数据结构与问题求解

张铭 主讲

采用教材：张铭，王腾蛟，赵海燕 编写
高等教育出版社，2008. 6（“十一五”国家级规划教材）

http://www.jpk.pku.edu.cn/pkujpk/course/sjjg

程序设计与算法
—— Coursera专项课程

- https://www.coursera.org/specializations/biancheng-suanfa
- **张铭主讲:** “数据结构基础”、“高级数据结构与算法”

——北京大学信息科学技术学院教授，博士生导师，ACM Education Council 中国委员兼ACM中国教育专委会主席

- 教材：**《数据结构与算法》**。张铭，王腾蛟，赵海燕编，高等教育出版社，2008年 6月。——普通高等教育“十一五”国家级规划教材（入选“十二五”）

计算机教育研究

- 2001 – 2013，教育部计算机教育指导委员会委员
- **2016年至今，CCF教育工委会副主任**
- 2013年至今，ACM Ed Council, ACM 中国教育委员会主席 -> 2016 **ACM China SIGCSE 主席，ACM/IEEE CS2013, ACM/IEEE IT2017**

  https://www.sojump.hk/jq/13902520.aspx
- **2015-2018，MOOC中用户流失问题，自然科学基金面上项目主持人**
- 李晓明、陈平、**张铭**、朱敏悦。“关于计算机人才需求的调研报告”。《计算机教育》，2004年8月，PP11-18。 —— 他引 130次
- **Ming Zhang**, Virginia Mary Lo: Undergraduate computer science education in China. **SIGCSE 2010**, pp. 396-400. 被列为CMU “Images of Computing” 课程 Prof. Carol Frieze, http://www.cs.cmu.edu/~cfrieze/courses/ 参考文献，他引 19 次
- **Ming Zhang**, Long Zhang: Undergraduate IT education in China. ACM Inroads 5(3): 49-55 (2014)
- **Ming Zhang**, et.al. Educational Evaluation in the PKU SPOC Course "Data Structures and Algorithms", **L@S 2015**: 237-240

机器学习和文本挖掘研究

- 2012-2016 ，**海量社会媒体中不实信息的分析与检测**，国家自然科学基金，张铭为课题组负责人。
- 2017－2020 **面向全流程智慧健康管理决策的多源异构大数据融合方法研究**，国家自然科学基金重点项目，张铭为北大课题组负责人。
- 唐建（2009-2014），获得**Montreal大学Bengjio组的Assistant Professor教职**
- Jian Tang，Zhaoshi Meng，XuanLong Nguyen，Qiaozhu Mei，**Ming Zhang**，Understanding the Limiting Factors of Topic Modeling via Posterior Contraction Analysis，The 31st International Conference on Machine Learning (**ICML2014**)，**最佳论文奖**，2014.6.21-2014.6.26，EI，他引73
- Jian Tang，Jingzhou Liu，**Ming Zhang**，Qiaozhu Mei，Visualizing Large-scale and High-dimensional Data，**WWW 2016**. 04.11-2016.04.15，**最佳论文提名**，他引21
- Jian Tang, Meng Qu, Mingzhe Wang, **Ming Zhang**, Jun Yan, Qiaozhu Mei. LINE: Large-scale Information Network Embedding. WWW 2015, 1067-1077. EI，他引135

What most schools don't teach

http://v.youku.com/v_show/id_XNTIxNTk1OTI4.html

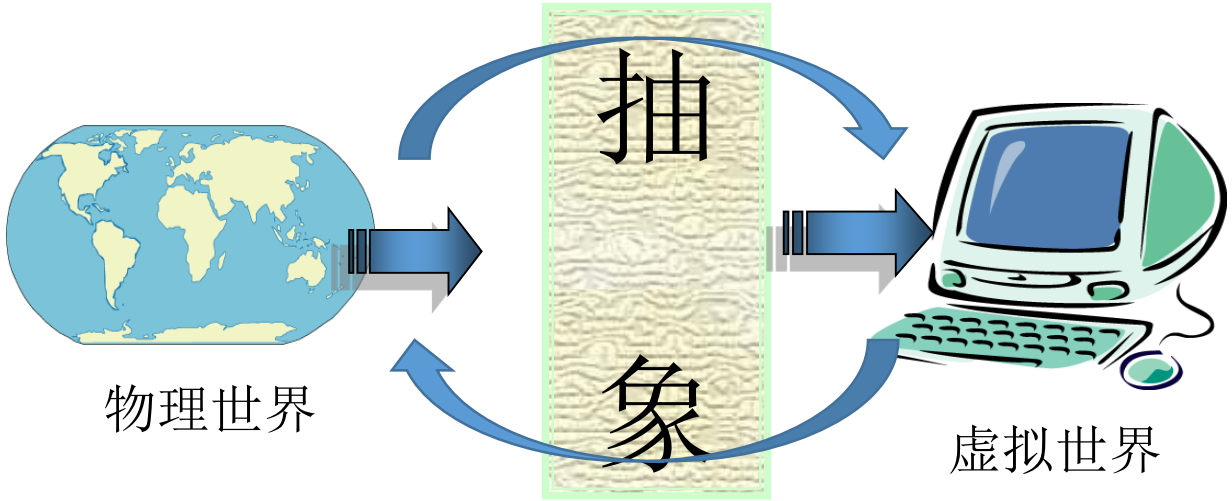
“计算作为一门学科”

- Denning, P.J. Comer, D.E. Gries, D. Mulder, M.C. Tucker, A. Turner, A.J. Young, P.R. , **Computing as a discipline**. Computer, Volume: 22, Issue: 2, Feb 1989. ACM, New York, NY. PP63-70.
  - **培养学生面向学科的思维能力，使学生领会学科的力量，以及从事本学科工作的价值之所在**
  - **希望能用类似于数学那样严密的方式将学生引入到计算学科各个富有挑战性的领域之中**

什么是计算学科

数学 —— 所有学科和上帝的关系

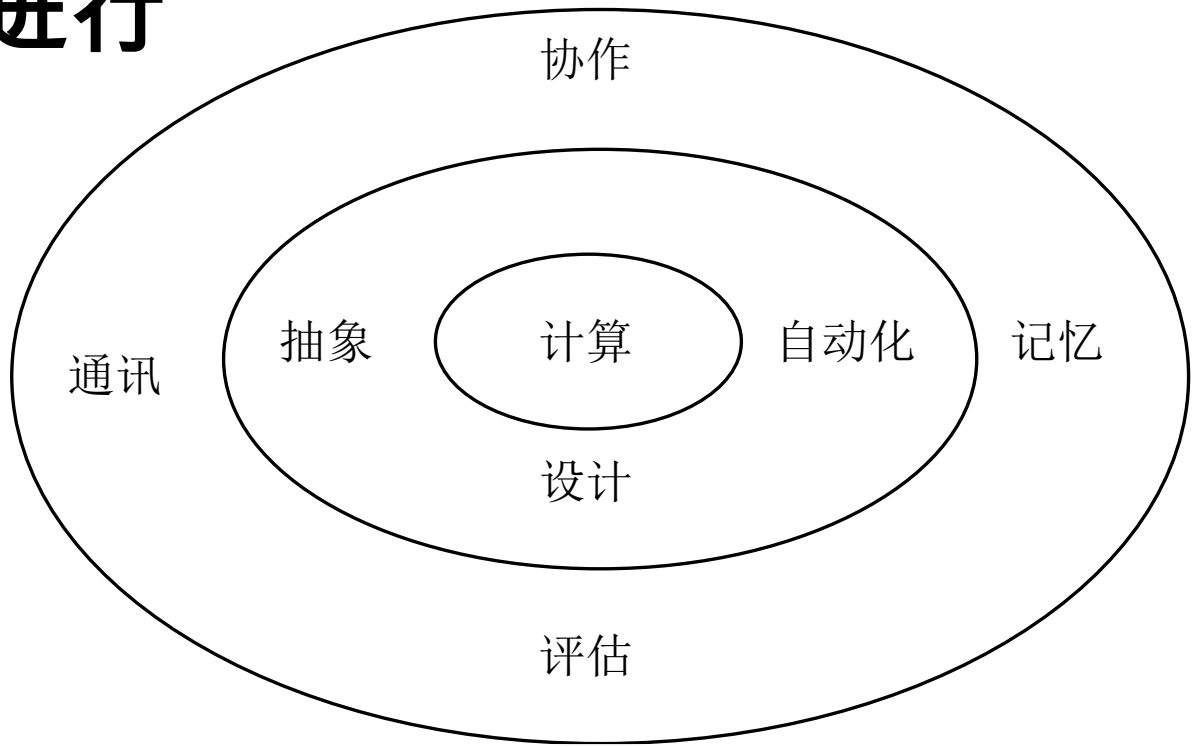
计算 —— 所有学科和人类的关系



计算学科的根本问题

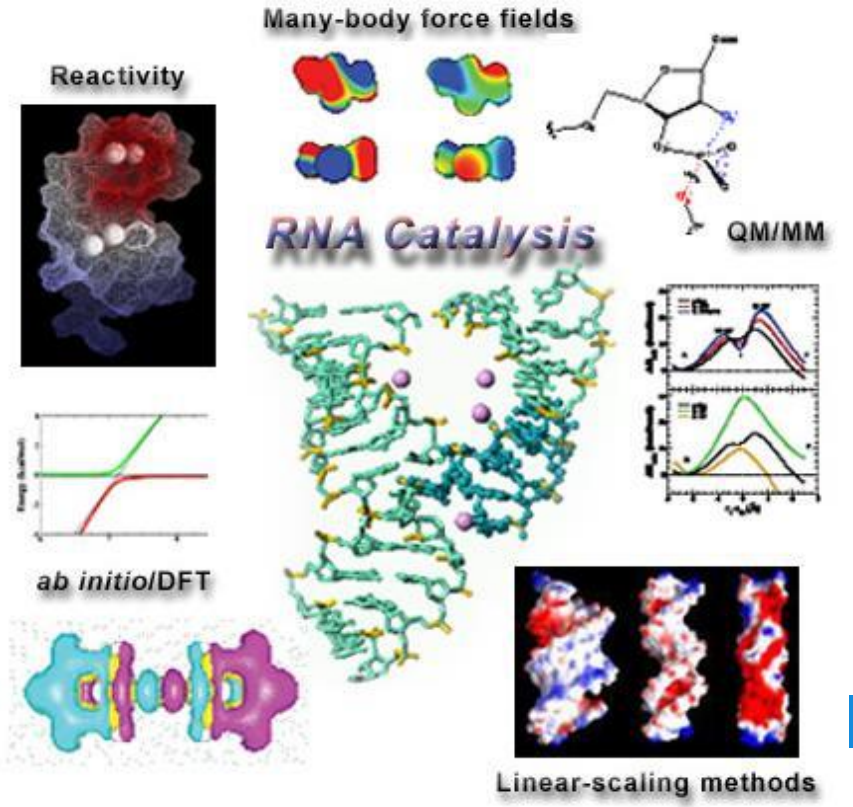
- 学科的根本问题
  - **什么能被（有效地）自动进行**

- 运用计算机科学的基础概念
  - **进行问题求解**
  - **系统设计**
  - **以及人类行为理解**



计算思维

- 2006年3月，美国卡内基·梅隆大学计算机科学系主任周以真教授在《Communications of the ACM》提出
- 计算思维是运用计算机科学的基础概念进行问题求解、系统设计、以及人类行为理解等涵盖计算机科学之广度的一系列思维活动
- **理论科学**、**实验科学**和**计算科学**作为科学发现三大支柱，正推动着人类文明进步和科技发展。



