软件工程专业导论

战德臣

哈尔滨工业大学　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

第4讲 软件之灵魂--算法

战德臣

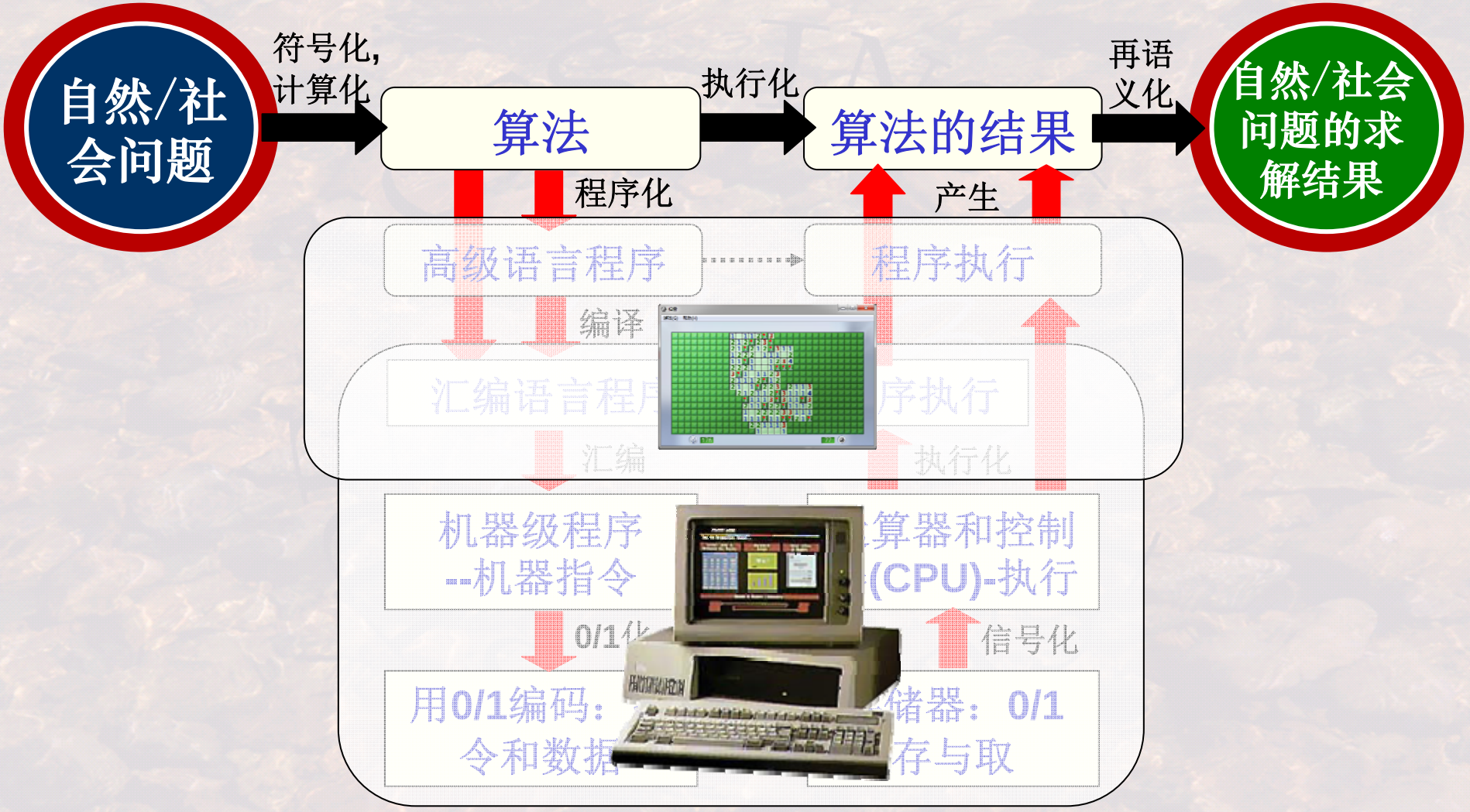
哈尔滨工业大学　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

软件之灵魂--算法
软件工程学科的创造能力

战德臣 教授



算法是软件的灵魂，利用软件系统进行问题求解的关键是发现、构造与设计求解问题的算法。

什么是与为什么需要算法?

战德臣

哈尔滨工业大学　教授.博士生导师

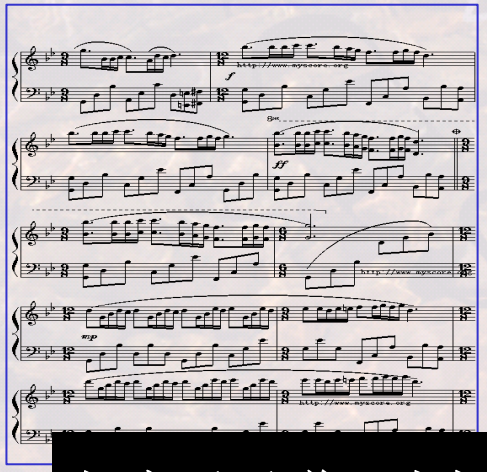
教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

什么是与为什么需要算法
(1)什么是算法?

算法---计算机与软件的灵魂。“算法”(Algorithm)一词源于数学家的名字：公元825年，阿拉伯数学家阿科瓦里茨米(AlKhowarizmi)写了著名的《波斯教科书》(Persian Textbook)，书中概括了进行四则算术运算的计算规则。

◆**算法**是一个有穷规则的集合，它用规则规定了解决某一特定类型问题的运算序列，或者规定了任务执行或问题求解的一系列步骤。



如音乐乐谱、太极拳谱等都可看作广义的算法

我们关注计算学科的算法
哟 … …

什么是与为什么需要算法
(2)一个典型的计算算法示例

战德臣 教授

寻找两个正整数的最大公约数的欧几里德算法

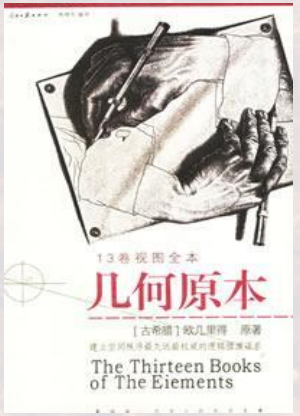
输入：正整数*M*和正整数*N*

输出：*M*和*N*的最大公约数(设M>N)

算法步骤：

Step 1. *M*除以*N*,记余数为*R*

Step 2. 如果*R*不是*0*，将*N*的值赋给*M*，*R*的值赋给*N*，返回Step 1；否则，最大公约数是*N*，输出*N*，算法结束。



	M	N	R	最大公约数
具体问题	32	24		?
算法计算过程				
1	32	24	8	
2	24	8	0	8
具体问题	31	11		?
算法计算过程				
1	31	11	9	
2	11	9	2	
3	9	2	1	
4	2	1	0	1

战德臣 教授

计算学科**算法**的基本特征

✓**有穷性**：一个算法在执行有穷步规则之后必须结束。

✓**确定性**：算法的每一个步骤必须要确切地定义，不得有歧义性。

✓**输入**：算法有零个或多个的输入。

✓**输出**：算法有一个或多个的输出/结果，即与输入有某个特定关系的量。

✓**能行性**：算法中有待执行的运算和操作必须是相当基本的**(可以由机器自动完成)**，并能在有限时间内完成。

寻找两个正整数的最大公约数的欧几里德算法

输入：正整数*M*和正整数*N*

输出：*M*和*N*的最大公约数(设M>N)

算法步骤：

Step 1. *M*除以*N*,记余数为*R*

Step 2. 如果*R*不是*0*，将*N*的值赋给*M*，*R*的值赋给*N*，返回**Step 1**；否则，最大公约数是*N*，输出*N*，算法结束。

基本运算：除法、赋值、逻辑判断

典型的“重复/循环”与“迭代”

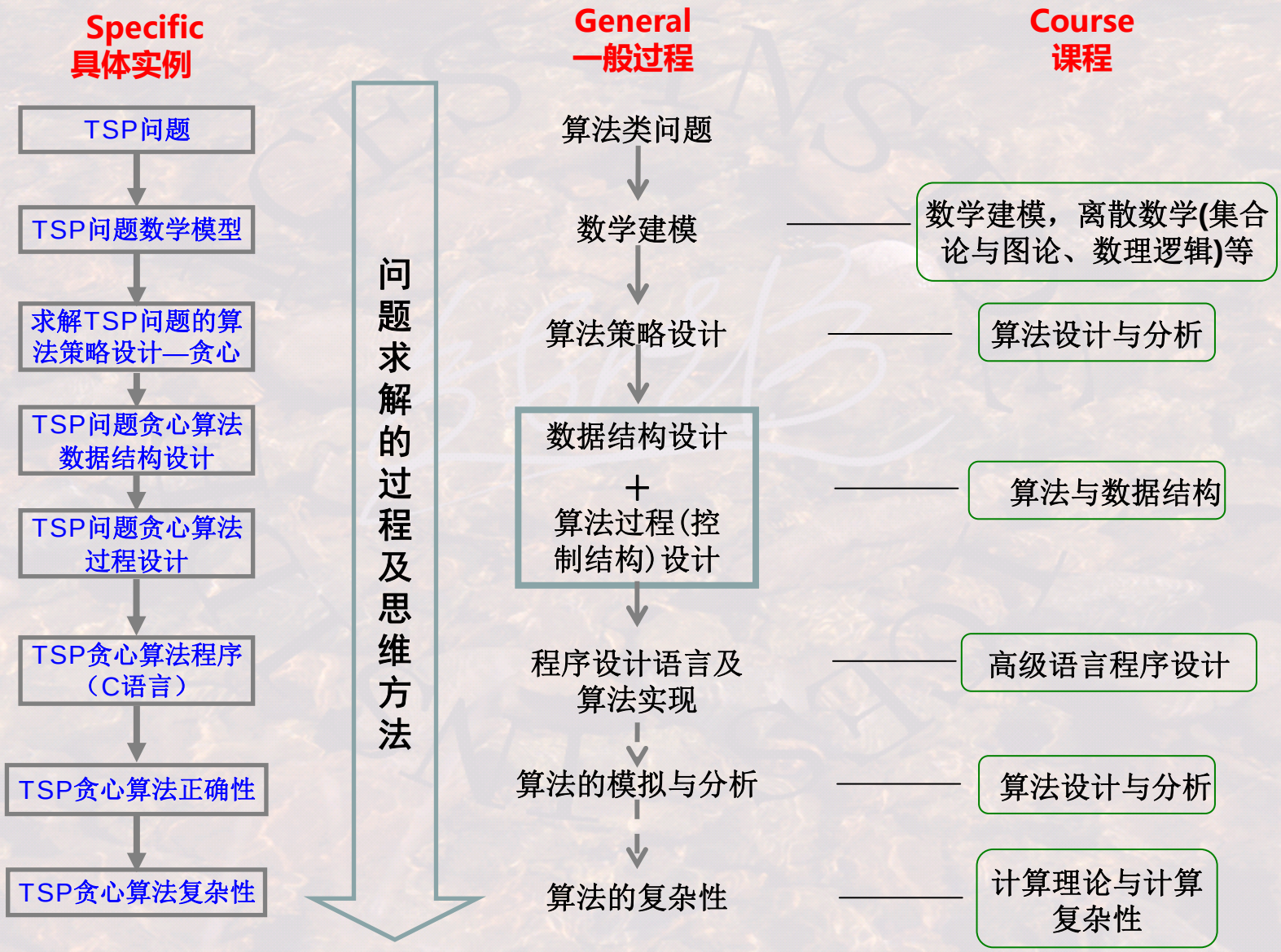
什么是与为什么需要算法
(4)为什么需要算法?

