算法设计与分析基础《Introduction to the Design and Analysis of Algorithms》

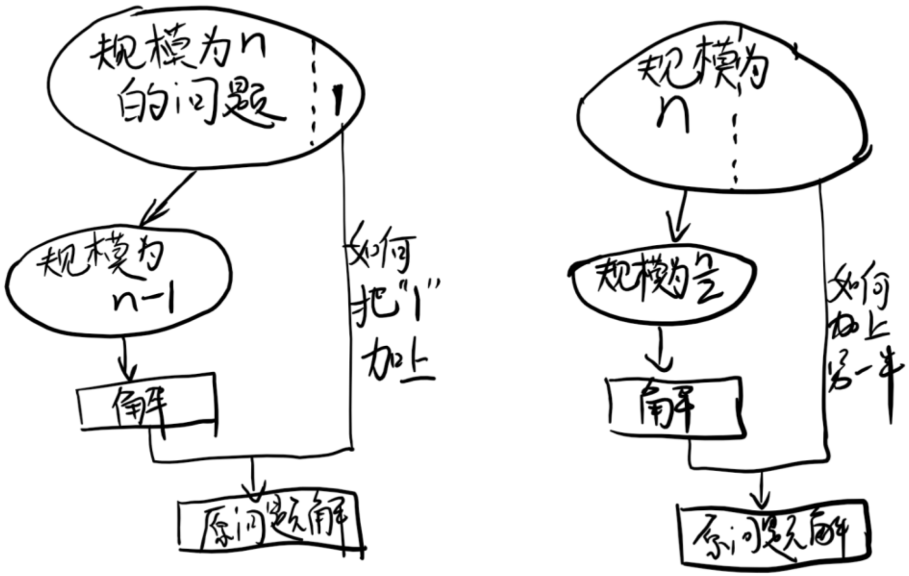
减治法

南京大学软件学院
李传艺
lcy@nju.edu.cn
费彝民楼917

目录

减治法回顾（1）

- 利用给定实例下问题的解和较小规模实例下相同问题的解之间关系
  - 减去一个常量　　　　1
  - 减去一个常因子　　　2
  - 减去的规模是可变的



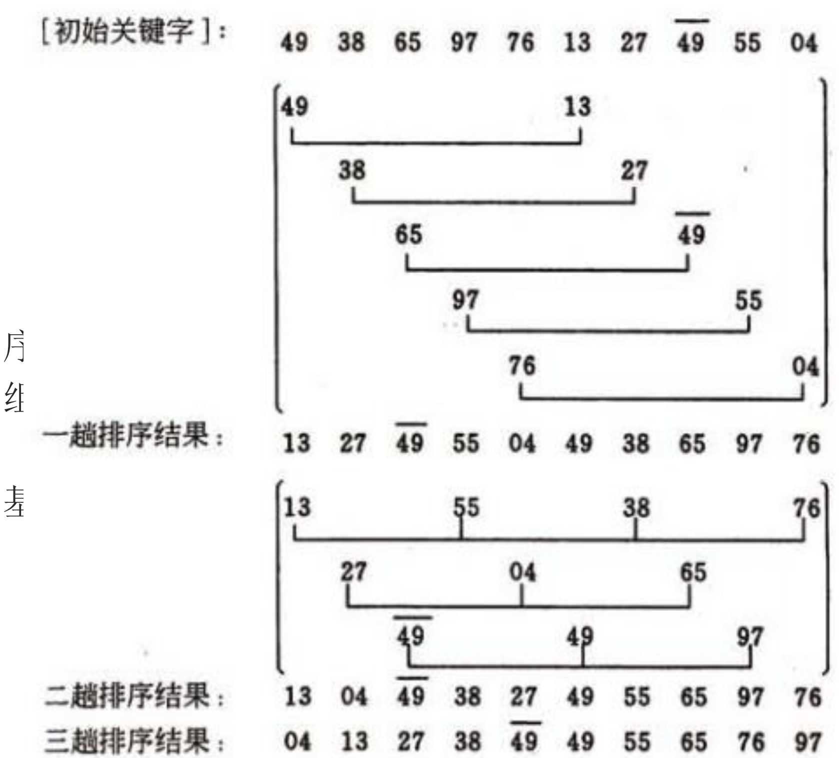
- 从顶至下（递归地）或者自底向上（非递归地）运用该关系

减治法回顾（2）

- 插入排序
  - 减一法：假设前n-1个元素已经排序好了，则如何将最后一个元素**插入**到其它元素中去

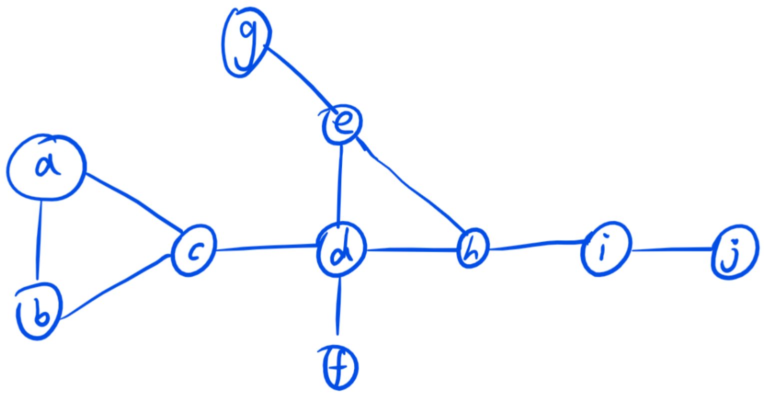
- 算法复杂度
  - 最坏的情况：原来数组是逆序的，$O(n^2)$
  - 最好的情况：原来数组是有序的，$O(n)$
  - 平均情况：$O(n^2)$

- Shell排序