讲座背景

Web信息处理
队列、图、字符、矩阵散列、排序、索引、检索

人工智能
广义表、集合、搜索树及各种有向图

图形图像
队列、栈、图、矩阵、空间索引树、检索

数据库概论
线性表、多链表、排序及B+索引树

操作系统
队列、存储管理表、排序及目录树

编译原理
字符串、栈、散列表及语法树

算法分析与设计

数据结构与算法实习

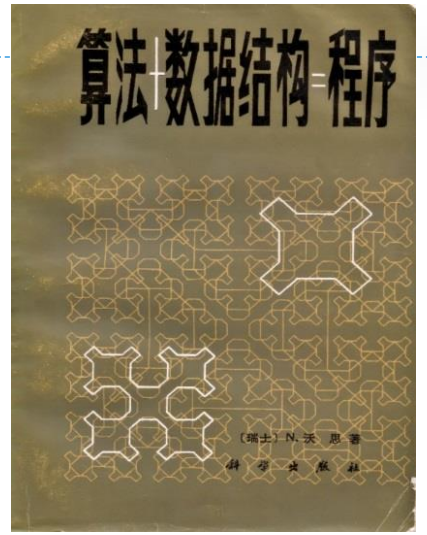
数据结构与算法

程序设计实习

概率统计

计算概论

集合论与图论

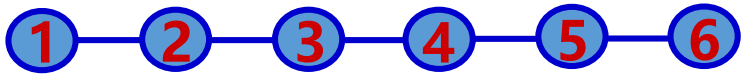


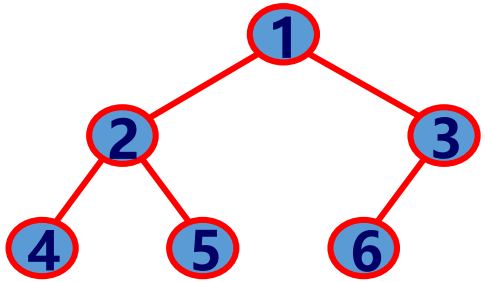
算法 + 数据结构

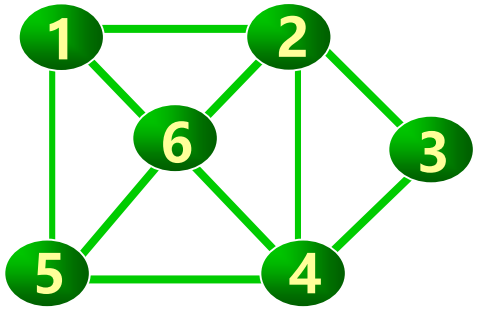
- 问题（problem）：从输入到输出的一种映射函数
- 数据结构（Data Structure）：逻辑数据结构在计算机中的存储表达，支持相应的操作
- 算法（algorithm）：对特定问题求解过程的描述方法
- 程序（program）：算法在计算机程序设计语言中的实现

数据结构的逻辑组织

- 线性结构
  - 线性表（表，栈，队列，串等）
- 非线性结构
  - 树（二叉树，Huffman树，
  - 二叉检索树等）
  - 图（有向图，无向图等）
- 图 ⊇ 树 ⊇ 二叉树 ⊇ 线性表

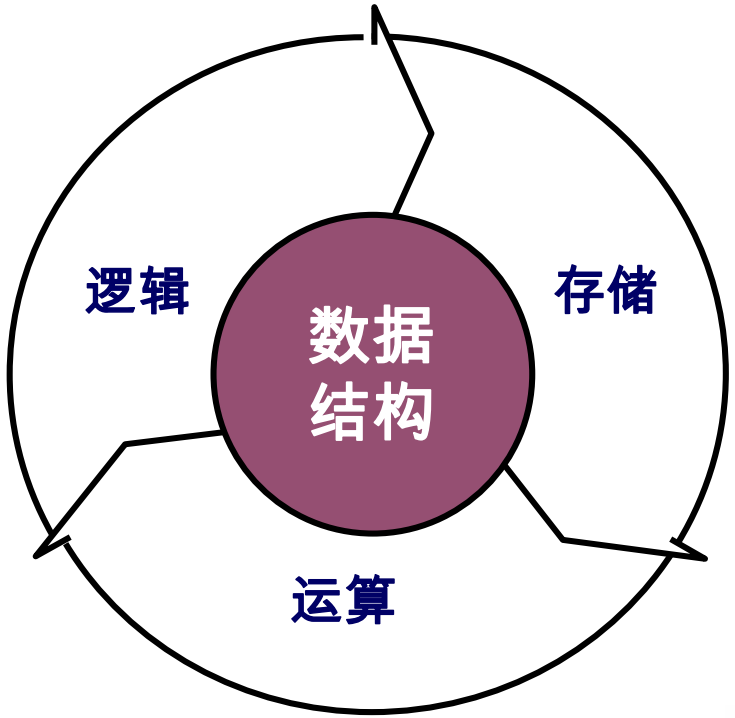


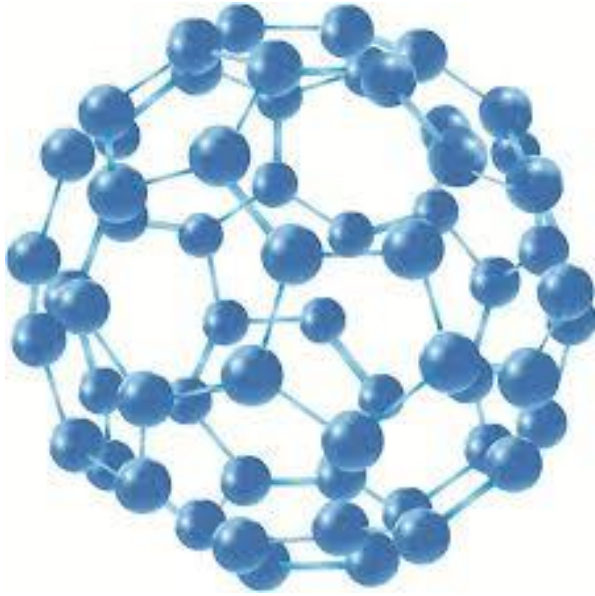


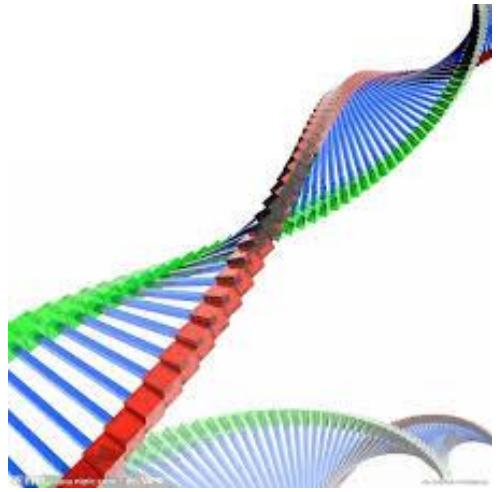


1.2 什么是数据结构

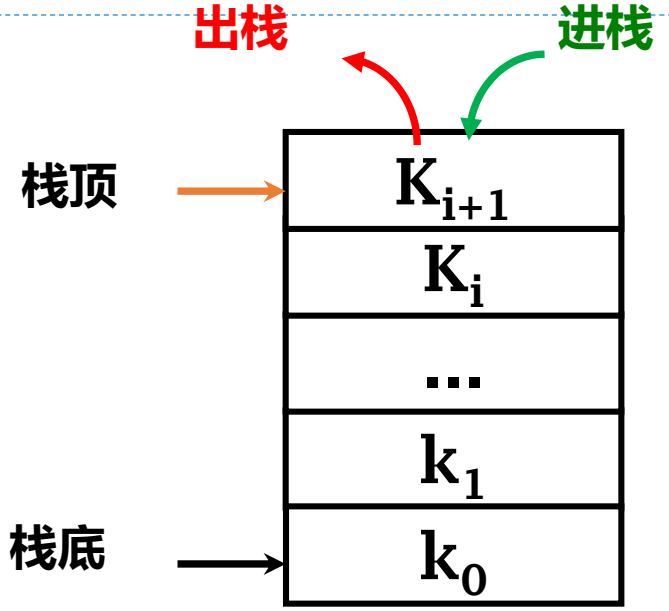
- **结构: 实体 + 关系**
- 数据结构:
  - 按照逻辑关系组织起来的一批数据,
  - 按一定的存储方法把它存储在计算机中
  - 在这些数据上定义了一个运算的集合







例：栈的抽象数据类型ADT

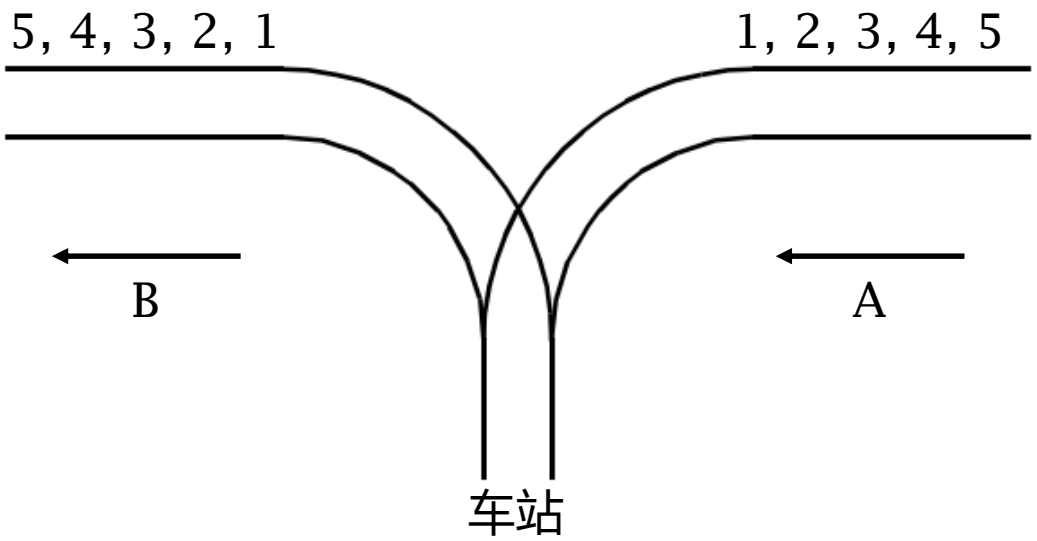


- 逻辑结构：线性表
- 操作特点：限制访问端口
  - 只允许在一端进行插入、删除操作
  - 入栈（ push ）、出栈（ pop ）、取栈顶（ top ）
    判栈空（ isEmpty ）