战德臣 教授

步骤书写的规范、语法规则、标准的集合
是人和计算机都能理解的语言

“是否会编程序”，本质上是“能否想出求解问题的算法”，其次才是将算法用计算机可以识别的形式书写出来

什么是与为什么需要算法
(5)算法相关的



问题求解第一步--数学建模

战德臣

哈尔滨工业大学　教授.博士生导师

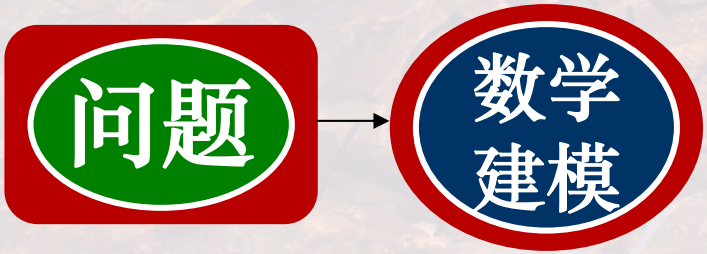
教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

问题求解第一步--数学建模?

(1)问题求解的第一步：数学建模

ICES 战德臣 教授



数学建模是用数学语言描述实际现象的过程，即建立数学模型的过程。

数学模型是对实际问题的一种数学表述，是关于部分现实世界为某种目的的一个抽象的简化的数学结构。

问题求解第一步--数学建模?
(2)构造算法为什么需要数学建模

战德臣 教授

✓哥尼斯堡七桥问题：“寻找走遍这7座桥且只许走过每座桥一次最后又回到原出发点的路径”。

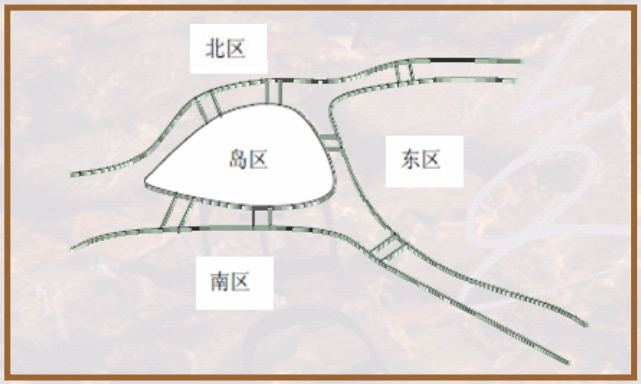
这个算法是怎样的呢?

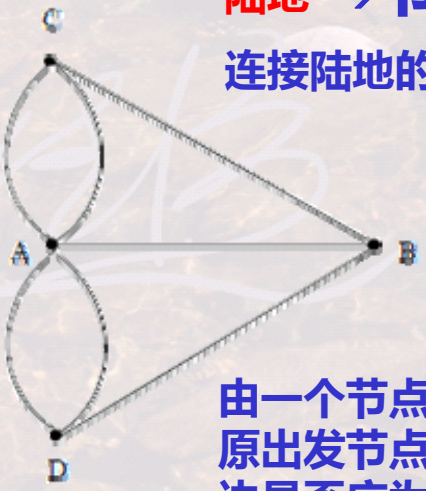
问题求解第一步--数学建模?
(2)构造算法为什么需要数学建模

战德臣 教授

哥尼斯堡七桥问题被抽象成一个“图(Graph)”

--由节点和边所构成的一种结构。





陆地--→节点

连接陆地的桥梁--→连接节点的边

由一个节点出发，又回到原出发节点—连接节点的边是否应为偶数

这个问题无解！无解的问题还用构造算法吗？

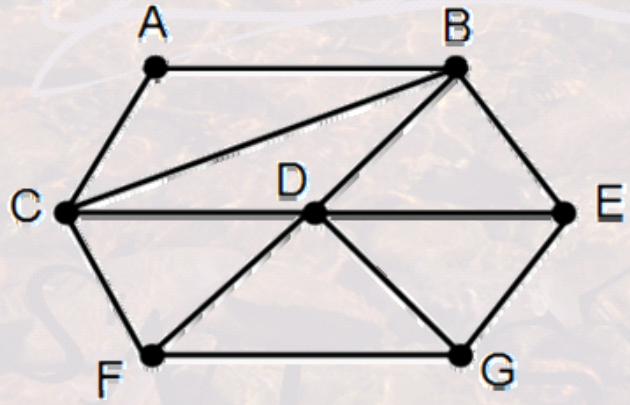
问题求解第一步--数学建模?
(2)构造算法为什么需要数学建模

战德臣 教授

✓**哥尼斯堡七桥问题**：

“寻找走遍这7座桥且只许走过每座桥一次最后又回到原出发点的路径”

✓**一般性问题**：“对给定的任意一个河道图与任意多座桥判定可能不可能每座桥恰好走过一次?”。



如能抽象成数学模型，则可将一个具体问题的求解，推广为一类问题的求解！

问题求解第一步--数学建模?
(2)构造算法为什么需要数学建模

战德臣 教授

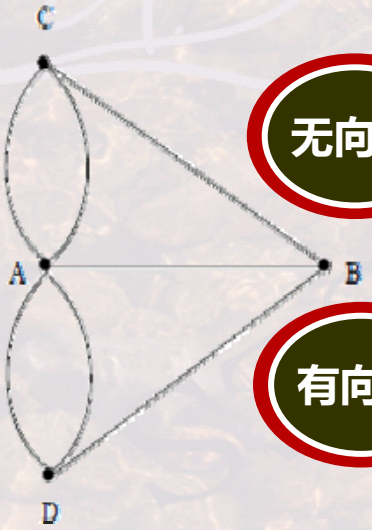
依据“图”，可发现该问题所蕴含的许多性质

- “连通----两个节点间有路径相连接”
- “可达----从一个节点出发能够到达另一个节点”
- “回路---从一个节点出发最后又回到该节点的一条路径”

“图论”专门讲这方面的内容哟！

经过图中每边一次且仅一次的回路

欧拉回路-欧拉图



无向图

节点的“度”

偶度节点

奇度节点

有向图

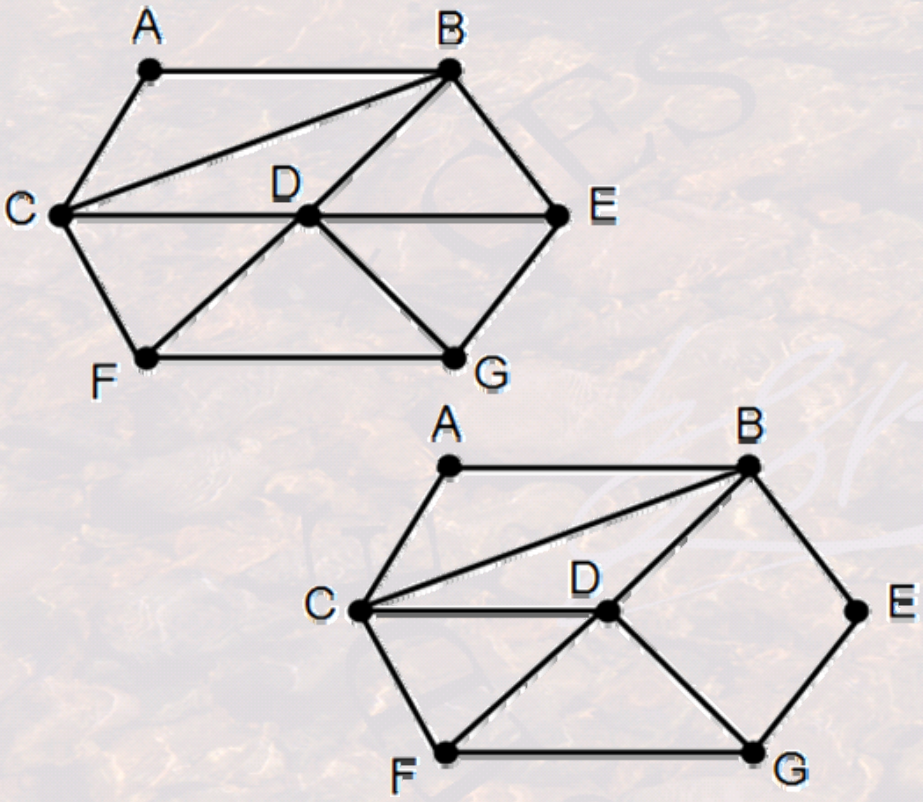
入度/出度

哈密尔顿回路-哈密尔顿图

经过图中每节点一次且仅一次的回路

问题求解第一步--数学建模?
(2)构造算法为什么需要数学建模

战德臣 教授



“一笔画”

- 凡是由偶度点组成的连通图，一定可以一笔画成。画时可以把任一偶度点为起点，最后一定能以这个点为终点画完此图；
- 凡是只有两个奇度点的连通图，其余都为偶度点，一定可以一笔画成。画时必须以一个奇度点为起点，另一个奇度点为终点。
- 其余情况都不能一笔画出。

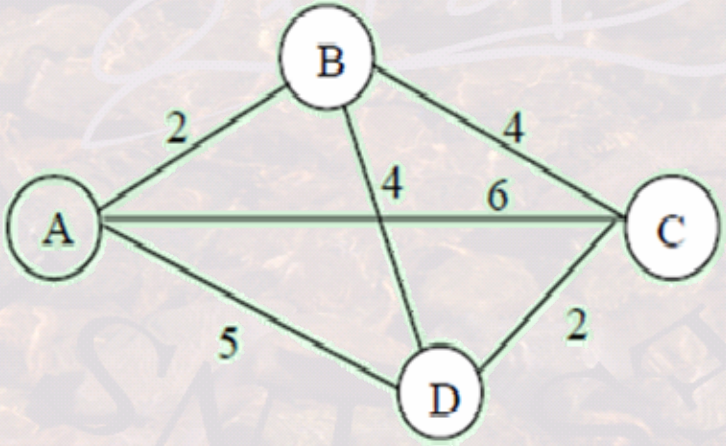
问题求解第一步--数学建模?
(3)TSP问题及其数学建模

战德臣 教授

◆**TSP问题(Traveling Salesman Problem，旅行商问题)**，威廉哈密尔顿爵士和英国数学家克克曼T.P.Kirkman于19世纪初提出TSP问题.