  - 更好的对较大序列进行排序：对一个基本有序的
  - 从大序列构造子序列：选择步长：由大到小，即
  - 每次都对每一个子序列内部进行插入排序
  - 最后一次迭代中只有一个和原来一样长的序列：

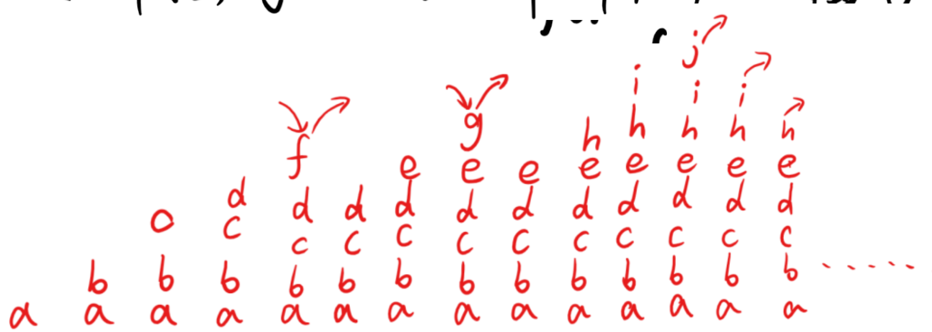


- 折半查找
  - 减常因子

深度优先查找（DFS）、广度优先查找（BFS）

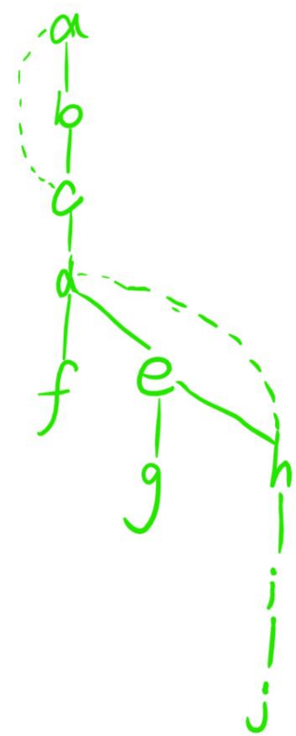


从a开始，将访问到的非端节点压栈



进栈顺序：a b c d f e g h i j

出栈顺序：f g j i h e d c b a



顶点，如果有多

深度优先查找树：

树向边

回边

深度优先查找（DFS）、广度优先查找（BFS）

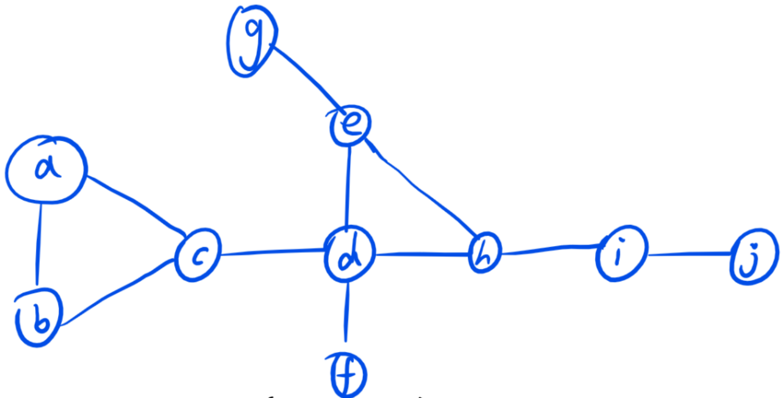
- 深度优先查找
  - 邻接矩阵表示的图
  - 邻接链表表示的图

  - 查找算法的复杂度→就是对表示图的数据结构的查找复杂度
  - 邻接矩阵：$\Theta(|V|^2)$
  - 邻接链表： $\Theta(|V|+|E|)$