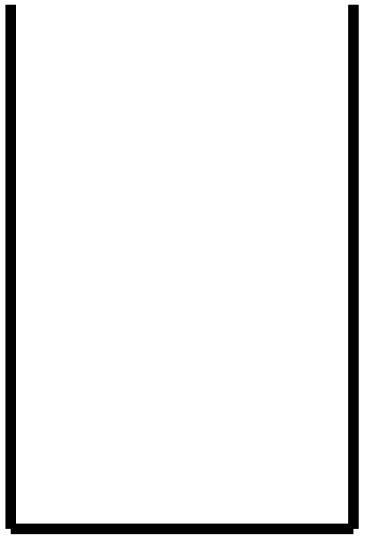
```
template <class T>                  // 栈的元素类型为 T
class Stack {
public:                             // 栈的运算集
    void clear();                   // 变为空栈
    bool push(const T item);        // item入栈，成功返回真，否则假
    bool pop(T & item);             // 弹栈顶，成功返回真，否则返回假
    bool top(T& item);              // 读栈顶但不弹出，成功真，否则假
    bool isEmpty(;                  // 若栈已空返回真
    bool isFull();                  // 若栈已满返回真
};
```

火车进出栈问题

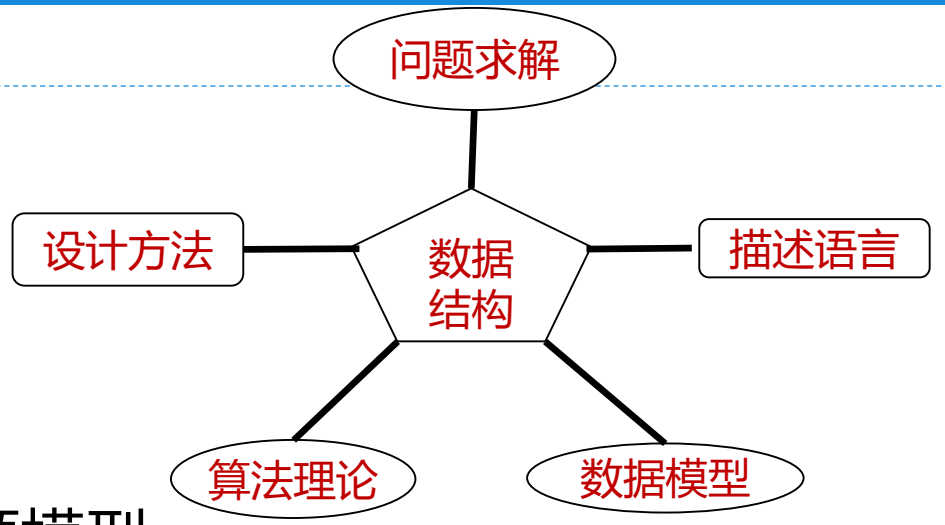
- 判断火车的出栈顺序是否合法
  - http://poj.org/problem?id=1363
- 编号为1,2,…,n的n辆火车依次进站，给定一个n的排列，判断是否是合法的出站顺序？



利用合法的重构找冲突



1.1 问题求解



- 编写计算机程序的目的？
  - 解决实际的应用问题
- 问题抽象
  - 分析和抽象任务需求，建立问题模型
- 数据抽象
  - 确定恰当的数据结构表示数学模型
- 算法抽象
  - 在数据模型的基础上设计合适的算法
- 数据结构 + 算法，进行程序设计
  - 模拟和解决实际问题