◆**TSP问题**：有若干城市，任何两个城市之间的距离都是确定的，现要求一旅行商从某城市出发必须经过每一城市且只能在每个城市逗留一次，最后回到原出发城市，问如何事先确定好一条最短的路线使其旅行的费用最少。

城市之间的距离



经过图中每节点一次且仅一次的回路

哈密尔顿回路-哈密尔顿图

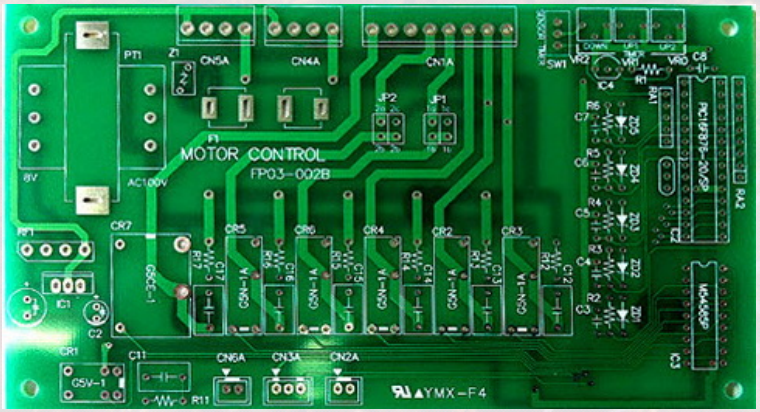
经过图中每节点一次且仅一次的回路；连接节点的边有“权值”

赋权-哈密尔顿图

问题求解第一步--数学建模?
(3)TSP问题及其数学建模

战德臣 教授

◆TSP是最有代表性的组合优化问题之一，在半导体制造(线路规划)、物流运输(路径规划)等行业有着广泛的应用。



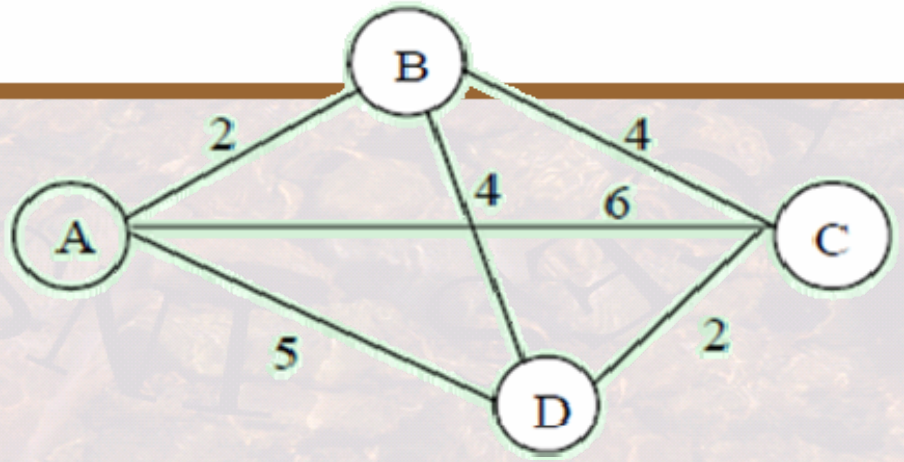
战德臣 教授

TSP问题的数学模型

输入: n个城市，记为$V=\{v_1,v_2,\ldots,v_n\}$，任意两个城市$v_i,v_j \in V$之间有距离$d_{v_iv_j}$

输出: 所有城市的一个访问顺序$T=\{t_1,t_2,\ldots,t_n\}$，其中$t_i \in V, t_{n+1}=t_1$，使得$\min \sum_{i=1}^{n} d_{t_it_{i+1}}$

问题求解的基本思想: 在所有可能的访问顺序T构成的状态空间Ω上搜索使得$\sum_{i=1}^{n} d_{t_it_{i+1}}$最小的访问顺序$T_{opt}$。



如能抽象成数学模型，则可从数学模型中发现问题求解的思路！

问题求解第一步--数学建模?
(3)TSP问题及其数学建模

战德臣 教授

TSP问题怎样求解呢?

算法策略设计---算法思想

战德臣

哈尔滨工业大学　教授.博士生导师

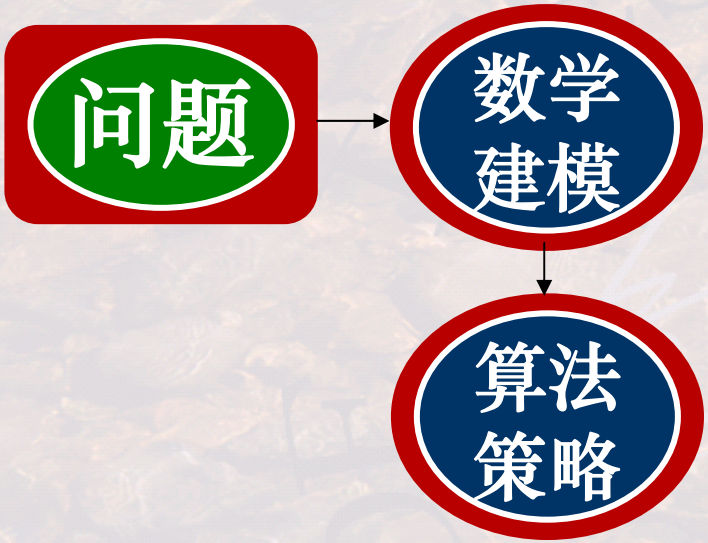
教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

算法策略设计—算法思想?

(1)问题求解可有不同的策略

战德臣 教授



问题的求解

算法策略是指求解一个问题的思想或说方法。同一问题可以采用不同的算法策略进行求解，而同一算法策略也可应用于不同问题的求解。

算法策略设计—算法思想?
(2)最容易想到的--“遍历”

战德臣 教授

问题求解的基本思想：在所有可能的访问顺序 T 构成的状态空间Ω上搜索使得$\sum_{i=1}^{n} d_{t_i t_{i+1}}$ 最小的访问顺序 T_{opt}。

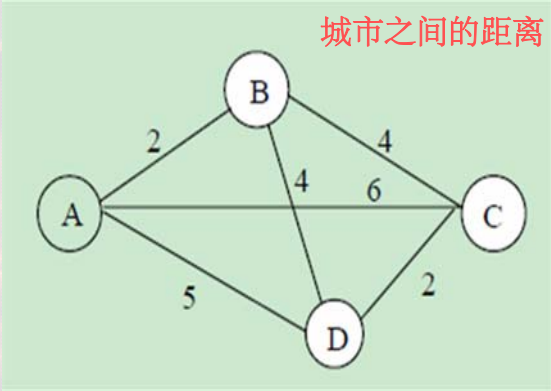
◆**遍历算法**：列出每一条可供选择的路线，计算出每条路线的总里程，最后从中选出一条最短的路线。

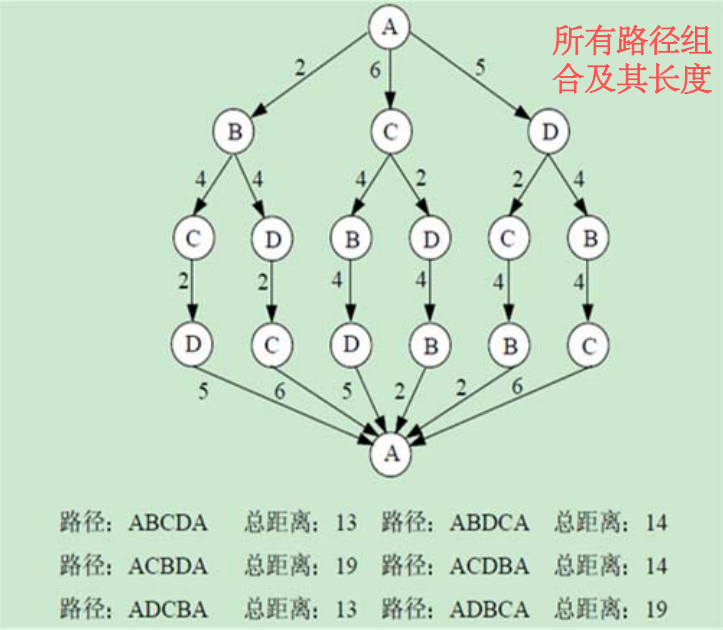
◆出现的问题：组合爆炸

✓路径组合数目：(n-1)!

✓20个城市，遍历总数1.216×10^{17}

✓计算机以每秒检索1000万条路线的计算速度，需386年。





算法策略设计—算法思想?
(2)最容易想到的--“遍历”

战德臣 教授

◆TSP**问题的难解性**：随着城市数量的上升，TSP问题的“遍历”方法计算量剧增，计算资源将难以承受。

◆2001年解决了德国**15112**个城市的TSP问题，使用了美国Rice大学和普林斯顿大学之间互连的、速度为500MHz 的Compaq EV6 Alpha 处理器组成的**110台计算机**，所有计算机花费的时间之和为**22.6年**。

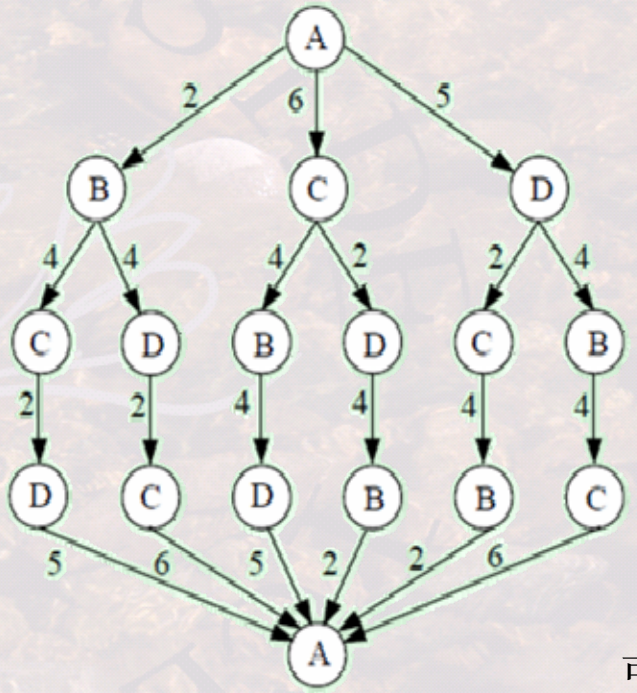
An optimal TSP tour through Germany's 15 largest cities. It is the shortest among 43 589 145 600 possible tours visiting each city exactly once.