- 深度优先遍历的产物
  - 图的类森林的表示方法
  - 产生两种节点顺序：入栈顺序和出栈顺序（不同的选择节点方式会产生不同的次序）

- 深度优先遍历的应用
  - 检查图的连通性
  - 计算图的连通分量？
  - 检查图的无环性？

深度优先查找（DFS）、广度优先查找（BFS）

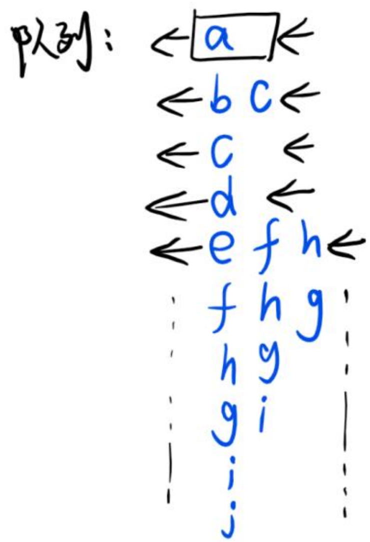


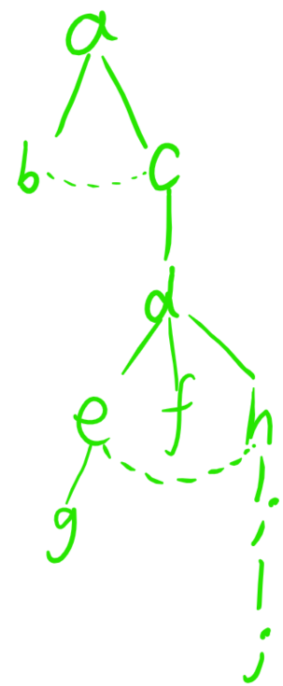
距离为两条边的顶点；直到找到查找键。

度优先则体现了“谨慎”。

从a开始广度优先遍历

- 使用队列：记录每一 一次从 为下一层待访问顶点的父节点





- 广度优先查找树
  - 树向边
  - 交叉边
- 如何递归？

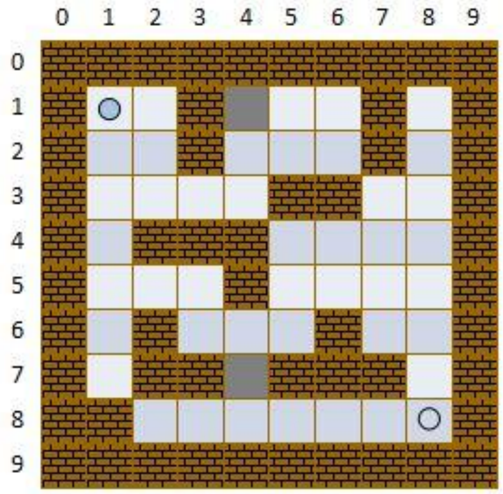
深度优先查找（DFS）、广度优先查找（BFS）

- 广度优先查找算法复杂度
  - 同样与图的表示方式相关
    - 邻接矩阵：$\Theta(|V|^2)$
    - 邻接链表：$\Theta(|V|+|E|)$