数据结构与问题求解

- 逻辑判断 —— 决策树
- 农夫过河 —— 状态表示、路径搜索
- 河内塔 —— 递归树
- 数独游戏 —— 十字链、完美覆盖
- 社会网络 —— 图的应用

职业判断

- 有A、B、C、D 四个人，其中1个是程序员，1个是教师，1个是工人，1个是会计。请根据下述条件判断他们的职业。

(1) A和B是邻居，每天一起骑车上班。

(2) A比C年龄大。

(3) A和D业余喜欢一起练武术。

(4) 程序员每天乘公交车上班。

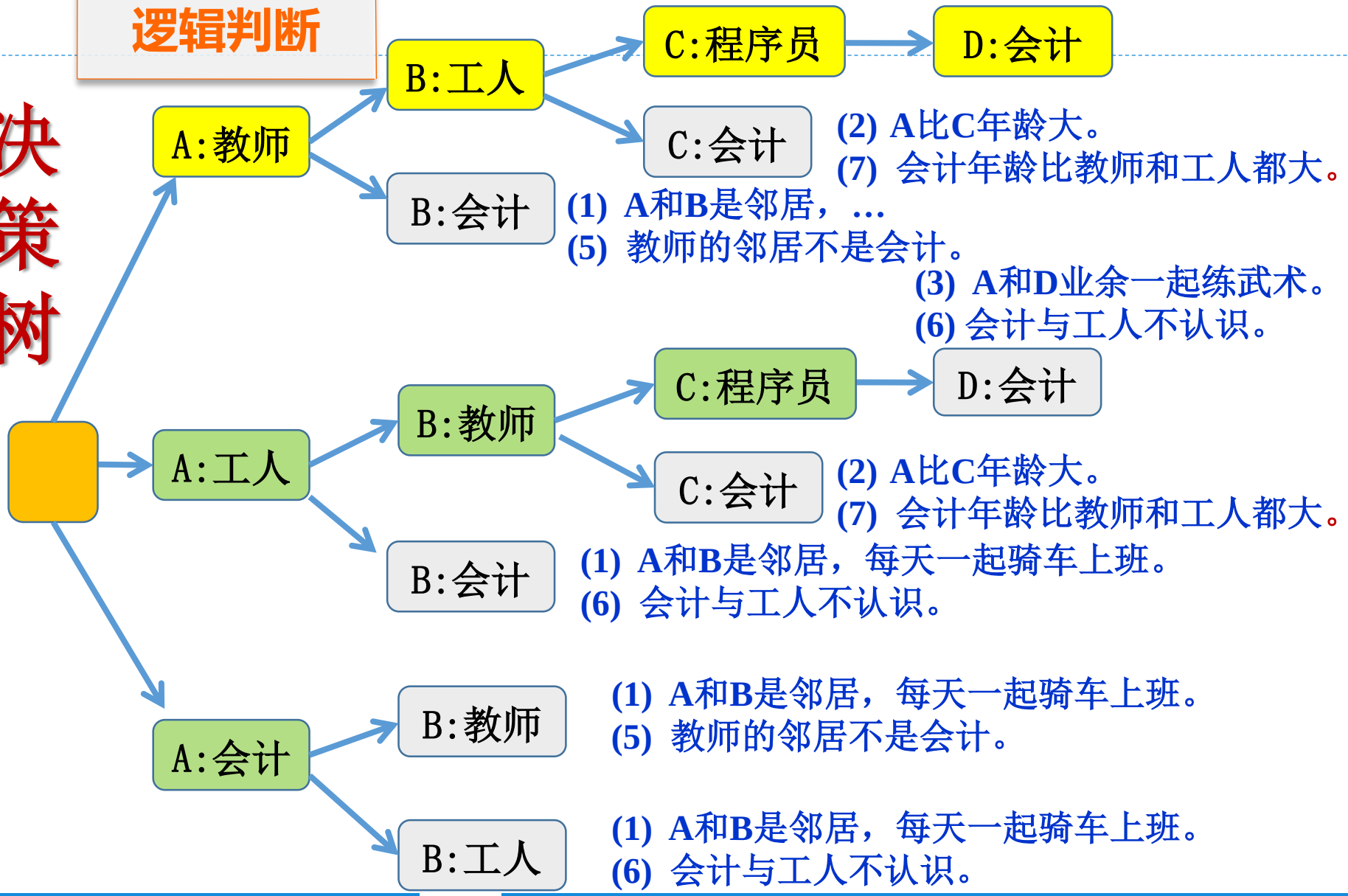
(5) 教师的邻居不是会计。

(6) 会计与工人互不认识。

(7) 会计的年龄比教师和工人的年龄都大。

(1),(4)⇒A与B不是程序员

决策树



(1) A和B是邻居，每天一起骑车上班。

(2) A比C年龄大。

(3) A和D业余喜欢一起练武术。

(4) 程序员每天乘公交车上班。

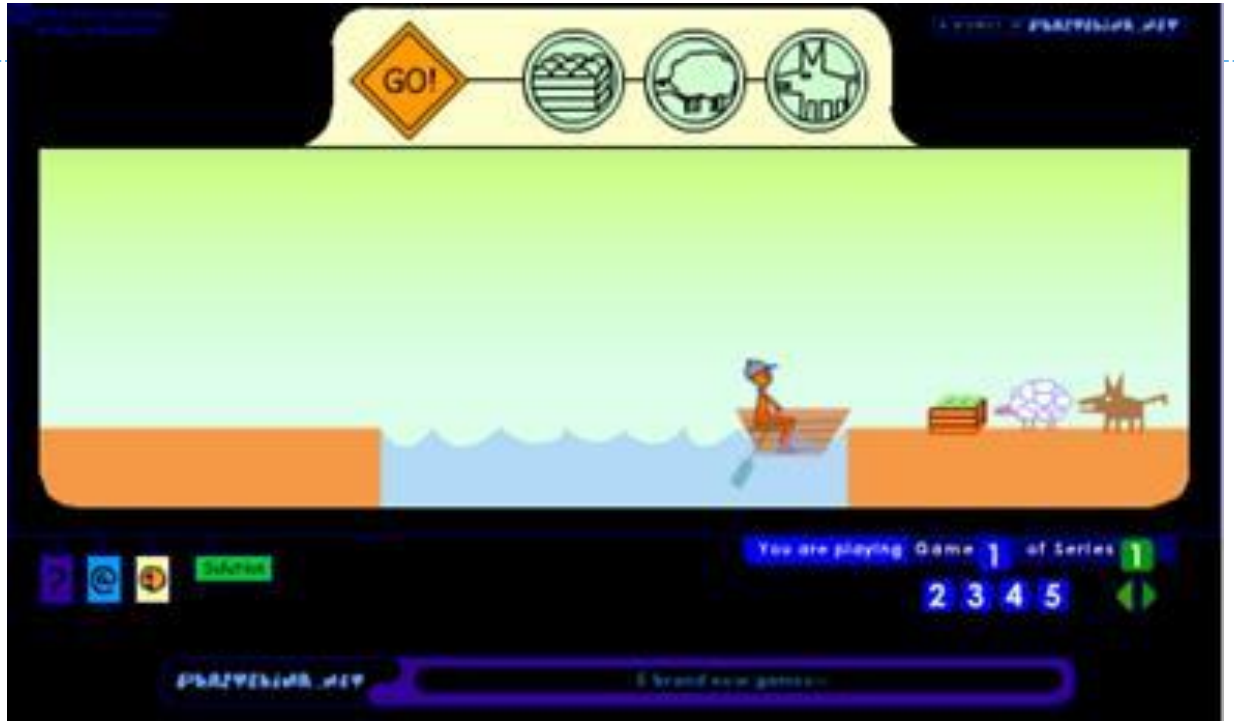
(5) 教师的邻居不是会计。

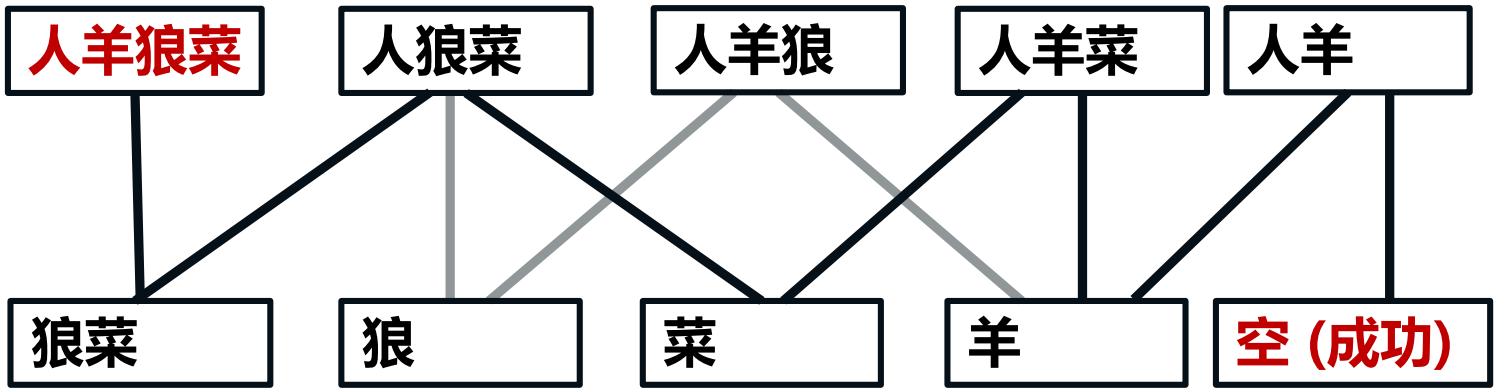
(6) 会计与工人互不认识。

(7) 会计的年龄比教师和工人的年龄都大。

农夫过河问题

- **问题抽象**：“人狼羊菜”乘船过河
  - 只有人能撑船，船只有两个位置（包括人）
  - 狼羊、羊菜不能在没有人时共处

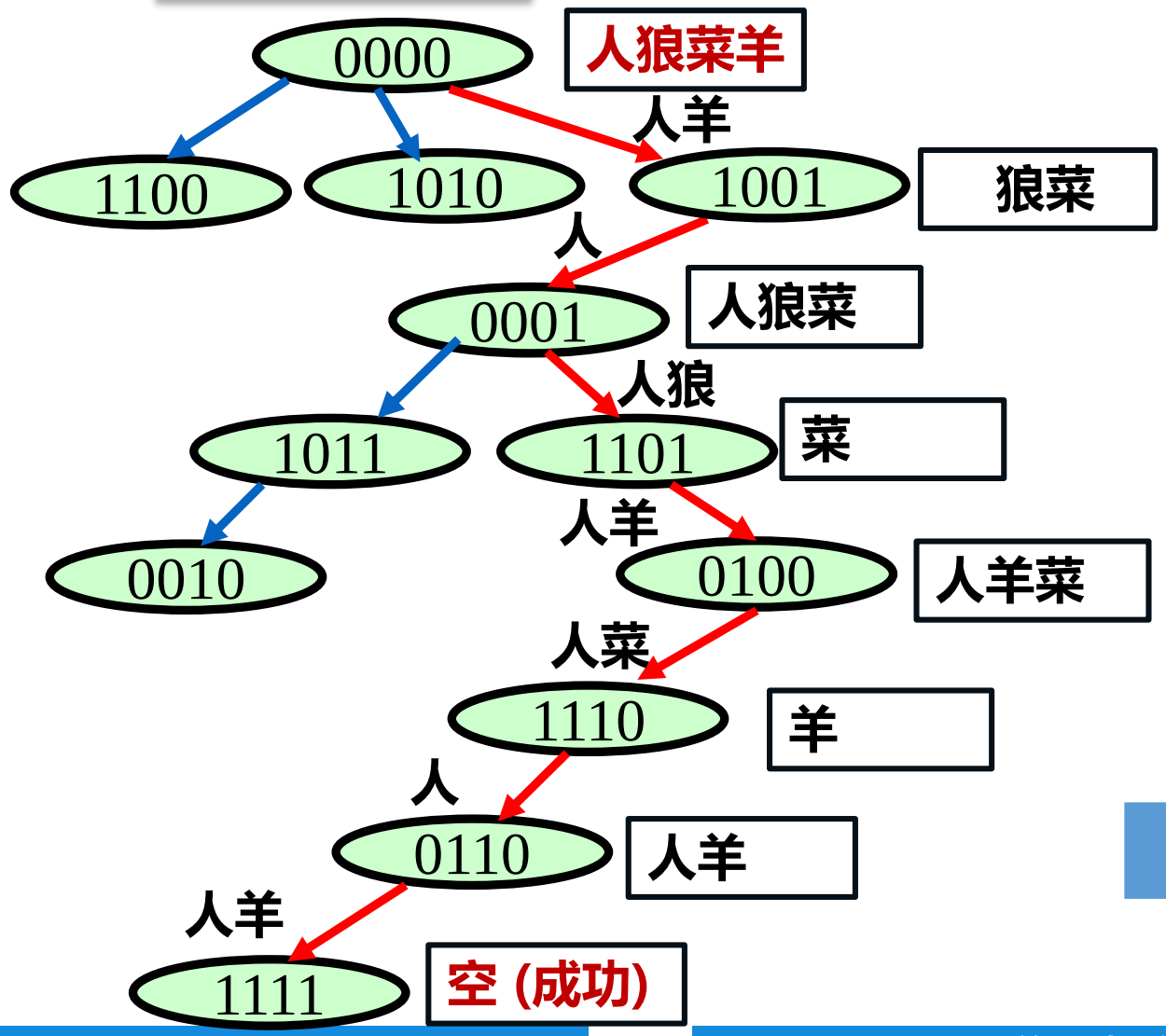


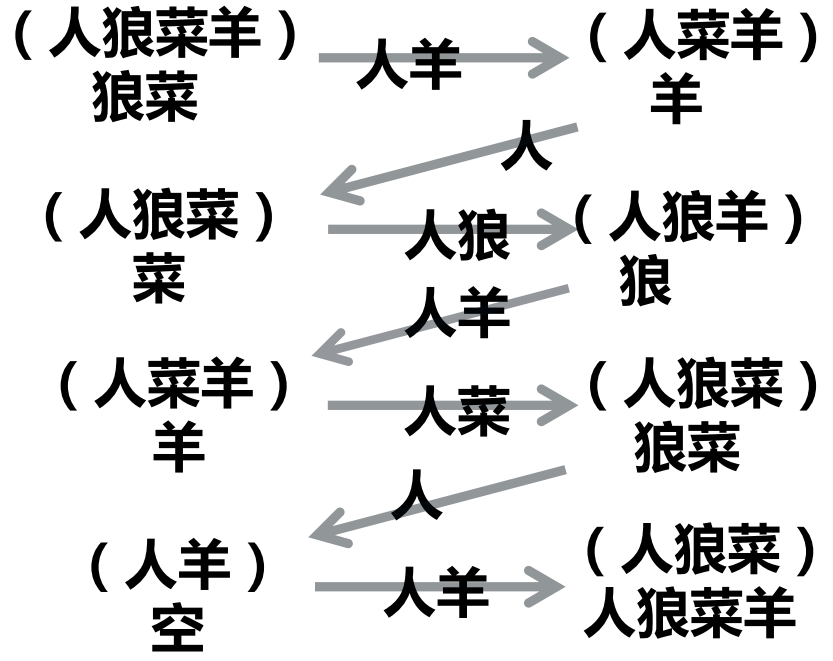


建模

结点：状态
边：　转移
状态：初始、终止
策略：路径

算法示意





0	1	0	1
人	狼	菜	羊

数据抽象

- 每个角色的位置进行描述
  - 农夫、狼、菜和羊，四个目标各用一位（假定按照农夫、狼、白菜、羊次序），目标在起始岸位置：0，目标岸：1
  - 如 010

0	1	0	1

羊在目标岸（此状态为不安全状态）

数据的表示

- 用整数 status 表示上述四位二进制描述的状态
  - 整数 0x08 表示的状态

1	0	0	0

  - 整数 0x0F 表示的状态

1	1	1	1

- 如何从上述状态中得到每个角色所在位置？
  - 函数返回值为真（1），表示所考察人或物在目标岸
  - 否则，表示所考察人或物在起始岸

确定每个角色位置的函数

```
bool farmer(int status)
{  return ((status & 0x08)  != 0);      }

bool wolf(int status)
{  return ((status & 0x04)  != 0);      }

bool cabbage(int status)
{  return ((status & 0x02)  != 0);      }

bool goat(int status)
{  return ((status & 0x01)  != 0);      }
```

人	狼	菜	羊
1	x	x	x
x	1	x	x
x	x	1	x
x	x	x	1

人	狼	菜	羊
0	1	0	1

安全状态的判断

```
bool safe(int status)                    // 返回 true:安全，false:不安全
{
    if ((goat(status) == cabbage(status)) &&
        (goat(status) != farmer(status)))
            return(false);               // 羊吃白菜
    if ((goat(status) == wolf(status)) &&
        (goat(status) != farmer(status)))
            return(false);               // 狼吃羊
    return(true);                        // 其它状态为安全
}
```