战德臣 教授

TSP问题，有没有其他求解算法呢？

- ◆最优解 vs. 可行解
- ◆不同的算法设计策略：
  - ✓ 遍历、搜索算法
  - ✓ 分治算法
  - ✓ 贪心算法
  - ✓ 动态规划算法
  - ✓ 遗传算法
  - ✓ ……

“算法设计与分析”专门讲这方面的内容哟！



可行解

最优解

路径：ABCDA 总距离：13　路径：ABDCA 总距离：14

路径：ACBDA 总距离：19　路径：ACDBA 总距离：14

路径：ADCBA 总距离：13　路径：ADBCA 总距离：19

TSP问题的可行解与最优解示意

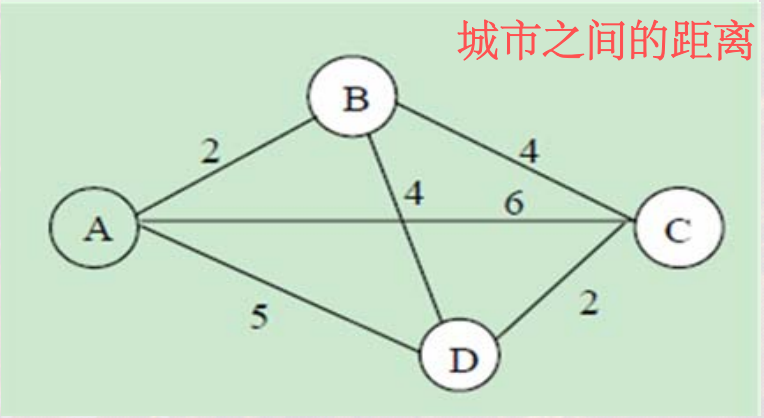
战德臣 教授

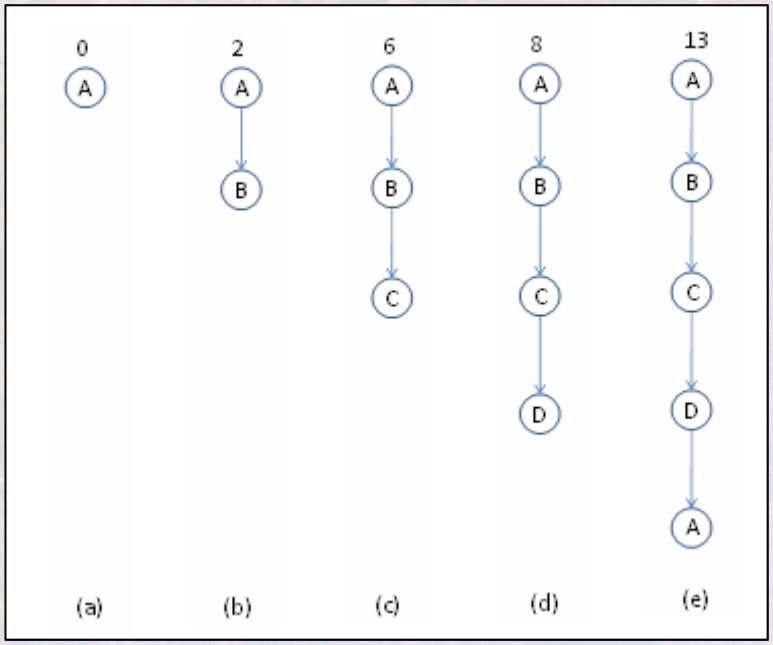
贪心算法是一种算法策略，或者说问题求解的策略。基本思想“今朝有酒今朝醉”。

✓一定要做当前情况下的最好选择，否则将来可能会后悔，故名“贪心”。

✓从某一个城市开始，每次选择一个城市，直到所有城市都被走完。

✓每次在选择下一个城市的时候，只考虑当前情况，保证迄今为止经过的路径总距离最短。





算法策略设计---算法思想
(4)最常用的--“贪心”?

战德臣 教授

怎样把贪心策略变成算法呢?

算法思想的精确表达
--算法的数据结构设计

战德臣

哈尔滨工业大学　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

算法思想的精确表达--算法的数据结构设计

(1)算法思想需要表达

战德臣 教授

■**算法的数据结构设计**---问题或算法相关的数据之间的逻辑关系及存储关系的设计

如何将数学模型中的数据转为计算机可以存储和处理的数据?

■**算法的控制结构设计**---算法的计算规则或计算步骤设计

如何构造和表达处理的规则，以便能够按规则逐步计算出结果?

战德臣 教授

数据结构是数据的逻辑结构、存储结构及其操作的总称，它提供了问题求解/算法的数据操纵机制。

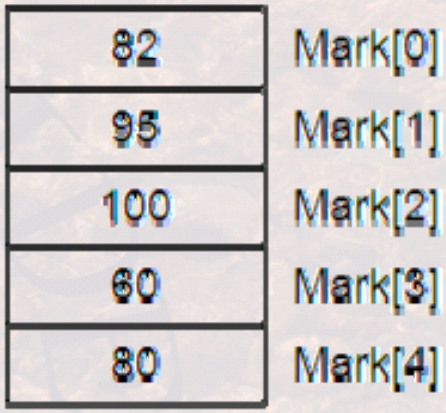
算法思想的精确表达--算法的数据结构设计
(3)简单的结构--多元素变量及其存储

战德臣 教授

◆向量或数组：有序数据的集合型变量，向量中的每一个元素都属于同一个数据类型，用一个统一的向量名和下标来唯一的确定向量中的元素。

◆向量名通常表示该向量的起始存储地址，而向量下标表示所指向元素相对于起始存储地址的偏移位置。

向量实例



编写求上述数组中值的平均值的程序

```
n = 4;
Sum=0;
For  J =0 to n Step 1
{    Sum = Sum + mark[ J ];
        }
Next  J
Avg = Sum/(n+1);
```

用变量名和元素位置共同表示存储地址，即向量		存储地址	存储内容(即变量值)
Mark	[0]	00000000 00000000 00000000 00000001	(注：82 的 4 字节二进制数可通过赋值发生改变)
	[1]	00000000 00000010 00000000 00000011	(注：95 的 4 字节二进制数可通过赋值发生改变)
	[2]	00000000 00000100 00000000 00000101	(注：100 的 4 字节二进制数可通过赋值发生改变)
	[3]	00000000 00000110 00000000 00000111	(注：60 的 4 字节二进制数可通过赋值发生改变)
	[4]	00000000 00001000 00000000 00001001	(注：80 的 4 字节二进制数可通过赋值发生改变)

向量存储实例

多元素变量使得程序可通过下标来操作多元素变量中的每一个元素

算法思想的精确表达--算法的数据结构设计
(3)简单的结构--多元素变量及其存储

ICES 战德臣 教授

◆矩阵或表：按行按列组织数据的集合型变量，二维向量，可表示为如M[2,3]或M[2][3]形式，即用符号名加两个下标来唯一确定表中的一个元素，前一下标为行序号，后一下标为列序号。(注意：有些语言下标从0开始，而有些语言下标则从1开始。)


```
Sum=0;
For  I =1 to 4 Step 1
{   For  J =1 to 4 Step 1
    { Sum = Sum + M[I][J];  }
    Next  J
}
Next  I
Avg = Sum/16;
```

逻辑上是二维的按行、列下标来操作一个元素，如M[2,3]或M[2][3]；物理上仍旧是一维存储的:

“元素[i][j]的存储地址= 表起始地址+ ((行下标i-1) * 列数 + (列下标j-1))*一个元素的存储单元数目”

这种转换可由系统自动完成，程序中只需按下标操作即可，即如M[2][3]

算法思想的精确表达--算法的数据结构设计
(4)TSP问题应该设计哪些数据结构?

战德臣 教授

◆数据结构设计就是针对选定的算法策略，设计其相应的数据结构及其运算规则。不同的算法可能有不同的数据结构及其运算规则！

城市映射为编号: A---1, B---2, C---3, D---4

城市间距离关系: 表或二维数组D，用D[i][j]或D[i,j]来确定欲处理的每一个元素

城市编号	1	2	3	4
1		2	6	5
2	2		4	4
3	6	4		2
4	5	4	2	

D

D[2][3]

访问路径/解: 一维数组S，用S[j]来确定每一个元素

S	1	4	3	2
	S[1]	S[2]	S[3]	S[4]

{A->D->C->B->A}

算法思想的精确表达
--算法的控制结构设计

战德臣

哈尔滨工业大学　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

算法思想的精确表达--算法的控制结构设计

(1)算法需要表达

ICES 战德臣 教授

■**算法的数据结构设计**---问题或算法相关的数据之间的逻辑关系及存储关系的设计

如何将数学模型中的数据转为计算机可以存储和处理的数据？

■**算法的控制结构设计**---算法的计算规则或计算步骤设计

如何构造和表达处理的规则，以便能够按规则逐步计算出结果?

ICES 战德臣 教授

基于类自然语言步骤描述法的算法表达

✓顺序结构：“**执行A，然后执行B**”，是按顺序执行一条条规则的一种结构 。

✓分支结构：“**如果Q成立，那么执行A，否则执行B**”，Q是某些逻辑条件，即按条件判断结果决定执行哪些规则的一种结构 。

✓循环结构：控制指令或规则的多次执行的一种结构---迭代(iteration)

◆循环结构又分为有界循环结构和条件循环结构。

✓有界循环：“**执行A指令N次**”，其中N是一个整数。

✓条件循环：某些时候称为无界循环，“**重复执行A直到条件Q成立**”或“**当Q成立时反复执行A**”，其中Q是条件。

基于类自然语言步骤描述法的算法表达

◆ 例如：sum=1+2+3+4+…+n求和问题的算法描述

Start of the algorithm(算法开始)

(1)输入N的值；

(2)设 i 的值为1；sum的值为0；

(3)如果 i<=N，则执行第(4)步，否则转到第(7)步执行；

(4)计算 sum + i，并将结果赋给sum；

(5)计算 i+1，并将结果赋给i;

(6)返回到第3步继续执行；

(7)输出sum的结果。

End of the algorithm(算法结束)