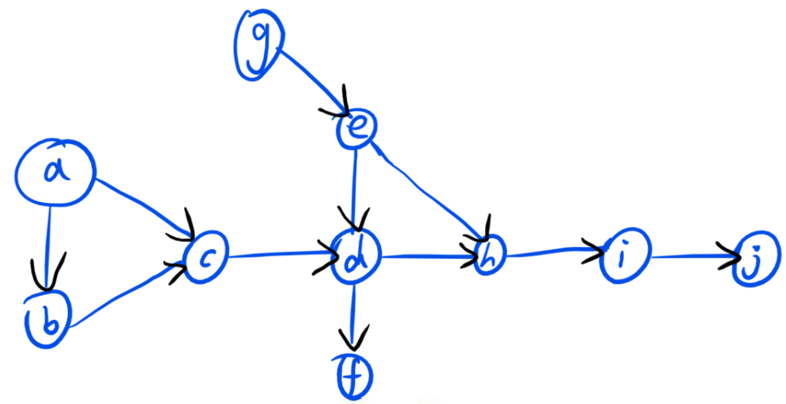
- 只产生一种顶点序列：队列的进入和取出顺序是相同的

- 广度优先遍历的应用
  - 检查连通性
  - 检查无环性
  - 给出两个顶点间边最少的路径

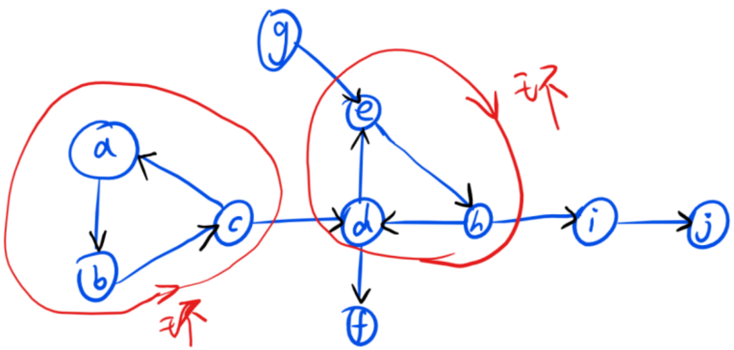
- 问题：迷宫问题使用深度优先遍历还是广度优先遍历？



拓扑排序（1）



- 有向图回顾
  - 所有边都是有方向的图
  - 邻接矩阵不一定表现出对称性
  - 每条边在邻接链表中只有一个对应的节点
- 有向图的DFS森林



a b c g e d h f i j

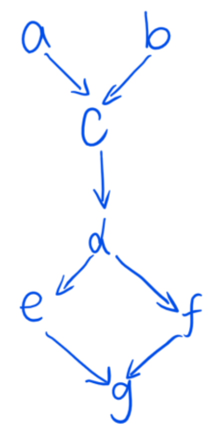
a ~~c~~ b g e ~~h~~ d i j f

c a b c

- 拓扑排序问题
  - 对一个有向无环图，获得一种顶点的序列，使得图中所有边的开始顶点都在结束顶点之前

拓扑排序（2）

- 解法一：DFS
  - 深度优先遍历的时候
    - 出栈是因为节点是端点
    - 所有还在栈内的点必定是其前面的点
    - 其之前的点不在栈内的肯定还没有压栈
    - 出栈的点必定在其之后
    - 出栈的逆序就是一个拓扑排序的解