算法抽象

- 问题变为

从状态0000（整数0）出发，寻找全部由安全状态构成的状态序列，以状态1111（整数15）为最终目标。

  - 状态序列中 每个 状态都可以从前一状态通过农夫（可以带一样东西）划船过河的动作到达。
  - 序列中不能出现 重复 状态

算法设计

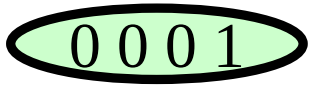
- 定义一个整数队列 moveTo，它的每个元素表示一个可以安全到达的中间状态
- 还需要定义一个数据结构 记录已被访问过的各个状态，以及已被发现的能够到达当前这个状态的路径
  - 用顺序表 route 的第 i 个元素记录状态i是否已被访问过
  - 若 route[i] 已被访问过，则在这个顺序表元素中记入前驱状态值； -1表示未被访问
  - route 的大小（长度）为 16

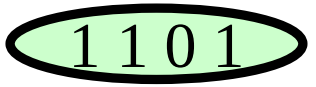
算法实现

```
void solve() {
    int movers, i, location, newlocation;
    vector<int> route(END+1, -1);
                        // 记录已考虑的状态路径
    queue<int> moveTo;
    // 准备初值
    moveTo.push(0x00);
    route[0]=0;
```

算法实现

人狼菜羊






```
while (!moveTo.empty() && route[15] == -1) {
        // 得到现在的状态
        status = moveTo.front();
        moveTo.pop();
        for (movers = 1; movers <= 8; movers <<= 1) {
        // 农夫总是在移动，随农夫移动的也只能是在农夫同侧的东西
                if (farmer(status)  == (bool)(status & movers)) {
                        newstatus = status ^ (0x08 | movers);
                        // 安全的，并且未考虑过的走法
                        if (safe(newstatus) && (route[newstatus] == -1)) {
                                route[newstatus] = status;
                                moveTo.push(newstatus); }
                }
        }
}
```

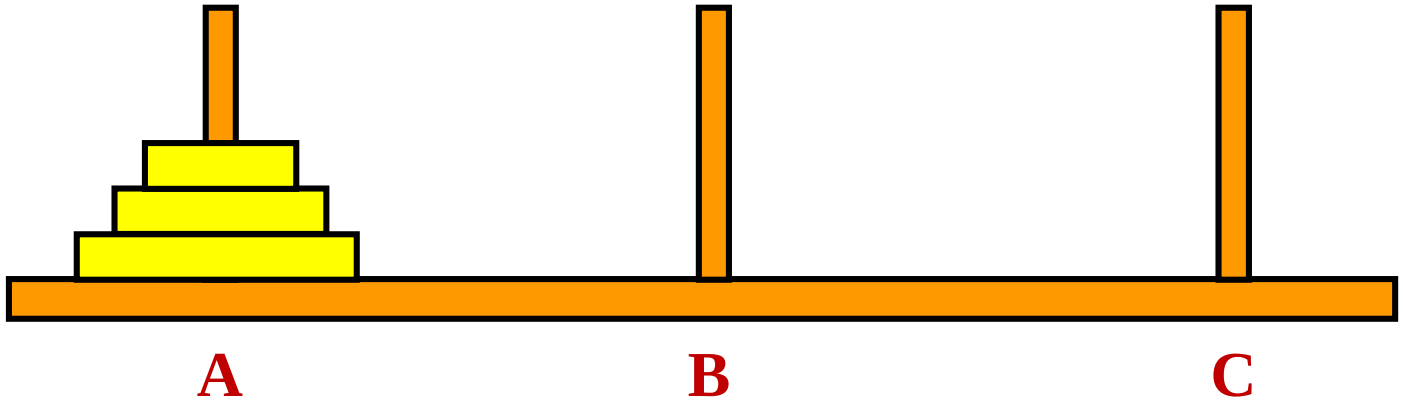
算法实现

```
    // 反向打印出路径
    if (route[15] != -1) {
        cout << "The reverse path is : “ << endl;
        for (int status = 15; status >= 0; status = route[status]) {
            cout << "The status is : “ << status << endl;
            if (status == 0) break;
        }
    }
    else
        cout << "No solution.“ << endl;
}
```

河内塔问题的递归求解程序

http://www.17yy.com/f/play/89425.html

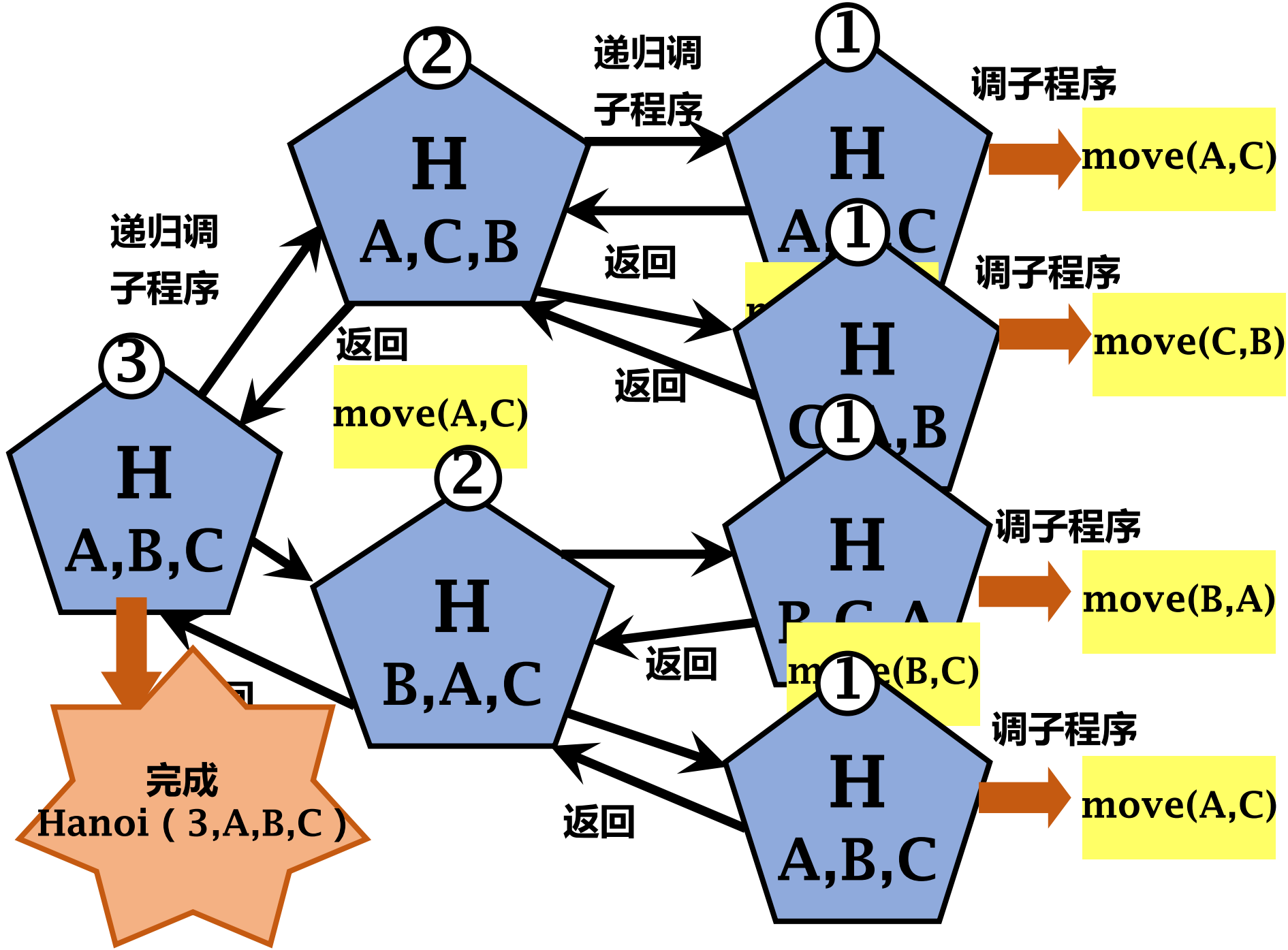
- hanoi(n,X,Y,Z)
  - 移动n个槃环
  - X柱出发，将槃环移动到Z 柱
  - X、Y、Z都可以暂存
    - 大盘不能压小盘



- 例如， hanoi(2, ‘B’, ‘C’ , ‘A’)
  - B柱上部的2个环槃移动 到A 柱

Hanoi 的递归算法

- *n*个盘子的移动方法：
  1. 用同样的方法把 *n*-1个盘子从 A 移到 B
  2. 把最大的盘子从 A 移到 C
  3. 用同样的方法把 *n*-1个盘子从 B 移到 C
- 算法 Hanoi (A, C, *n*)
  1. Hanoi (A, B, *n*-1)
  2. move (A, C)
  3. Hanoi (B, C, *n*-1)
- 递归：把对较大规模问题的处理归约为同样类型的较小规模问题的处理；处理时使用同样的方法；当问题规模足够小时，直接求解。


递归调
子程序
调子程序
move(A,C)
返回
递归调
子程序
返回
move(A,C)
调子程序
move(C,B)
返回
move(B,A)
返回
move(B,C)
调子程序
move(A,C)
返回
完成
Hanoi (3,A,B,C)

算法复杂度

时间复杂度函数	问题规模					
	10	20	30	40	50	60
n	10^{-5}	$2*10^{-5}$	$3*10^{-5}$	$4*10^{-5}$	$5*10^{-5}$	$6*10^{-5}$
n^2	10^{-4}	$4*10^{-4}$	$9*10^{-4}$	$16*10^{-4}$	$25*10^{-4}$	$36*10^{-4}$
n^3	10^{-3}	$8*10^{-3}$	$27*10^{-3}$	$64*10^{-3}$	$125*10^{-3}$	$216*10^{-3}$
n^5	10^{-1}	3.2	24.3	1.7 分	5.2 分	13.0 分
2^n	.001 秒	1.0 秒	17.9 分	12.7 天	35.7 年	366 世纪
3^n	.059 秒	58 分	6.5 年	3855 世纪	$2*10^8$ 世纪	$1.3*10^{13}$ 世纪