类自然语言表示的算法容易出现二义性、不确定性等问题

战德臣 教授

基于程序流程图的算法表达

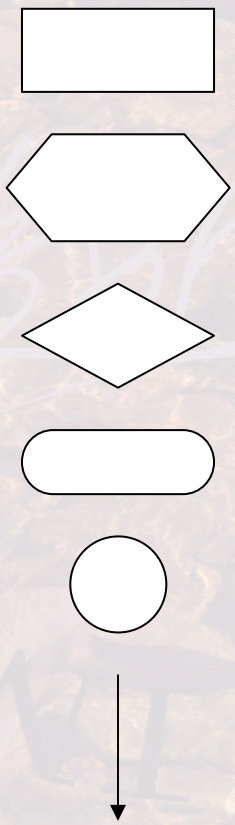
流程图的基本符号

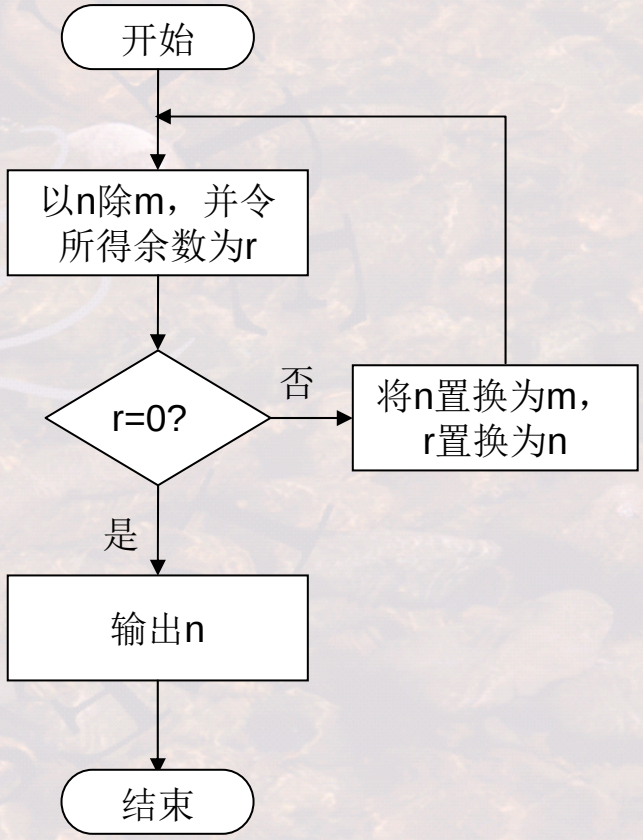
✓矩形框：表示一组顺序执行的规则或者程序语句。

✓菱形框：表示条件判断，并根据判断结果执行不同的分支。

✓圆形框：表示算法或程序的开始或结束。

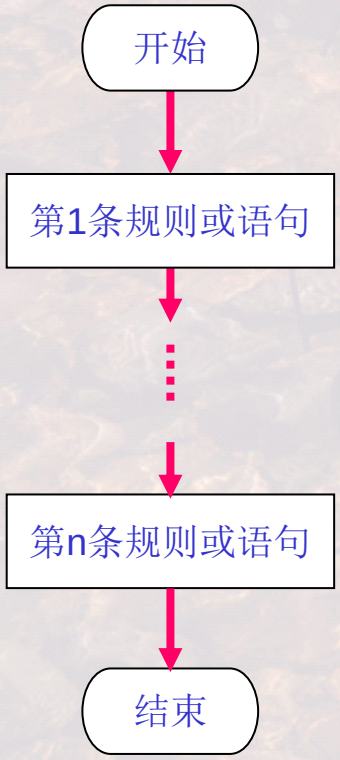
✓箭头线：表示算法或程序的走向。





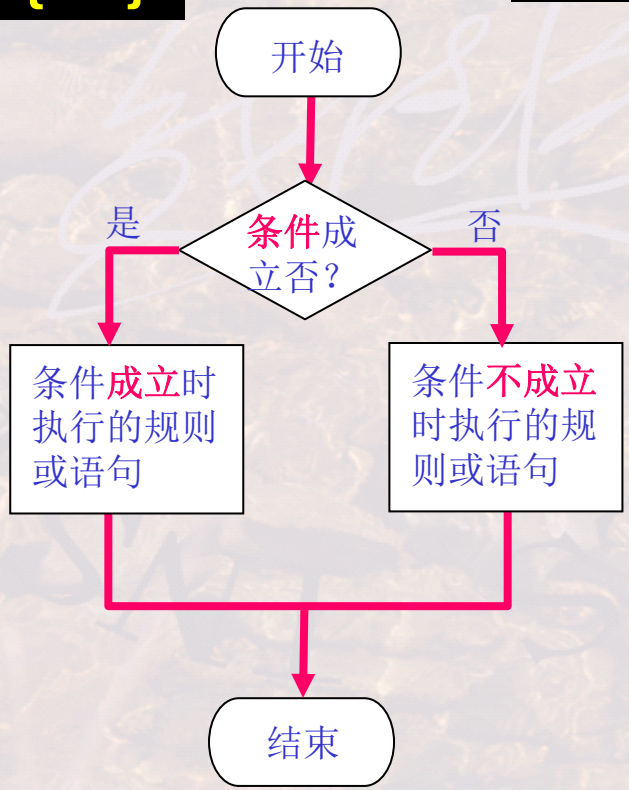
战德臣 教授

基于程序流程图的算法表达



顺序结构的流程图

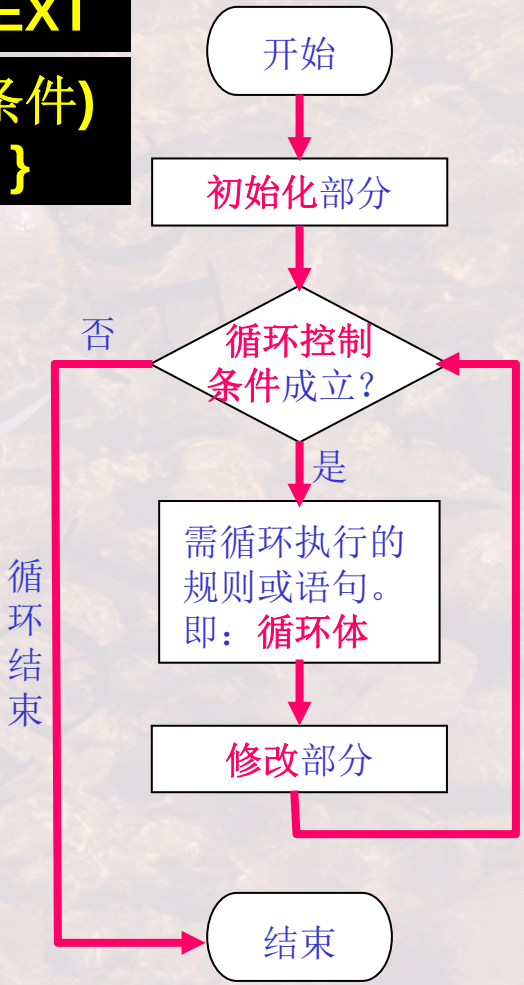
IF (条件)
Then { … }
Else { … }



分支结构的流程图

FOR (条件)
{ … } NEXT

While(条件)
DO { … }



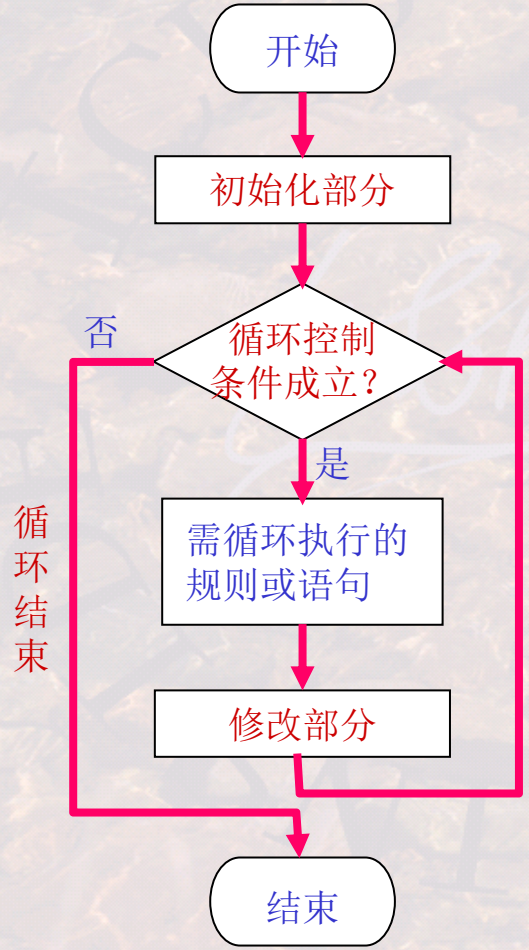
循环结构的流程图

战德臣 教授

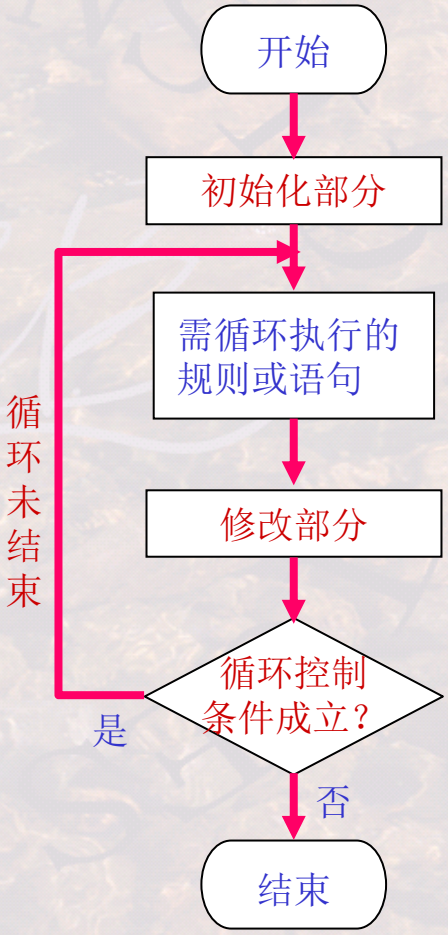
基于程序流程图的算法表达

FOR (条件)
{ … } NEXT

While(条件)
DO { … }



“先判断，后执行”
的循环结构



DO { … }
While(条件)

“先执行，后判断”
的循环结构

算法思想的精确表达--算法的控制结构设计
(3)具体算法的表达

战德臣 教授

求解TSP问题的遍历算法

列出每一条可供选择的路线，计算出每条路线的总里程，最后从中选出一条最短的路线。

将思想/策略转变为“步骤”

S

S[1]	S[2]	S[3]	S[4]
1	2	3	4
1	2	4	3
1	3	2	4
1	3	4	2

.....

Distance =
D[S[4]][S[1]]+
∑ D[S[i]][S[i+1]]
i=1 to 3

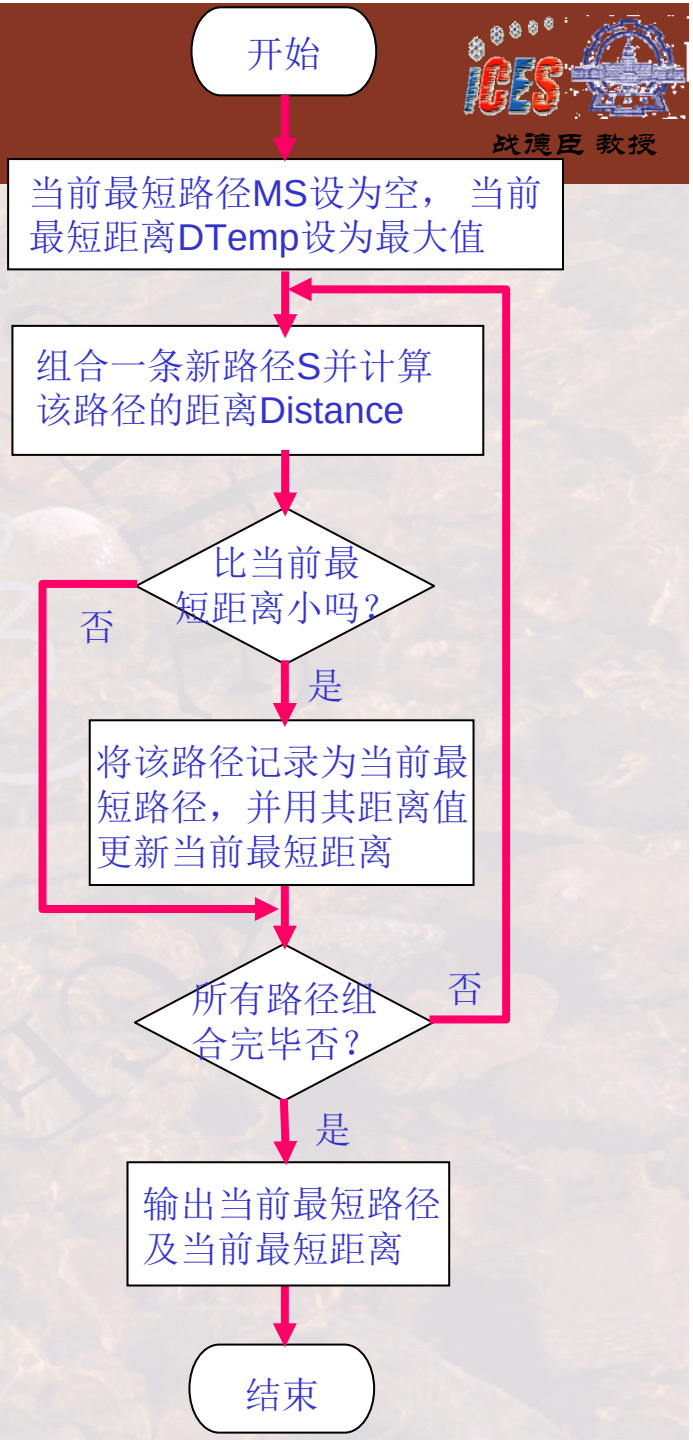
当前最短路径 MS

1	2	3	4

当前最短距离 DTemp

D

城市编号	1	2	3	4
1		2	6	5
2	2		4	4
3	6	4		2
4	5	4	2	



战德臣 教授

求解TSP问题的贪心算法

✓依次访问过的城市编号被记录在S[1], S[2], …, S[N}中,即第i 次访问的城市记录在S[i]中。

✓Step(1)：从第1个城市开始访问起，将城市编号1赋值给S[1]。

✓Step(6)：第I次访问的城市为城市j，其距第I-1次访问城市的距离最短。

S

S[1]	S[2]	S[3]	S[4]
1			

I = 2, 3, 4 ---当前要找第几个

j = 1, 2, 3, 4 ---城市号

D

城市编号	1	2	3	4
1		2	6	5
2	2		4	4
3	6	4		2
4	5	4	2	

Start of the Algorithm

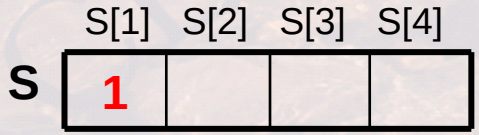
(1) S[1]=1;
(2) Sum=0;
(3) 初始化距离数组D[N, N];
(4) I=2;
(5) 从所有未访问过的城市中查找距离S[I-1]最近的城市j;
(6) S[I]=j;
(7) I=I+1;
(8) Sum=Sum+Dtemp;
(9) 如果I<=N，转步骤(5)，否则，转步骤(10);
(10)Sum=Sum+D[1, j];
(11)逐个输出S[N]中的全部元素;
(12)输出Sum。

End of the Algorithm

战德臣 教授

求解TSP问题的贪心算法(Cont.)

