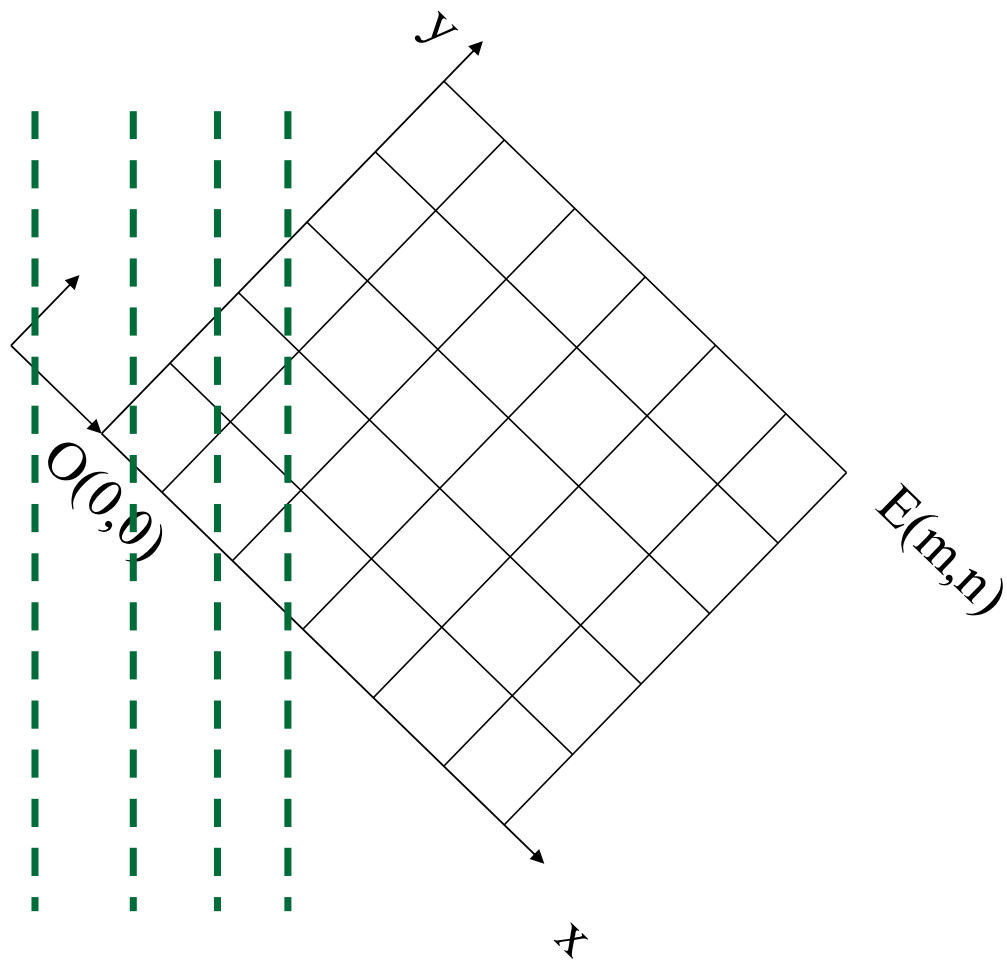
# 格路问题

求从起点 $O(0,0)$ 到终点 $E(m,n)$ 的最短路。则分别用穷举法和动态规划法的加法次数和比较次数各是多少？



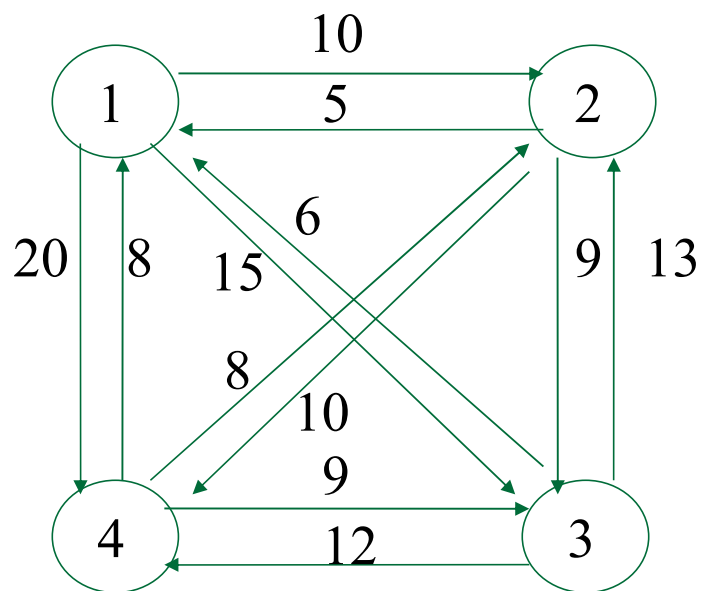
加法次数	比较次数

# 格路问题



点数	加法次数	比较次数

# 货郎担问题



	$V_1$	$V_2$	$V_3$	$V_4$
$V_1$	$\infty$	10	15	20
$V_2$	5	$\infty$	9	10
$V_3$	6	13	$\infty$	12
$V_4$	8	8	9	$\infty$