算法的评价

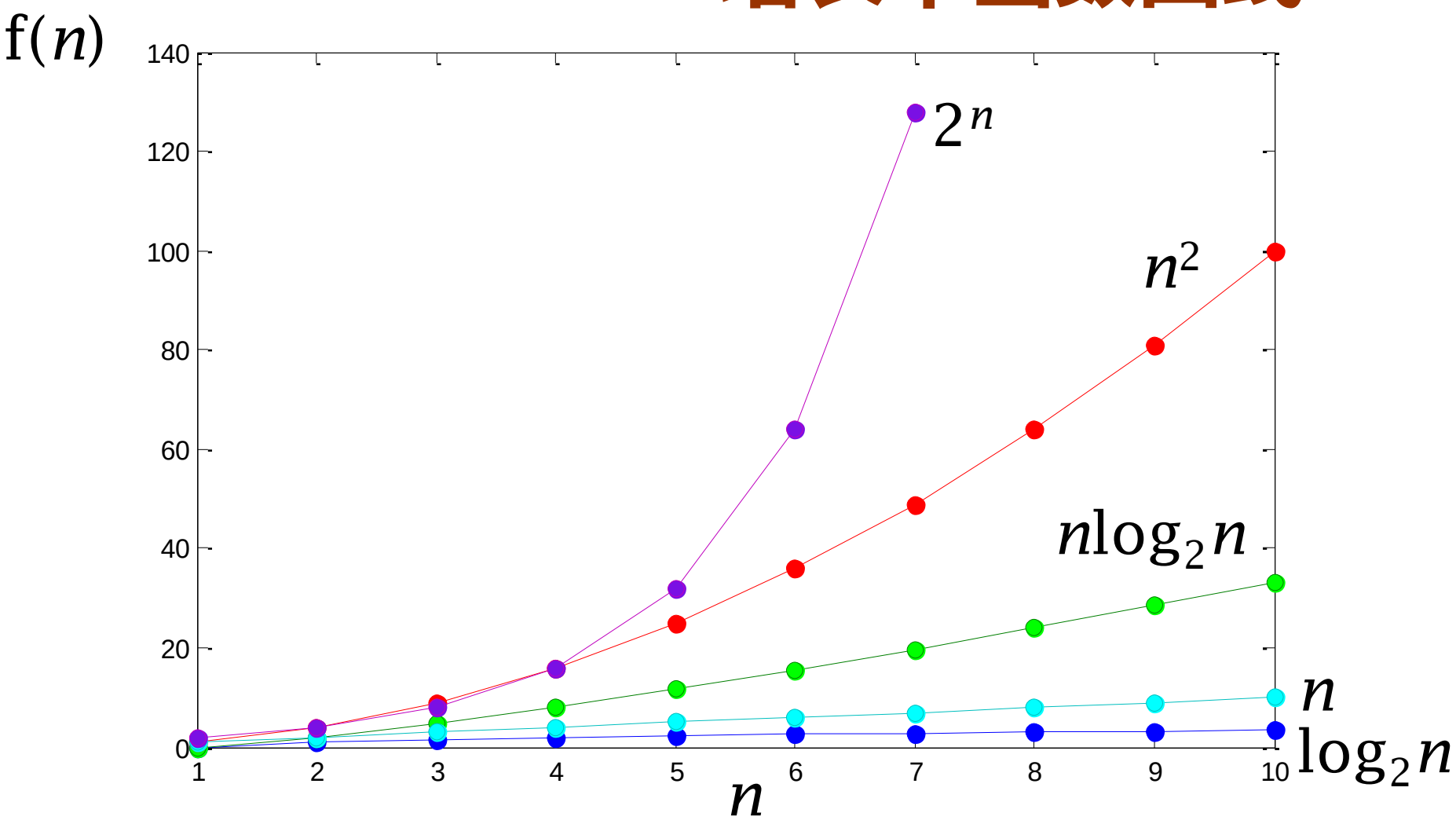
- 设 $T(n)$表示 n个盘子的移动次数

$$T(n) = 2\,T(n-1) + 1,\ \ T(1) = 1$$

$$\begin{aligned}
T(n) &= 2T(n-1)+1 \\
&= 2[2T(n-2)+1]+1 = 2^2T(n-2)+2+1 \\
&= 2^2[2T(n-3)+1]+2+1 = 2^3T(n-3)+2^2+2+1 \\
&= \ldots \\
&= 2^{n-1}T(1)+2^{n-2}+\ldots+2+1 \\
&= 2^{n-1}+2^{n-2}+\ldots+2+1 = 2^n-1
\end{aligned}$$

- 1秒移1次，64个盘子要多少时间？（5000亿年）

增长率函数曲线



Sudoku 数独游戏

- Sudoku就是我们平常提到的数独。数独分为2阶、3阶、以及高阶数独。
- 最简单的是2阶，我们平常接触的是3阶，再次更高的有4阶、5阶甚至更高。

5						3		
	9		5			4		
		4				7		
	5	1		3	7	2	8	9
3		2		8		6		4
		8		5	2	1	3	7
	3	5				9		
6		9				8	2	3
	8			2	3			6

			14	13		6		1			9		5		8
			7			11	5		10	16		1			
			1			8	7		3				6		12
3	11	10	9		14					6				2	
			2	1		3		5					4		15
5	12					2	11			1	8		16		
		16	15				4		12			10		14	9
			10	15	12				2	13					11
4					6	12				7	2	16			
16	3		12			5		8				2	15		
		15		9	4			16						1	13
2		6					16		15		1	8			
	7				16					8		5	10	12	3
10		4				1		9	13			6			
			8		15	4		7	5			14			
15		1		10			8		6		16	7			

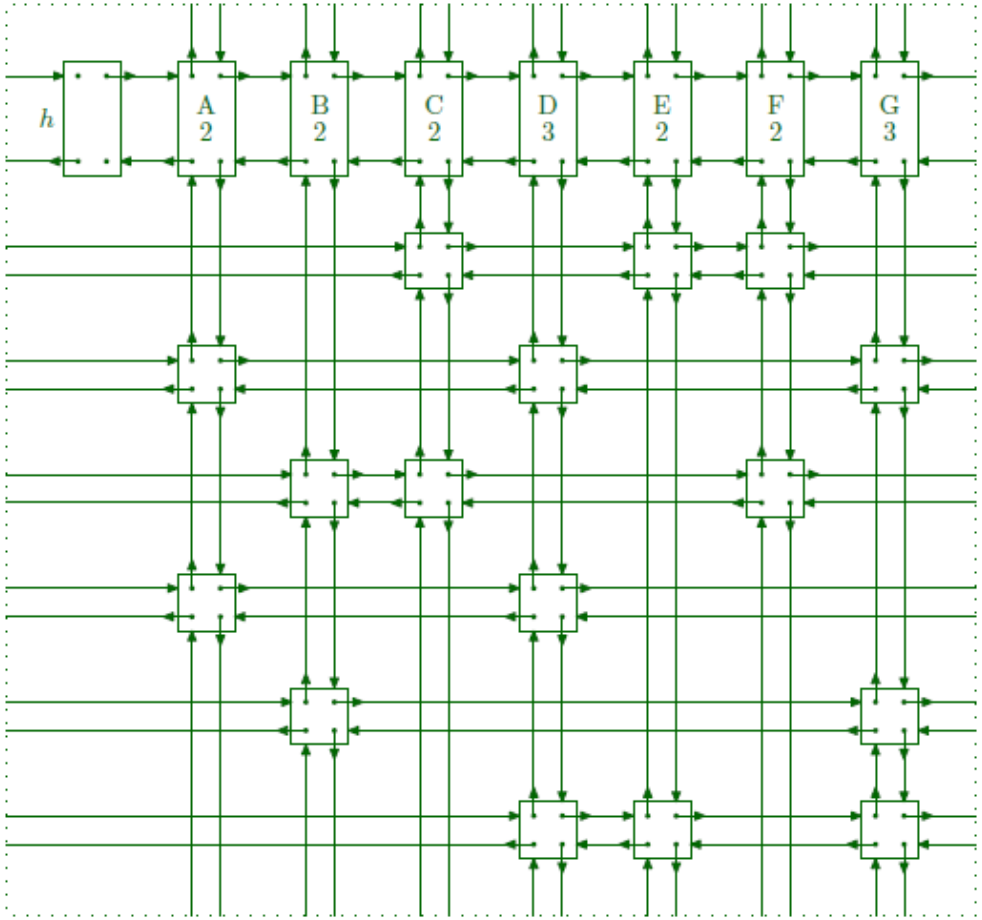
数独游戏

- n阶数独是一个n×n行n×n列的矩阵，矩阵中每个数都是1~n×n的整数。
- 数独要满足以下要求
  - 每行的数字不重复
  - 每列的数字不重复
  - 把矩阵看做n×n个n×n的子矩阵拼接而成，则得到的每个子矩阵中的数字不重复

Dancing Links 的引入

- 主体是个十字链表
- 每个节点有个4个指针分别链到上下左右的邻居节点（如果到了左边界，则从最右边开始循环查找，其他方向类似）
- 最上面一行额外增加一行表头节点。并且额外的记录这一列当前的有多少节点。
- 每个普通节点存一个到表头节点的链接。
- 所有表头节点的左边增加一个哨兵节点。

Donald E. Knuth, Stanford University



核心操作

- Dancing Links 上有两个基础的操作 Cover 和Recover
- Cover 是指用第 i 行的第 j 个元素覆盖 j 这一列，因为是所求的是 Exact Cover，所以在 j 这一列有元素的覆盖方案都不可行了，所以也同时将他们在这个过程中去除。
- Recover 则是 Cover 的逆运算，利用之前残留的信息，重新修补链接。

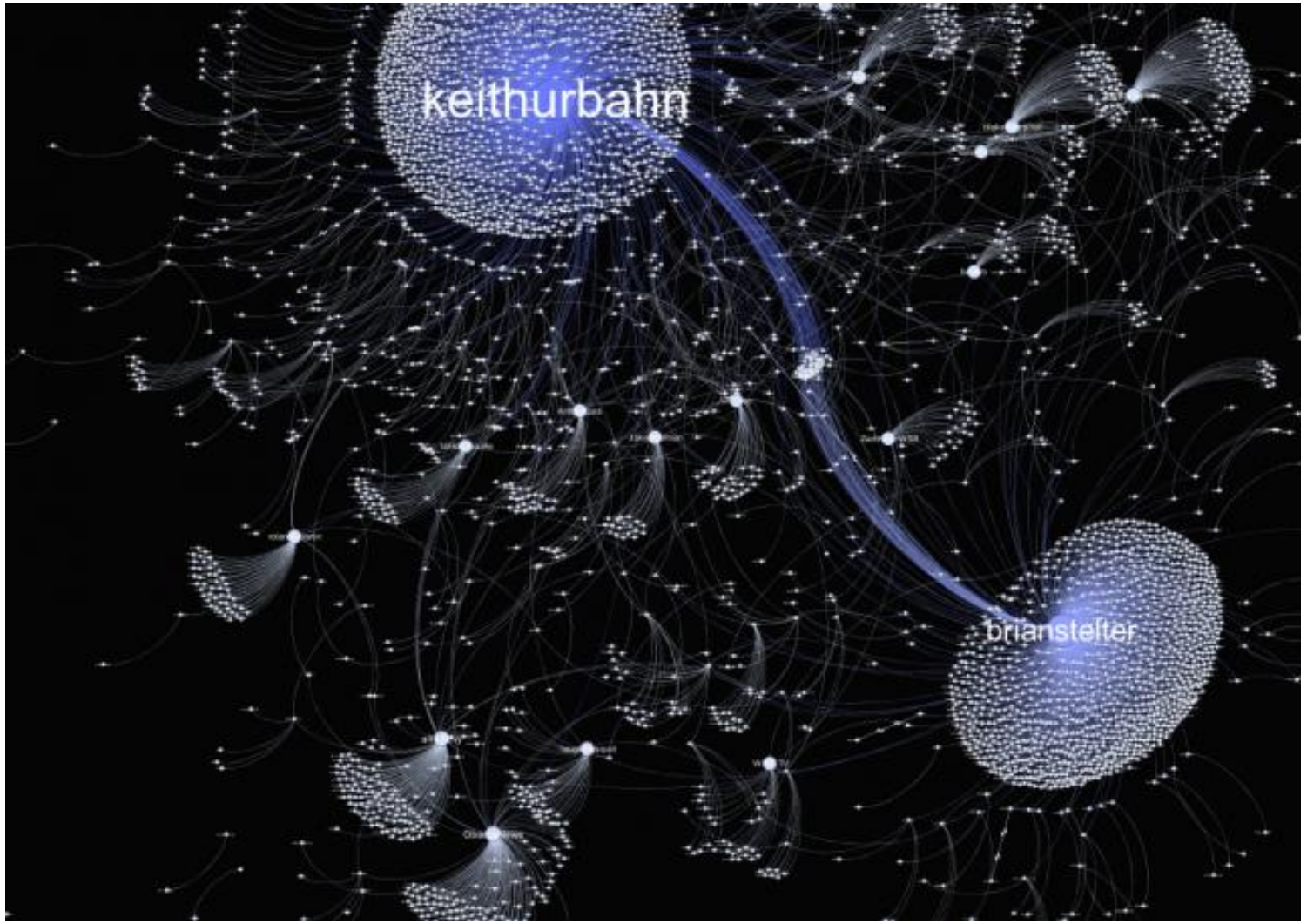
抽象数据类型

```
class Node {
    Node* u,d,l,r;   // 上下左右四个邻居节点
    Node* pos;       // 该列的表头节点
    int size;        // 表头结点中该列的当前可填空格数
    Datatype data; //表示这个覆盖方案的信息
}
```