◆前述第5步“从所有未访问过的城市中查找距离S[I-1]最近的城市j”还是不够明确，需要进一步细化



I = 2, 3, 4 ---当前要找第几个

L = 1, 2, …,I-1 ---从第1个访问过的，到第I-1个访问过的

K = 2, 3, 4 ---城市号

(5.1)K=2;
(5.2)将Dtemp设为一个大数(比所有两个城市之间的距离都大)
(5.3)L=1；
(5.4)如果S[L]==K，转步骤5.8；//该城市已出现过，跳过
(5.5)L=L+1；
(5.6)如果L<I，转5.4；
(5.7)如果D[K,S[I-1]]<Dtemp,则j=K; Dtemp＝D[K,S[I-1]]；
(5.8)K=K+1;
(5.9)如果K<=N，转步骤5.3。

战德臣 教授

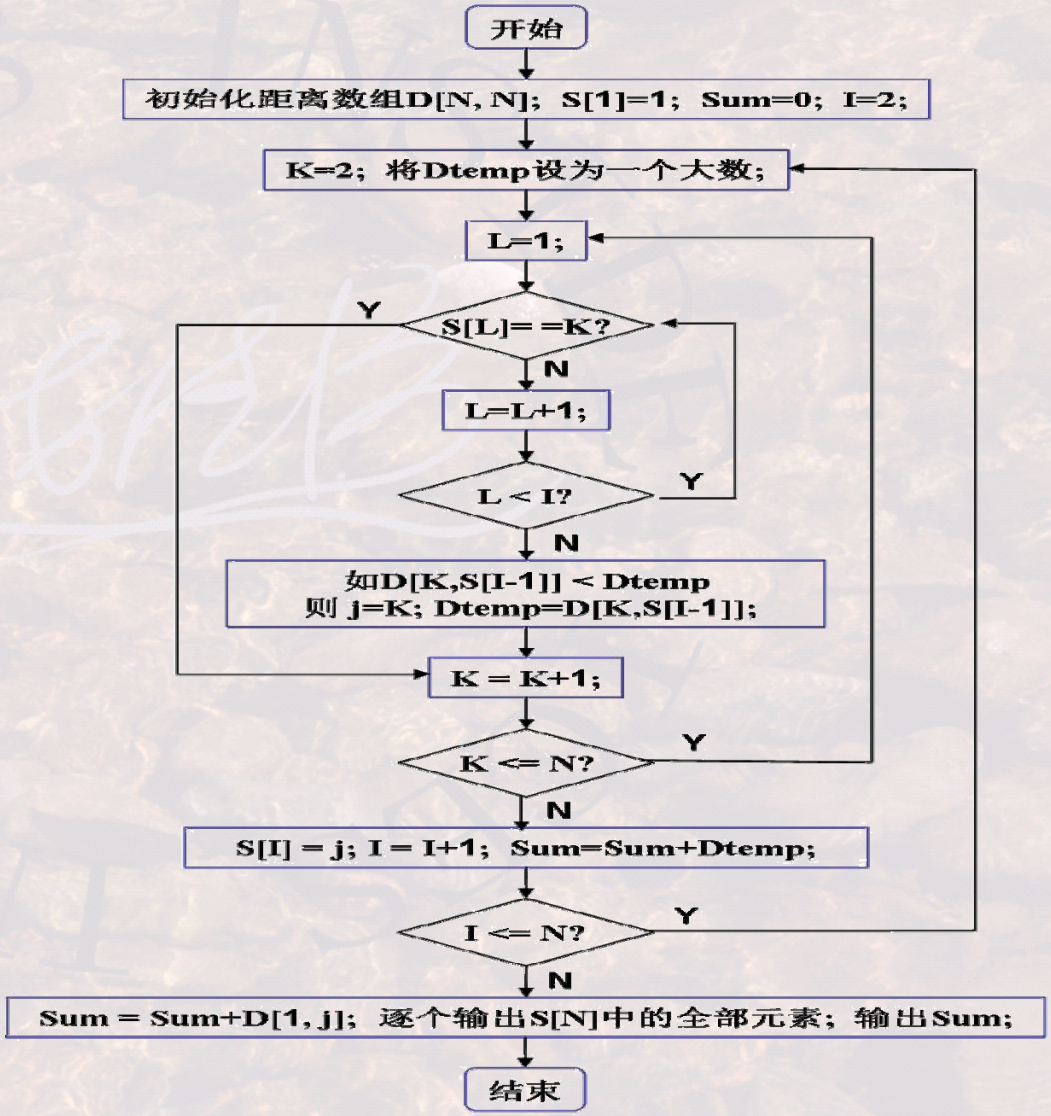
求解TSP问题的贪心算法

	S[1]	S[2]	S[3]	S[4]
S	1			

I = 2, 3, 4 ---当前要找第几个

K = 2, 3, 4 ---城市号

L = 1, 2, …, I-1 ---从第1个访问过的，到第I-1个访问过的



战德臣 教授

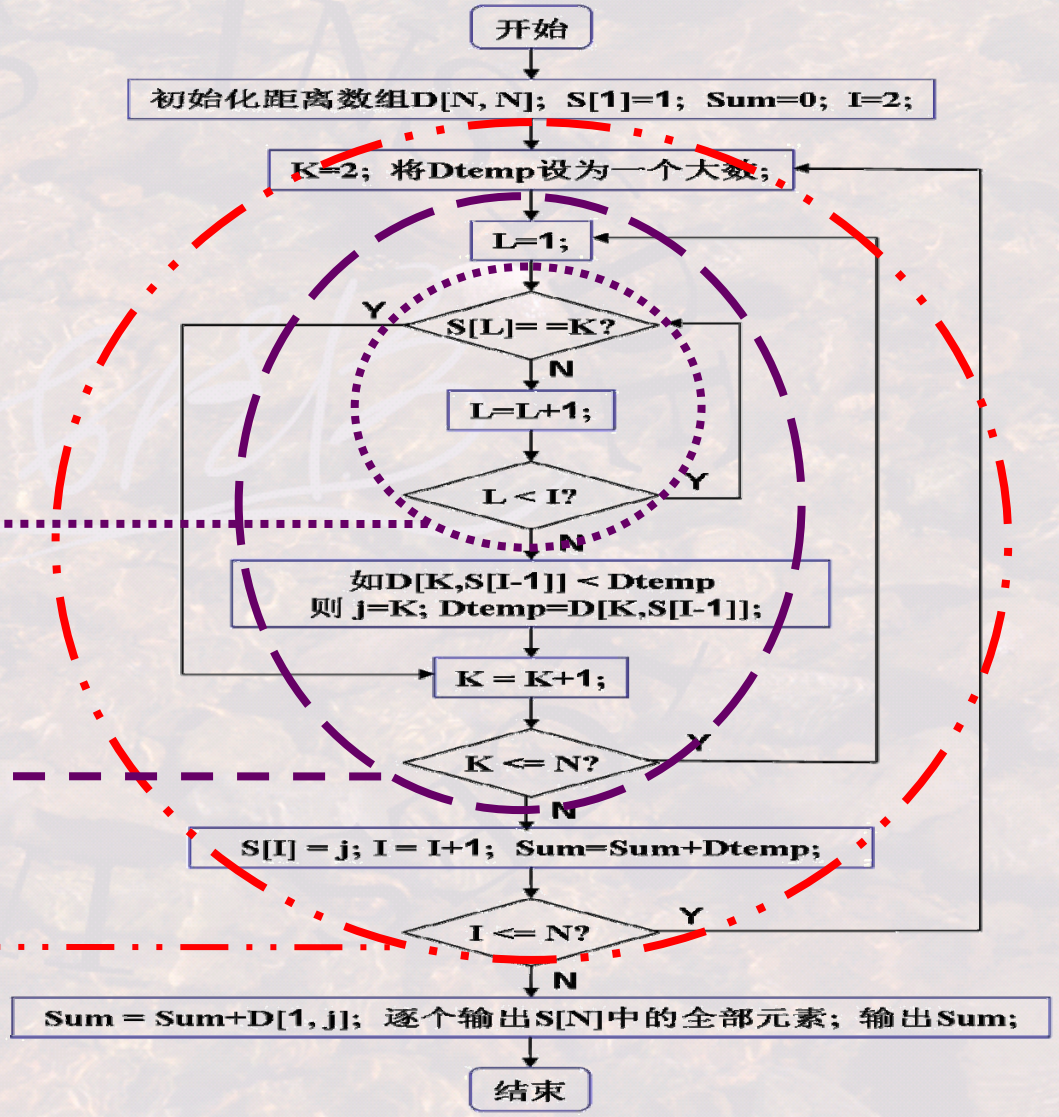
算法思想解读

◆软件工程学科的学生不仅能够设计算法，而且会描述和表达算法，更要能读懂算法。

内层循环，L从1至I-1，循环判断第K个城市是否是已访问过的城市，如是则不参加最小距离的比较；

中层循环，K从第2个城市至第N个城市循环，判断D[K, S[I-1]]是否是最小值，j记录了最小距离的城市号K。

外层循环，I从2至N循环；I-1个城市已访问过，正在找与第I-1个城市最近距离的城市；已访问过的城市号存储在S[]中。



算法的实现---程序设计

战德臣

哈尔滨工业大学　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

算法的实现-程序设计

(1)算法实现要选定计算机语言，你知道吗?