不失一般性，考虑以结点1为起点和终点的一条哈密顿回路。每一条这样的路线都由一条边 $\langle 1, k \rangle$ 和一条由结点k到结点1的路径组成，其中 $k \in V - \{1\}$ ，而这条由结点k到结点1的路径通过 $V - \{1, k\}$ 的每个结点各一次。如果这条周游路线是最优的，则这条由结点k到结点1的路径必定是通过 $V - \{1, k\}$ 的每个结点各一次的由k到1的最短路径。

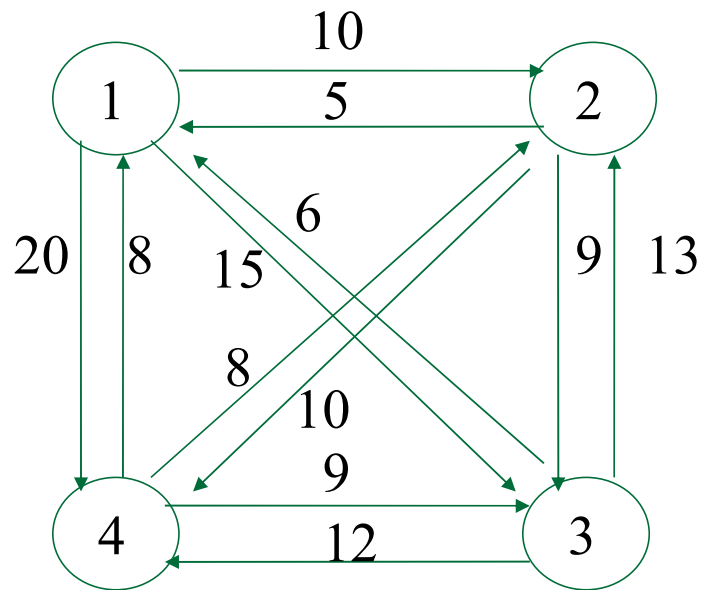


设 $T(i, S)$ 是由结点  $i$  出发, 经过结点集  $S$  中每个结点各一次并回到初始结点 1 的一条最短路径长度, 则  $T(1, V - \{1\})$  就是一条最优的周游路线长度。动态规划模型:

$$T(1, V - \{1\}) = \min \{d_{1k} + T(k, V - \{1, k\})\}$$
$$2 \leq k \leq n$$

$$T(i, s) = \min \{d_{ij} + T(j, S - \{j\})\}$$
$$j \in S, i \notin S$$

# 货郎担问题



	$V_1$	$V_2$	$V_3$	$V_4$
$V_1$	$\infty$	10	15	20
$V_2$	5	$\infty$	9	10
$V_3$	6	13	$\infty$	12
$V_4$	8	8	9	$\infty$