数独游戏

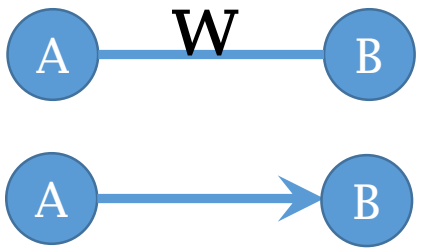
```
bool AlgorithmX() {
    if 所有表头都被覆盖
        then return true
    遍历表头节点，找到 size 域最小的 k
    if (k.size == 0)
        then return false;
    Cover(k);
    for (i←k.u; i!=k; i←i.u){
        for (j←i.r; j!=i; j←j.r) Cover(j.pos);   // 记录方案i
        if algorithmX() then return true;
        for (j←i.r;j!=i;j←j.r) Recover(j.pos);  //去除方案i
    } }
```

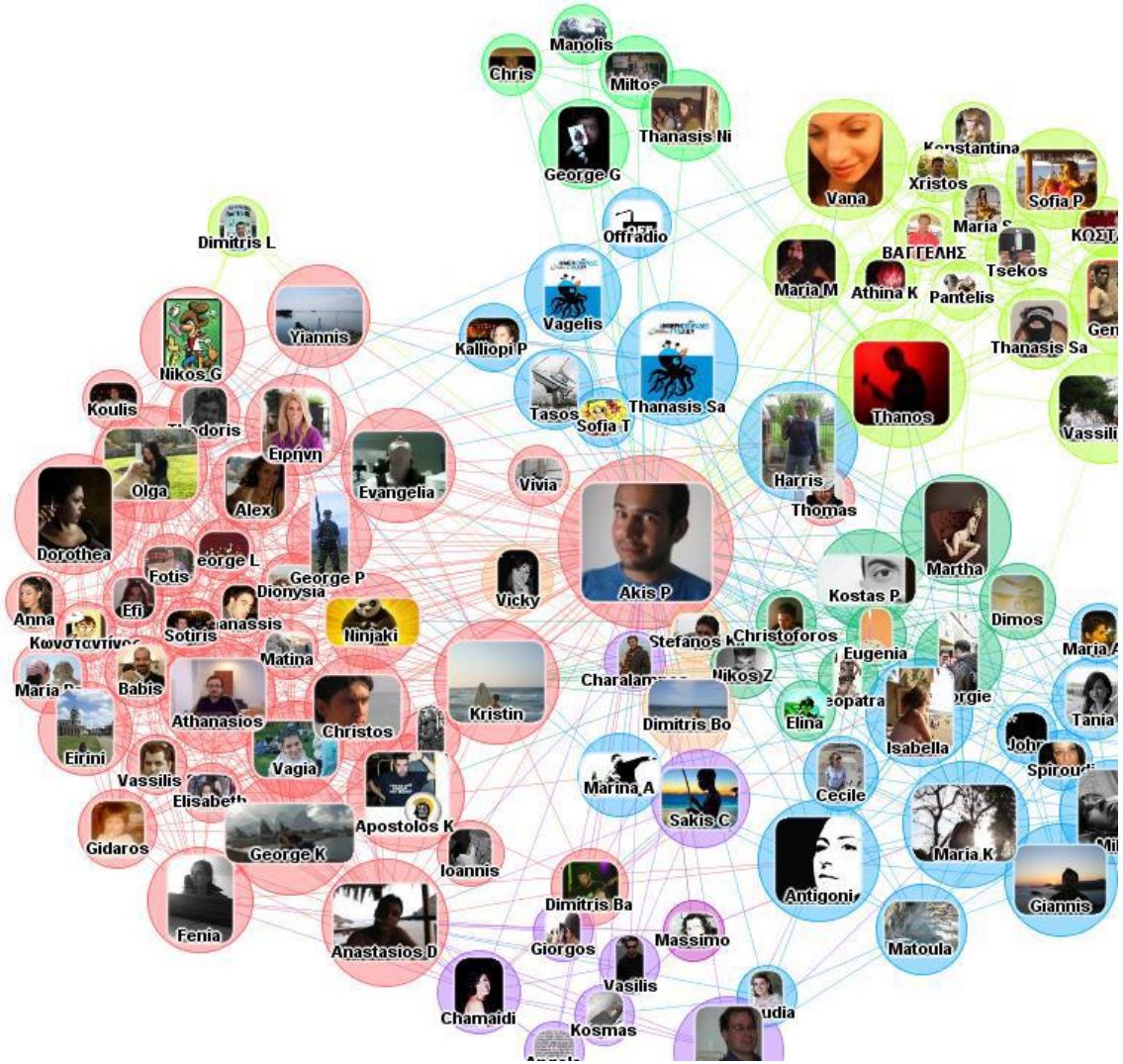
本拉登被击毙



社会网络图

- 往往用图模型
  - 结点（node）→ 用户
  - 边（edge）→ 关系或交互

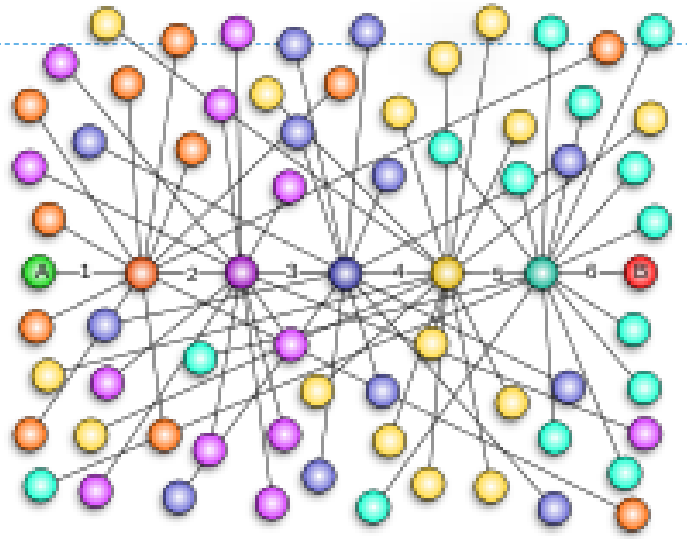




社会网络的特点

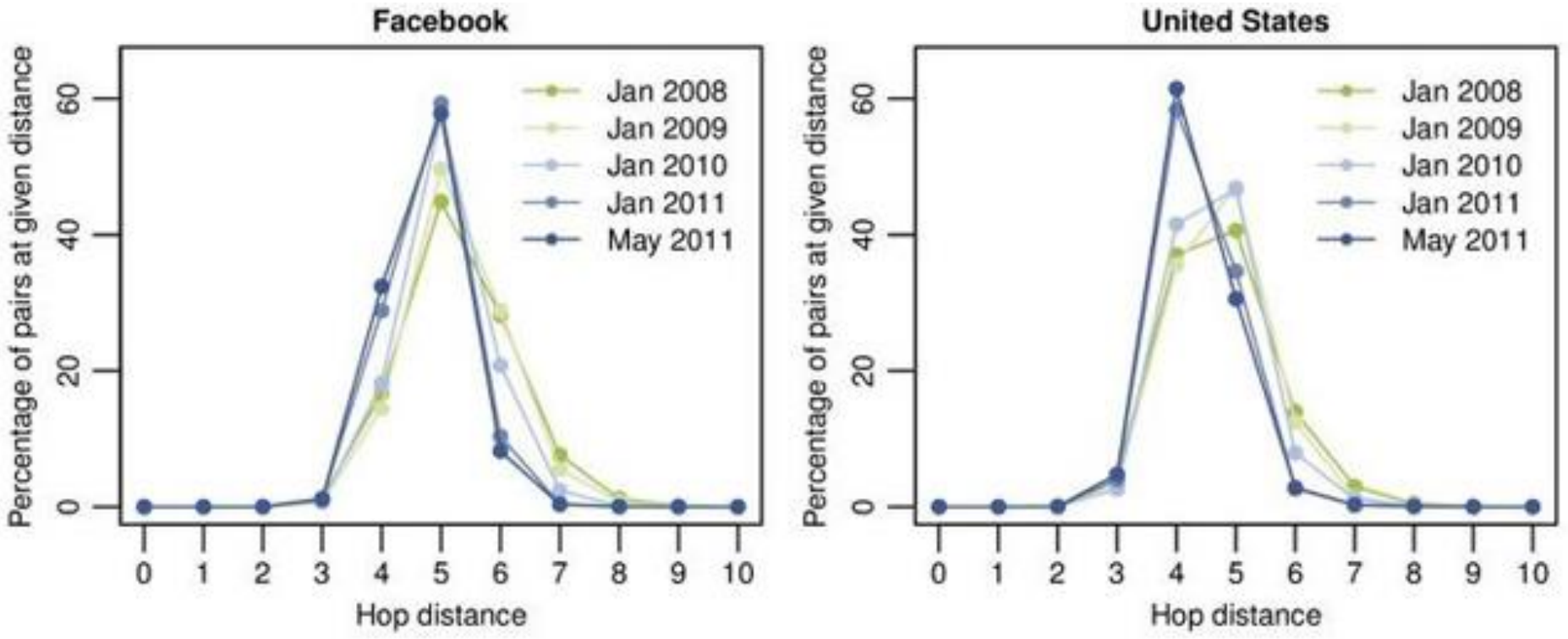
- 小世界网络（Small World Networks）！
- **高群聚度**
  - 假设该结点有k个邻居，集聚系数为此 k 个邻居间真正形成的连结总数除以可能形成的连结总数
  - 网络的集聚系数为所有结点集聚系数的平均值
- **低分隔度**
  - 两个结点之间的分隔度为最短路径的长度
  - 网络的分隔度为所有结点对分隔度之平均值

小世界理论



- 20世纪60年代，美国心理学家Stanley Milgram设计了一个连锁信件实验。
- 从不同发信人到指定收件人，每封信平均经手6.2次到达。
- 推论：世界上任意两个人之间建立联系，最多只需要6个人
- **六度分隔理论（Six Degrees of Separation）**

Facebook 将世界变得更小
—— 六度分隔缩水至四度



有趣的实例（1/2）

- 认识右边这个演员吗?