战德臣 教授

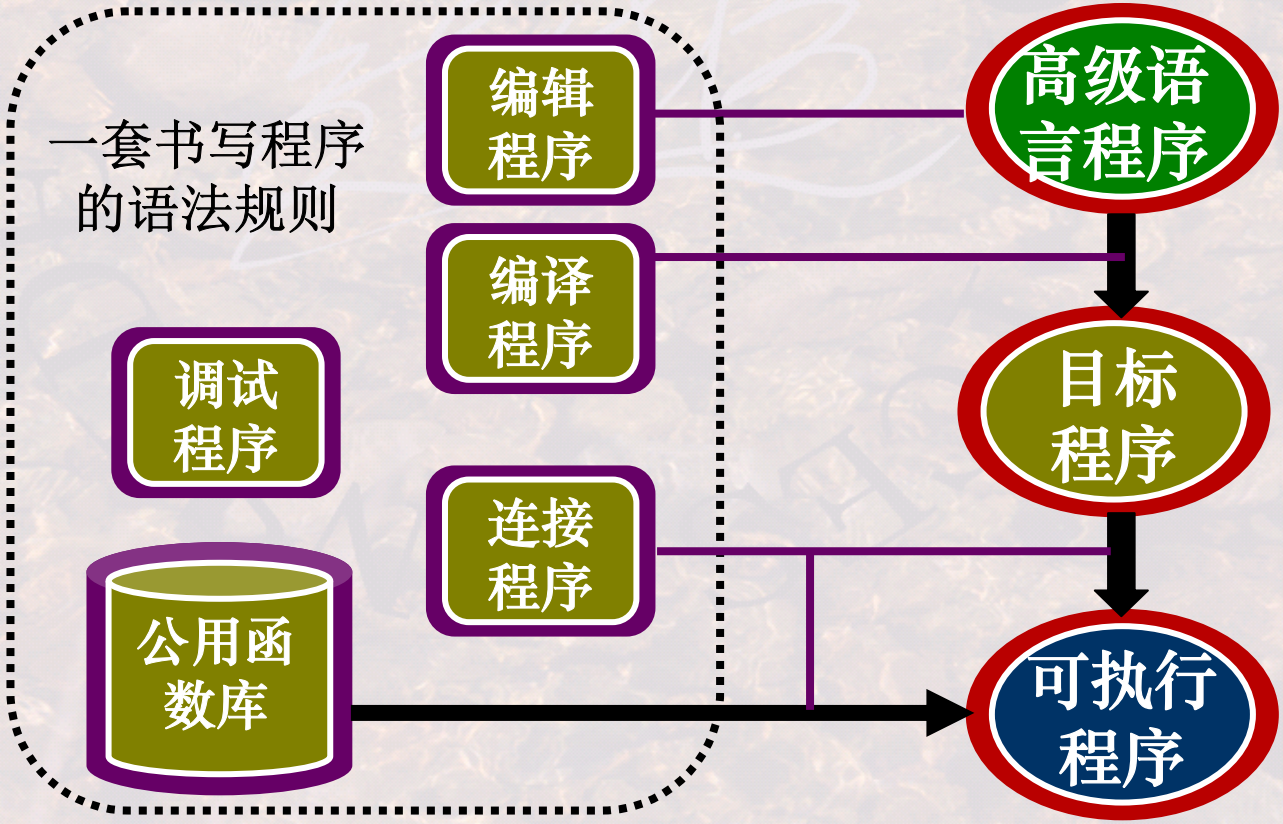
算法的实现-程序设计
(1)算法实现要选定计算机语言，你知道吗?

战德臣 教授

◆程序是算法的一种机器相容(Compatible)的表示，是利用计算机程序设计语言对算法描述的结果，是可以在计算机上执行的算法。

◆程序设计过程：编辑源程序➔编译➔链接➔执行。

计算机语言程序设计环境：编辑、编译、连接、调试、运行一体化平台

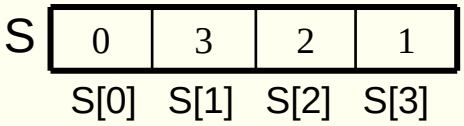


战德臣 教授

判断城市K是否是已访问过的城市

```
main() {
…
//前面已为 K 和 I 赋过值
L=0;    Found=0;
do {
        if(S[L]==K)
        { Found=1; break; }
        else L=L+1;
    } while(L<I);        //L从0到I-1循环
    if Found==0 { //城市K未出现过 }
    else { //城市K出现过 }
…
}
```

- K从1,…,n-1是城市的编号
- I是至今已访问过的城市数
- S[0]至S[I-1]中存储的是已访问过的城市的编号

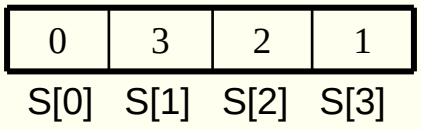


寻找下一个未访问过的城市j,且其距当前访问城市S[I-1]距离最短

```
main() {
…
    K=1;   Dtemp=10000;
    do{  // K从1到n-1循环
         L=0;   Found=0;
         do{
              if(S[L]==K)
              {   Found=1;   break;   }
             else L=L+1;
          } while(L<I);     //L从1到I循环
         if (Found==0 && D[K][S[I-1]]<Dtemp)
         {   j=K;   Dtemp=D[K][S[I-1]];   }
         K=K+1;
    } while(K<n);                          //K从1到n-1循环
    S[I]=j; Sum=Sum+Dtemp;
…
}
```

- K从1,…,n-1是城市的编号
- I是至今已访问过的城市数
- S[I-1]中存储的是当前访问过的城市编号，要找第I个

程序



- D[K][S[I-1]]为城市K距当前访问过的城市的距离

战德臣 教授

算法的实现

TSP 问 题 贪 心 算法程序实例

```
#include <stdio.h>
#define  n  4
main(){
  int   D[n][n], S[n], Sum, I, j, K, L, Dtemp, Found;
  S[0]=0;  Sum=0;   D[0][1]=2;  D[0][2]=6;  D[0][3]=5;  D[1][0]=2;  D[1][2]=4;
  D[1][3]=4;  D[2][0]=6;  D[2][1]=4;  D[2][3]=2;  D[3][0]=5;  D[3][1]=4;  D[3][2]=2;
  I=1;                    //注意程序是从 0 开始，流程图是从 1 开始的，下同.
  do {                    //I 从 1 到 n-1 循环
      K=1;  Dtemp=10000;
      do{                 //K 从 1 到 n-1 循环
          L=0;  Found=0;
          do{             //L 从 0 到 I-1 循环
              if(S[L]==K)
              {  Found=1;  break;  }
              else L=L+1;
            } while(L<I);     //L 从 0 到 I-1 循环
          if(Found==0 && D[K][S[I-1]]<Dtemp)
          {  j=K;  Dtemp=D[K][S[I-1]];  }
          K=K+1;
      } while(K<n);         //K 从 1 到 n-1 循环
      S[I]=j;  I=I+1;  Sum=Sum+Dtemp;
    } while(I<n);       //I 从 1 到 n-1 循环
  Sum=Sum+D[j][0];
  for(j=0;j<n;j++){  printf("%d,",S[j]);  }  //输出城市编号序列
  printf("\n");                              //换行
  printf("Total Length:%d",Sum);             //输出总距离
}
```

算法的正确性与复杂性分析

战德臣

哈尔滨工业大学　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

算法的正确性与复杂性分析

(1)算法需要分析正确性与复杂性

战德臣 教授

ICES 战德臣 教授

算法是正确的吗?

◆算法的正确性问题:

- ✓ 问题求解的过程、方法——算法是正确的吗?算法的输出是问题的解吗?
- ✓ 20世纪60年代,美国一架发往金星的航天飞机由于控制程序出错而永久丢失在太空中

◆算法的效果评价:

- ✓ 算法的输出是最优解还是可行解?如果是可行解,与最优解的偏差多大?

◆两种评价方法:

- ✓ 证明方法:利用数学方法证明;
- ✓ 仿真分析方法:产生或选取大量的、具有代表性的问题实例,利用该算法对这些问题实例进行求解,并对算法产生的结果进行统计分析。

算法获得的解是最优的吗?

算法的正确性与复杂性分析
(2)算法是正确的吗?

战德臣 教授

◆TSP问题贪心算法的正确性评价：

✓直观上只需检查算法的输出结果中，每个城市出现且仅出现一次，该结果即是TSP问题的可行解，说明算法正确地求解了这些问题实例

	S[1]	S[2]	S[3]	S[4]
S	1	2	3	4

✓TSP问题贪心算法的效果评价：

✓如果实例的最优解已知（问题规模小或问题已被成功求解），利用统计方法对若干问题实例的算法结果与最优解进行对比分析，即可对其进行效果评价；

✓对于较大规模的问题实例，其最优解往往是未知的，因此，算法的效果评价只能借助于与前人算法结果的比较。

```
#include <stdio.h>
#define  n  4
main(){
  int   D[n][n], S[n], Sum, I, j, K, L, Dtemp, Found;
  S[0]=0;  Sum=0;   D[0][1]=2;  D[0][2]=6;  D[0][3]=5;  D[1][0]=2;  D[1][2]=4;
  D[1][3]=4;  D[2][0]=6;  D[2][1]=4;  D[2][3]=2;  D[3][0]=5;  D[3][1]=4;  D[3][2]=2;
  I=1;                    //注意程序是从 0 开始，流程图是从 1 开始的，下同.
  do {                    //I 从 1 到 n-1 循环
     K=1;  Dtemp=10000;
     do{                        //K 从 1 到 n-1 循环
        L=0;  Found=0;
        do{                     //L 从 0 到 I-1 循环
           if(S[L]==K)
           {  Found=1;  break;  }
           else L=L+1;
        } while(L<I);           //L 从 0 到 I-1 循环
        if(Found==0 && D[K][S[I-1]]<Dtemp)
        {  j=K;  Dtemp=D[K][S[I-1]];  }
        K=K+1;
     } while(K<n);              //K 从 1 到 n-1 循环
     S[I]=j;  I=I+1;  Sum=Sum+Dtemp;
  } while(I<n);                 //I 从 1 到 n-1 循环
  Sum=Sum+D[j][0];
  for(j=0;j<n;j++){  printf("%d,",S[j]);  }  //输出城市编号序列
  printf("\n");                             //换行
  printf("Total Length:%d",Sum);            //输出总距离
}
```

战德臣 教授

算法获得结果的时间有多长?

◆算法的复杂性分析

◆时间复杂性: 如果一个问题的规模是n，解这一问题的某一算法所需要的时间为T(n)，它是n的某一函数，T(n)称为这一算法的“时间复杂性”。

◆“大O记法”:

✓基本参数 n——问题实例的规模

✓把复杂性或运行时间表达为n的函数。

✓ “O”表示量级 (order)，允许使用“=”代替“≈”，如$n^2+n+1 = O(n^2)$ 。

◆空间复杂性：算法在执行过程中所占存储空间的大小。

算法复杂性分析示例

```
sum=0;                          （1次）
for( i=1; i<=n; i++)            （n次）
{  for( j=1; j<=n; j++)         （n²次）
        { sum++;  }               （n²次）
}
```

解：$T(n) = 2n^2+n+1 = O(n^2)$

主要关注点：循环的层数

算法的正确性与复杂性分析
(3)算法的计算时间有多长?