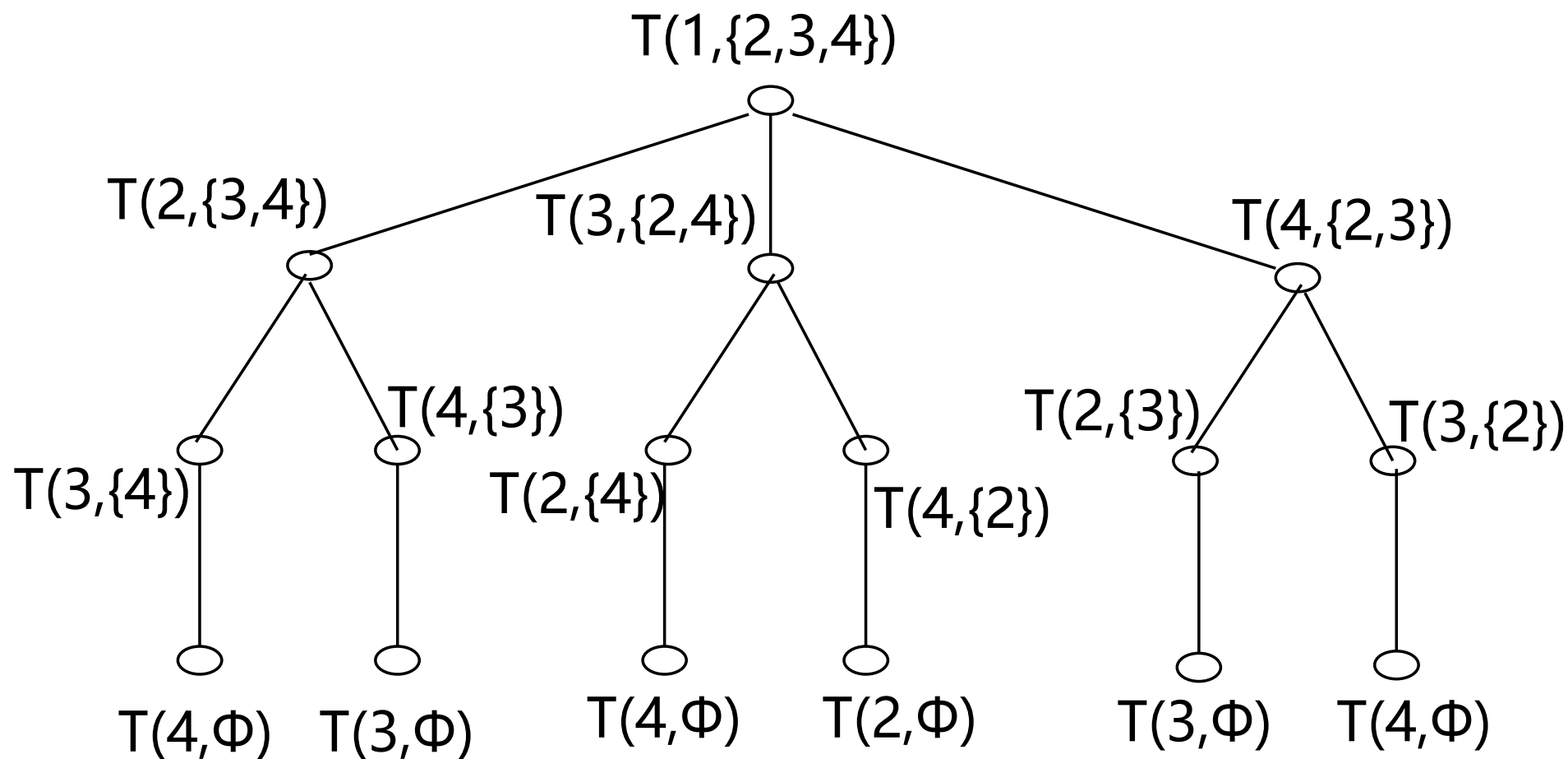
$$T(1, \{2,3,4\}) = \min \{ d_{12} + T(2, \{3,4\}), d_{13} + T(3, \{2,4\}), d_{14} + T(4, \{2,3\}) \}$$

$$T(2, \{3,4\}) = \min \{ d_{23} + T(3, \{4\}), d_{24} + T(4, \{3\}) \}$$

$$T(3, \{4\}) = d_{34} + T(4, \Phi)$$

$$T(4, \Phi) = d_{41}$$

# 货郎担问题



### 算法分析题

3. 考虑下面的整数线性规划问题 .

即给定 序列  $a_1, a_2, \dots, a_n$  求

$$\max c_1 x_1 + c_2 x_2 + \dots + c_n x_n$$

满足  $a_1 x_1 + a_2 x_2 + \dots + a_n x_n \leq b$ ,  $x_i$  为非负整数

## 本章作业

算法实现题:

数字三角形问题

问题描述: 给定一个有 $n$ 行数字组成的数字三角形, 如下图所示. 试设计一个算法, 计算出从三角形的顶至底的一条路径, 使该路径经过的数字和最大.

算法设计: 对于给定的 $n$ 行数字组成的三角形, 计算从三角形顶至底的路径经过的数字和的最大值.

数据输入: 由文件input.txt提供输入数据. 文件的第1行数字三角形的行数 $n$ ,  $1 \leq n \leq 100$ . 接下来 $n$ 行是数字三角形各行中的数字. 所有数字在 $0 \sim 99$ 之间.