- Kevin Bacon 六度理论
  - Kevin Bacon 是一名好莱坞演员，生平出演的电影无数
  - 全世界绝大多数的演员的都可以在有限步数内与 Bacon 相连

有趣的实例（2/2）

- **Bacon数**：研究学者就 Bacon 为中心
  - Kevin Bacon 的 Bacon 数是 0
  - 和他一起演出过的演员的 Bacon 数为1，和 Bacon 数为 1 的演员一起演出过的演员的为 2，…
  - 全世界绝大多数的演员的 Bacon 数不会超过 5，这就是 Kevin Bacon 六度理论。
- 对超过 133 万名世界各地的演员的统计得出，
  - 平均“贝肯数”是 ？ 2.981
  - 最大的有限 Bacon 数？ 8

小世界网络的特点

- 少数几个结点拥有非常多的邻居，大多数结点的邻居数目都不多。
  - 自由尺度网络（Scale-free Network）

自由尺度网络特点

- **增长（Growth）**
  - 结点的数目随时间而增加
- **偏好连接（Preferential Attachment）**
  - **富者更富**
  - 幂律（Power-low）分布
  - 新结点加入网络时，更有可能连接到已有很多邻居的结点

思考：社会网络挖掘

- 为什么存在这种短路径？
  - 在不了解网络全局结构时能不能找到这种短路径？
- 为什么存在幂律分布？

……

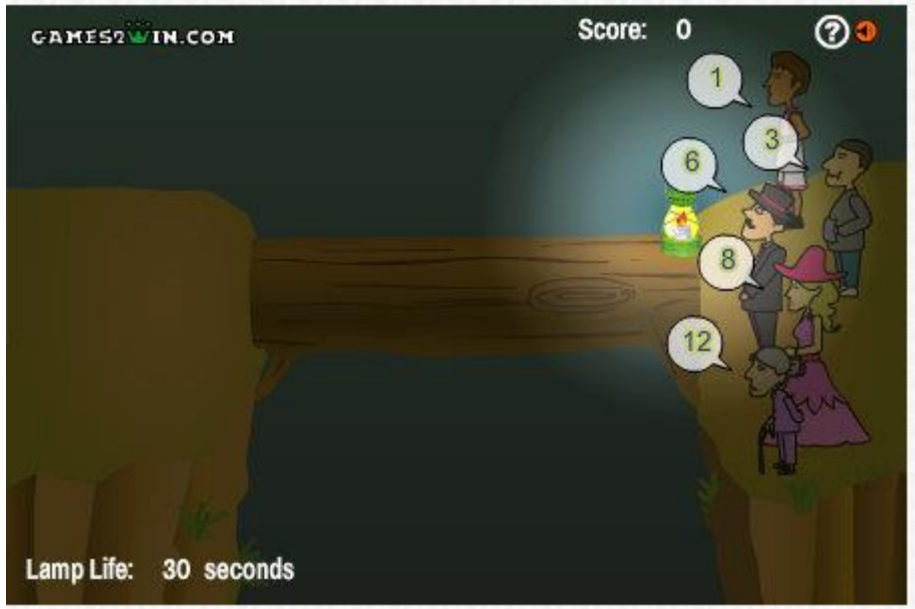
- 多学科综合：数学、计算机科学、社会学、心理学、生物学、经济学、…

思考：集体婚礼

- 三对情侣参加婚礼，三个新郎为A、B、C，三个新娘为X、Y、Z。
- 有人不知道谁和谁结婚，于是询问了六位新人中的三位，但听到的回答是这样的：A说他将和X结婚；X说她的未婚夫是C；C说他将和Z结婚。
- 这人听后知道他们在开玩笑，全是假话。
- 请找出谁将和谁结婚。

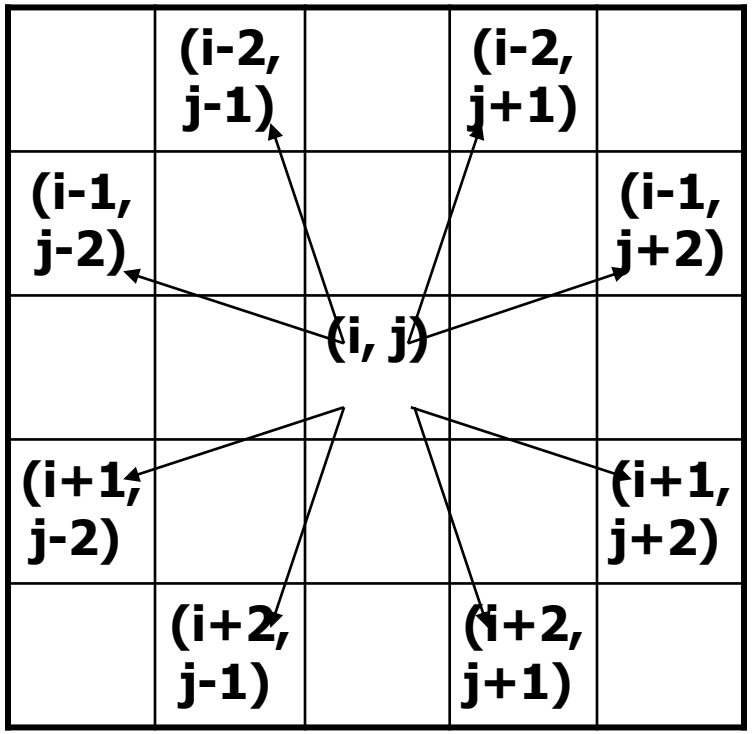
思考：五人提灯过独木桥

- 有一盏灯能使用30秒，要在灯熄灭前过这座桥
- 一家五口人每个人过桥的速度不同：哥哥 1 秒，弟弟 3 秒，爸爸 6 秒，妈妈 8 秒，奶奶 12 秒；
- 每次只能过两个人
- 过去后，对岸要有一个人再把灯送回来



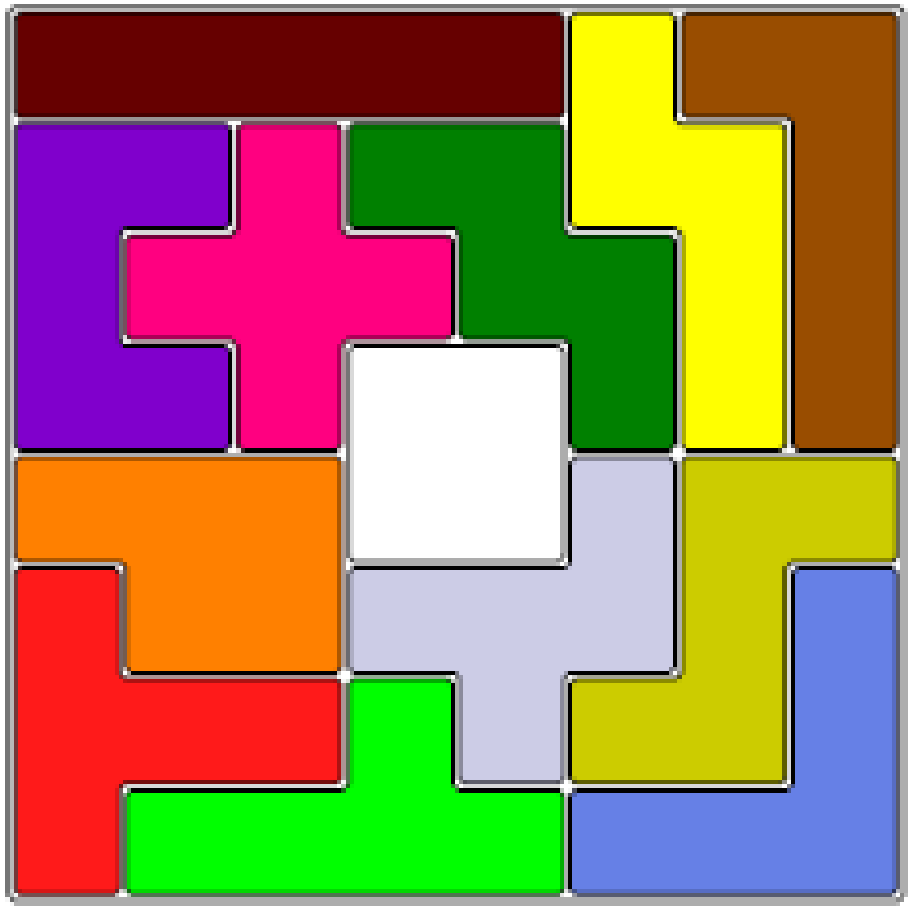
思考：骑士巡游问题

- 给定一个 n × n 的网格，有一个国际象棋的马置于一个方格上。要求找到一条路径，使马按国际象棋的允许走法，从开始的方格出发，不重复地把 n^2 个方格都恰好经过一次。
- 设马所在方格坐标为(i, j)，则它一步可达的方格坐标如图所示。



思考：Pentomino Problem

- 五方连块游戏
- Pentomino Problem是说如何用五方连块填满一个n×m的矩阵（可以挖去一些部分）
- Scott's Pentomino Problem是用FILPNTUVWXYZ这12种五方连块填满8×8的矩阵（挖去中间2×2）



- http://net.pku.edu.cn/dlib/
- http://summer.pku.edu.cn

数据结构与算法

谢谢聆听

国家精品课“数据结构与算法”
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

张铭，王腾蛟，赵海燕
高等教育出版社，2008. 6。“十一五”国家级规划教材