战德臣 教授

TSP问题贪心算法的复杂性

◆一个关于n的三重循环。

◆时间复杂度是$O(n^3)$。

```
#include <stdio.h>
#define  n  4
main(){
  int    D[n][n], S[n], Sum, I, j, K, L, Dtemp, Found;
  S[0]=0;  Sum=0;   D[0][1]=2;  D[0][2]=6;  D[0][3]=5;  D[1][0]=2;  D[1][2]=4;
  D[1][3]=4;  D[2][0]=6;  D[2][1]=4;  D[2][3]=2;  D[3][0]=5;  D[3][1]=4;  D[3][2]=2;
  I=1;                    //注意程序是从 0 开始，流程图是从 1 开始的，下同.
  do {                    //I 从 1 到 n-1 循环---将被执行 n-1 次，简记约 n 次
     K=1;  Dtemp=10000;
     do{                  //K 从 1 到 n-1 循环---将被执行(n-1)*(n-1)次，简记约 n²次
        L=0;  Found=0;
        do{               //L 从 0 到 I-1 循环---将被执行，简记约 n³次
           if(S[L]==K)
           {  Found=1;  break;  }
           else L=L+1;
        } while(L<I);     //L 从 0 到 I-1 循环
        if(Found==0 && D[K][S[I-1]]<Dtemp)
        {  j=K;  Dtemp=D[K][S[I-1]];  }
        K=K+1;
     } while(K<n);        //K 从 1 到 n-1 循环
     S[I]=j;  I=I+1;  Sum=Sum+Dtemp;
  } while(I<n);           //I 从 1 到 n-1 循环
  Sum=Sum+D[j][0];
  for(j=0;j<n;j++){  printf("%d,",S[j]);  } //输出城市编号序列
  printf("\n");                             //换行
  printf("Total Length:%d",Sum);            //输出总距离
}
```

n

n^2

n^3

战德臣 教授

$O(n^3)$与$O(3^n)$的差别，$O(n!)$与$O(n^3)$的差别

问题规模n	计算量
10	10!
20	20!
100	100!
1000	1000!
10000	10000!

$20! = 1.216 \times 10^{17}$

$20^3 = 8000$

$O(n^3)$	$O(3^n)$
0.2秒	4×10^{28}秒 =1015年

注：每秒百万次，n=60，1015年相当于10亿台计算机计算一百万年

$O(1)$, $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^b)$ | $O(b^n)$, $O(n!)$

战德臣 教授

◆当算法的时间复杂度的表示函数是一个多项式时，如$O(n^2)$时，则对于大规模问题，计算机是可以执行该算法的。

$O(1), O(\log n), O(n), O(n \log n), O(n^b)$

◆当算法的时间复杂度是用指数函数表示时，如$O(2^n)$或阶乘函数时，如$O(n!)$，当n很大（如10000）时计算机是无法执行该算法的。

$O(b^n), O(n!)$

计算复杂性与可求解及难求解问题

战德臣

哈尔滨工业大学　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

战德臣 教授

现实世界中的问题分类

- 计算机在有限时间内能够求解的(可求解问题)
- 计算机在有限时间内不能求解的(难求解问题)
- 计算机完全不能求解的(不可计算问题)

问题的计算复杂性

计算复杂性是指问题的一种特性，即利用计算机求解问题的难易性或难易程度，其衡量标准：

◆计算所需的步数或指令条数(即时间复杂度)

◆计算所需的存储空间大小(即空间复杂度)

----通常表达为关于问题规模n的一个函数 O(f(n))

战德臣 教授

TSP问题的计算复杂性

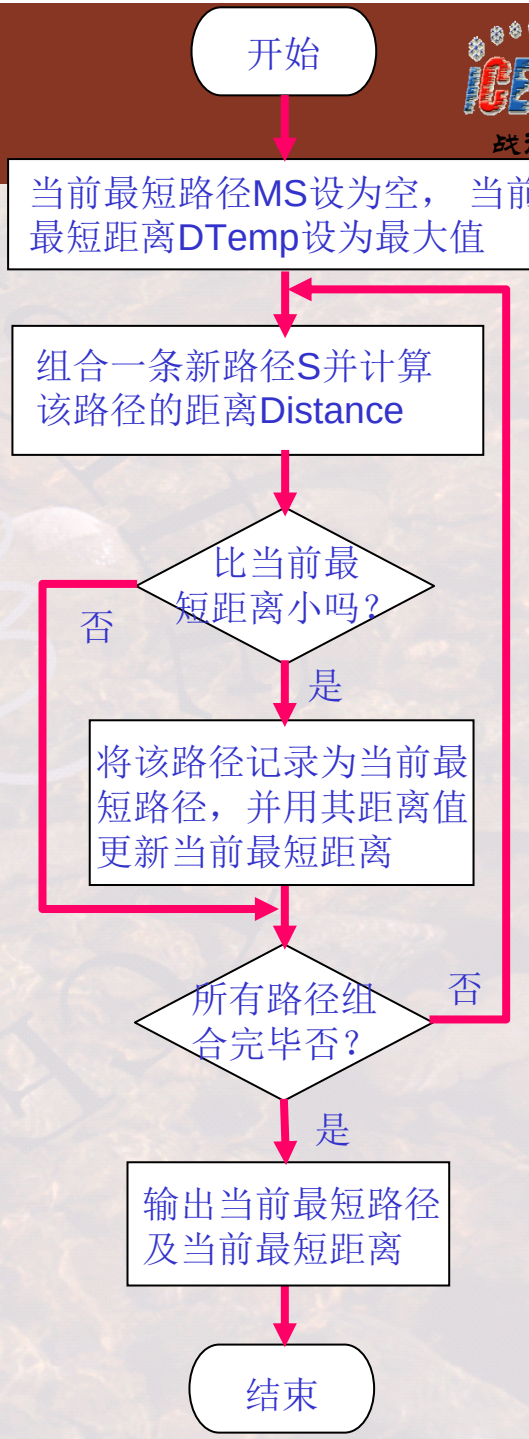
◆TSP问题的计算复杂性，即其遍历算法的复杂性：

✓列出每一条可供选择的路线，计算出每条路线的总里程，最后从中选出一条最短的路线。

✓路径组合数目为(n-1)!

✓时间复杂度是O((n-1)!)

✓精确解的求解只能是遍历



ICES
战德臣 教授

P类问题，NP类问题，NPC类问题

■**P类问题**：多项式问题(Polynomial Problem)，指计算机可以在有限时间内求解的问题，即：P类问题是可以找出一个呈现$O(n^a)$复杂性算法的问题，其中a为常数。

■**NP类问题**：非确定性多项式问题(Non-deterministic Polynomial)。有些问题，其答案是无法直接计算得到的，只能通过间接的猜算或试算来得到结果，这就是非确定性问题(Non-deterministic)。虽然在多项式时间内难于求解但不难判断给定一个解的正确性的问题，即：在多项式时间内可以由一个算法验证一个解是否正确的非确定性问题，就是NP类问题。

■**NPC问题**：完全非确定性多项式问题(NP-Complete)。如果NP问题的所有可能答案都可以在多项式时间内进行正确与否的验算的话就叫做完全非确定性多项式问题，即NP-Complete问题。

问：加密算法应该设计成一个什么问题呢？

计算复杂性与可求解及难求解问题
(2)问题的分类

战德臣 教授

可求解问题　难求解问题

算法复杂性　计算复杂性

可求解与难求解问题
(3)NPC类问题如何求解?

战德臣 教授

TSP问题的遍历算法

穷举法或称遍历法：对解空间中的每一个可能解进行验证，直到所有的解都被验证是否正确，便能得到精确的结果---精确解

可能是O(n!)或O(a^n)

TSP问题的贪心算法

近似解求解算法---近似解

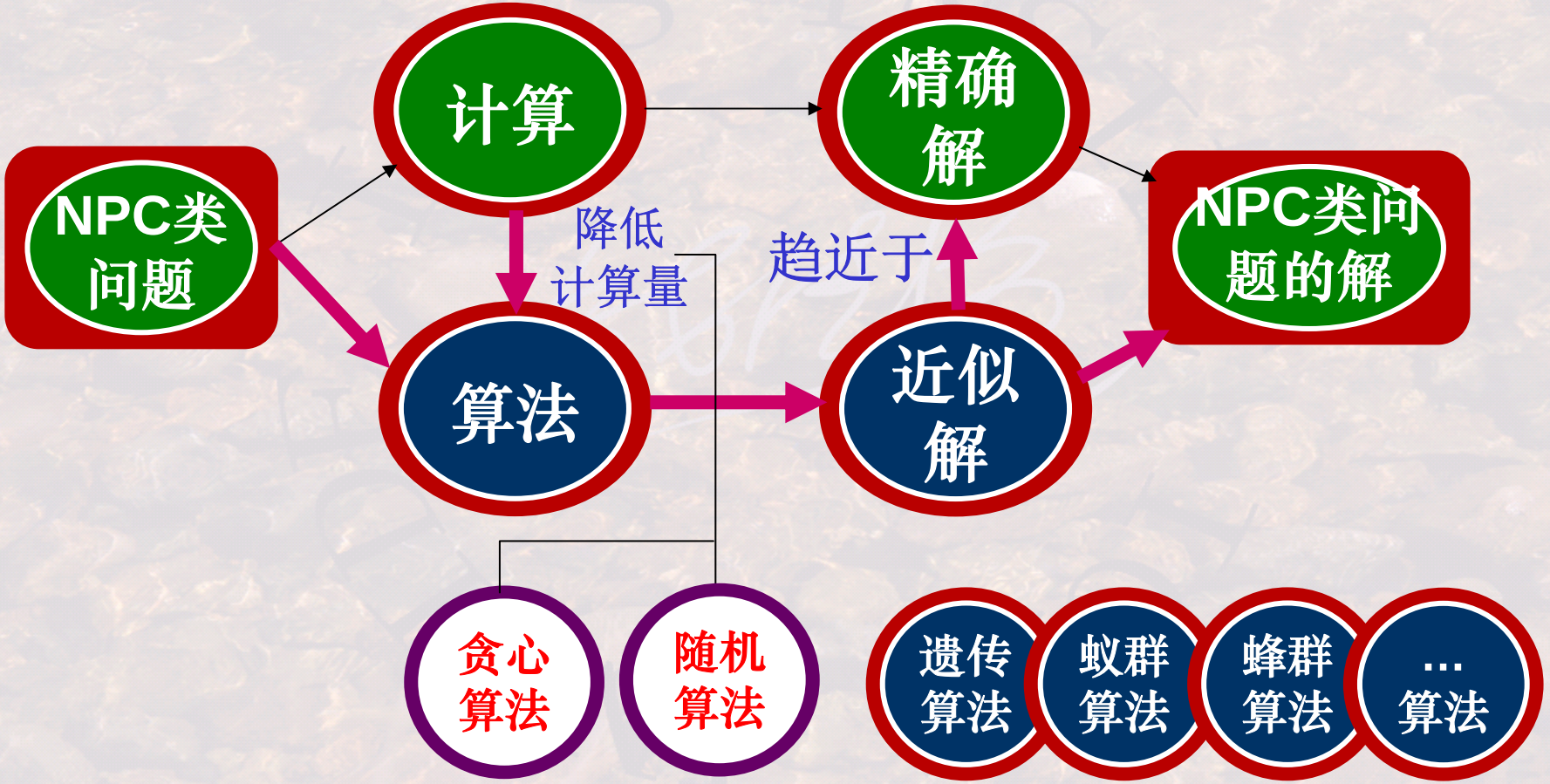
Δ = | 近似解 － 精确解 |
满意解： Δ充分小时的近似解

应该是O(n^a)

可求解与难求解问题

(3)NPC类问题如何求解?

战德臣 教授



第4讲 软件之灵魂--算法

战德臣

哈尔滨工业大学　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent
Computing for Enterprises & Services,
Harbin Institute of Technology

回顾本讲学习了什么

战德臣 教授