结果输出: 将计算结果输出到文件output.txt, 文件第1行中的数是计算出的最大值.

```
  7
 3 8
8 1 0
2 7 4 4
4 5 2 6 5
数字三角形
```

输入文件示例

input.txt

```
5
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5
```

输出文件示例

output.txt

30

## 本章作业

算法实现题: 租用游艇问题

问题描述: 长江游艇俱乐部在长江上设置了 $n$ 个游艇出租站 $1, 2, \dots, n$ . 游客可在这些游艇出租站租用游艇, 并在下游的任何一个游艇出租站归还游艇. 游艇出租站 $i$ 到出租站 $j$ 之间的租金为 $r(i, j)$ ,  $1 \leq i < j \leq n$ . 试设计一个算法, 计算出从游艇出租站1到游艇出租站 $n$ 所需的最少租金, 并分析算法的计算复杂性.

算法设计: 对于给定的游艇出租站 $i$ 到游艇出租站 $j$ 的租金 $r(i, j)$ ,  $1 \leq i < j \leq n$ . 计算出出租站1到 $n$ 所需的最少租金.

数据输入: 由文件input.txt提供输入数据. 文件的第1行有一个正整数 $n$ ,  $n \leq 200$ , 表示有 $n$ 个游艇出租站. 接下来 $n-1$ 行是 $r(i, j)$ ,  $1 \leq i < j \leq n$ .

结果输出: 将计算出的游艇出租站1到 $n$ 最少租金输出到文件output.txt.

输入文件示例

input.txt

3

5 15

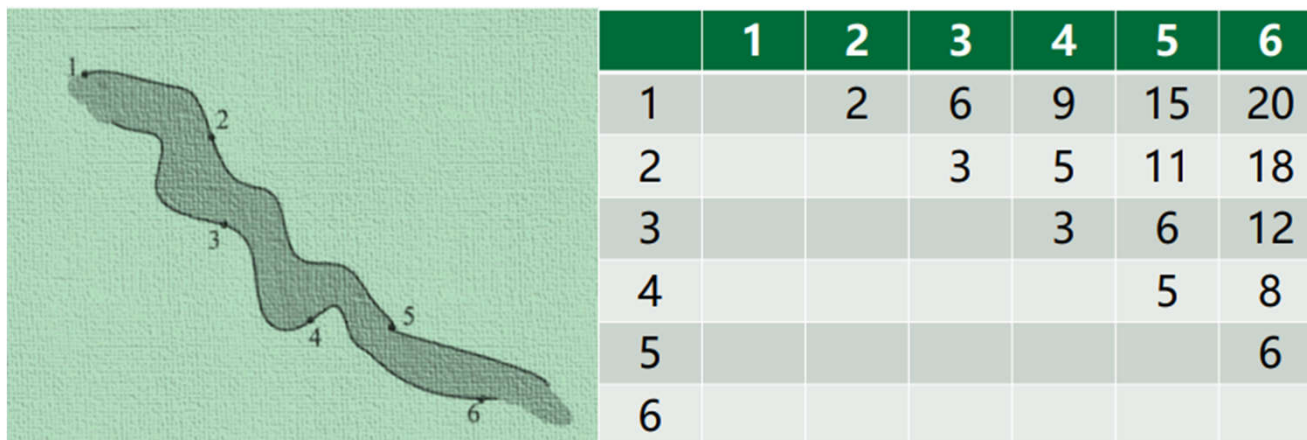
7

输出文件示例

output.txt

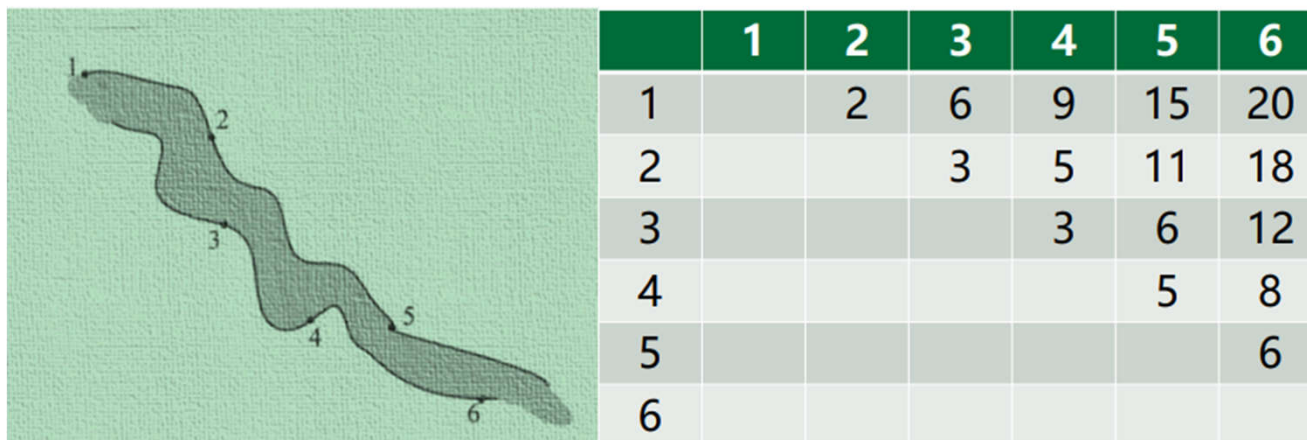
12

# 本章作业



$$m[i][j] = \begin{cases} 0 & , j = i \\ r[i][j] & , j = i + 1 \\ \min_{i < k < j} \{m[i][k] + m[k][j], r[i][j]\} & , j > i + 1 \end{cases}$$

# 本章作业



$$m[i][j] = \begin{cases} 0 & , j = i \\ r[i][j] & , j = i + 1 \\ \min_{i < k < j} \{m[i][k] + m[k][j], r[i][j]\} & , j > i + 1 \end{cases}$$

输入文件示例  
input.txt  
3  
5 15  
7

输出文件示例  
output.txt  
12



Michael喜欢滑雪百这并不奇怪， 因为滑雪的确很刺激。可是为了获得速度，滑的区域必须向下倾斜，而且当你滑到坡底，你不得不再次走上坡或者等待升降机来载你。

Michael想知道载一个区域中最长底滑坡。

区域由一个二维数组给出。

数组的每个数字代表点的高度。

一个人可以从某个点滑向上下左右相邻四个点之一，当且仅当高度减小。在右面的例子中，一条可滑行的滑坡为24-17-16-1。当然25-24-23-...-3-2-1更长。事实上，这是最长的一条。

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

# 字符串相似

大家都知道，基因可以看作一个碱基对序列。它包含了 4 种核苷酸，简记作 A,C,G,T。生物学家正致力于寻找人类基因的功能，以利用于诊断疾病和发明药物。在一个人类基因工作组的任务中，生物学家研究的是：两个基因的相似程度。因为这个研究对疾病的治疗有着非同寻常的作用。两个基因的相似度的计算方法如下：对于两个已知基因，例如 AGTGATG 和 GTTAG，将它们的碱基互相对应。当然，中间可以加入一些空碱基

A	G	T	G	A	T	-	G
-	G	T	-	-	T	A	G

# 字符串相似

	A	C	G	T	-
A	5	-1	-2	-1	-3
C	-1	5	-3	-2	-4
G	-2	-3	5	-2	-2
T	-1	-2	-2	5	-1
-	-3	-4	-2	-1	*

$$dp[i][j] = \max \begin{cases} dp[i-1][j] + S[x_i][-] \\ dp[i][j-1] + S[y_j][-] \\ dp[i-1][j-1] + S[x_i][y_j] \end{cases}$$

输入：  
AGTGATG  
GTTAG

A	G	T	G	A	T	G
-	G	T	T	A	-	G

相似度为：  $(-3)+5+5+(-2)+5+(-1)+5=14$

## 0-1背包问题推广 资源分配问题

某公司准备将4名销售员分配到3个销售点甲乙丙。 各销售点增加若干名销售员后可增加的月销售额如下表：

增加销售额（万元）	增1人	增2人	增3人	增4人
甲	12	22	30	38
乙	11	20	24	30
丙	13	25	30	36

公司每月最多可以增加销售额？ 万元

## 0-1背包问题推广 资源分配问题

某公司准备将4名销售员分配到3个销售点甲乙丙。 各销售点增加若干名销售员后可增加的月销售额如下表：

增加销售额（万元）	增1人	增2人	增3人	增4人
甲	12	22	30	38
乙	11	20	24	30
丙	13	25	30	36

公司每月最多可以增加销售额？ 万元

$$dp[i][n] = \max(dp[i][n], arr[i][k] + dp[i-1][n-k])$$

	甲	乙	丙
0	0	0	0
1	12	11	13
2	22		
3	30		
4	38		

## 本章小结

最优子结构性质OSP

动态规划算法的设计步骤

矩阵连乘

最长公共子序列

最大子段和

最长递增子序列

0-1背包, 分数背包, 最优装载

最短路

习题