a→⑥
c→⑤
d→④
e→②
g→①
f→③
b→⑦

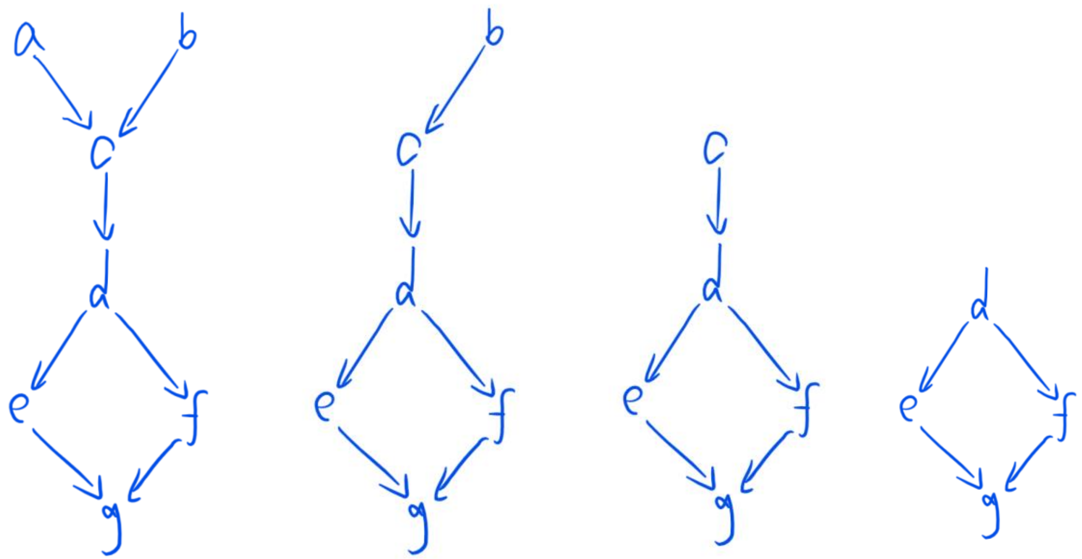
bacdfeg

- 解法二：减一法
  - 规模变为|V|-1个顶点
  - 关键问题：减去哪一个顶点？使得问题变得简单
  - 找到一个序列的问题——即，找到肯定在其它顶点之前的点
  - 减一问题
    - 减去肯定在其它点之前的那个点　问题变为在余下点中找肯定出现在其它点之前的点
    - 问题没有变，规模变小

没有输入的点

拓扑排序（3）

- 减一拓扑排序示例



a,b,c,d,……

- 思考：如何获得所有可能的拓扑排序？

a,b,c,d,e,f,g
a,b,c,d,f,e,g
b,a,c,d,f,e,g
b,a,c,d,e,f,g

生成组合对象的方法（1）

- 组合对象的数量一般是呈指数增长的，有时会更快
- 很多问题成为NP问题的原因就是其解空间的规模是组合对象的个数

- 不确定算法可解的多项式类问题
  - 给出一个解
  - 多项式时间内判断解的正确性

- 如何给出一个解？如何生成组合对象？
- 用于穷举查找

- 排列和子集

生成组合对象的方法（2）——生成排列

- 减一法
  - 生成n-1个数的排列
  - 将第n个数依次插入n-1个数的每一个排列中
  - 缺点
    - 记录所有中间结果，耗费存储空间

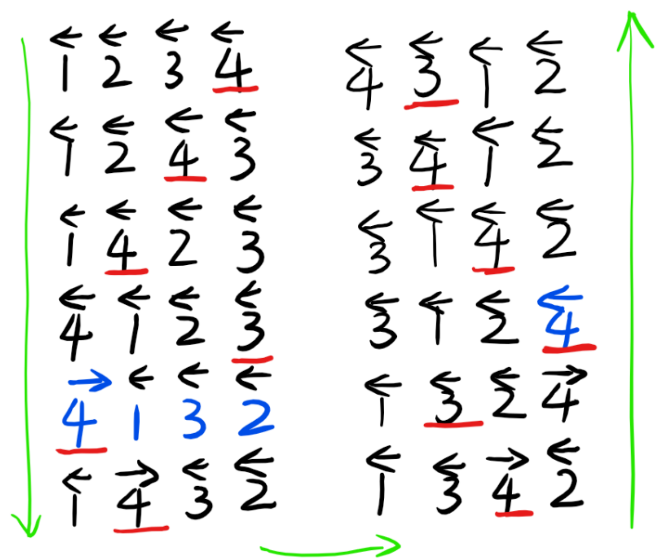
- Johnson-Trotter算法
- 利用一个排列的变换获得所有排列——只需要一个大小为n的数组空间

- 基础定义
  - 序列中的每一个整数都是有方向的
  - 整数指向的方向上相邻元素如果小于当前整数，则该整数称为活动的
  - 1永远都是不活动的
  - n永远都是活动的，除非：n在第一个且指向左；n在最后一个且指向右

生成组合对象的方法（3）——生成排列

- 算法过程
  - 构造第一个排列1,2,3,…,n，每一个数的方向初始化为向左
  - 当当前排列中存在活动整数时
    - 找到最大活动整数m
    - 改变m与其指向相邻元素的位置
    - 改变所有满足p>m的整数p的方向
    - 获得一个新的排列
  - 结束

- 示例



生成组合对象的方法（4）——生成子集

- 减一法
  - 生成集合$\{a_1,\ldots,a_{n-1}\}$的所有子集
  - 在每一个上述子集中加入a_n，得到包含a_n的所有子集
  - 包含a_n的子集和集合$\{a_1,\ldots,a_{n-1}\}$的子集合并为$\{a_1,\ldots,a_n\}$的子集

- $\{a_1,\ldots,a_n\}$的2^n个子集与长度为n的2^n个比特串一一对应

- 是否存在从一个串演变出2^n个所有比特串的方法？
  - 只需要维护一个长度为n的比特串

- 二进制反射格雷码
  - 000，001，011，010，110，111，101，100
  - 如何生成？
  - 思考：生成格雷码的递归算法？

1.初始化
2.改变右边第一个元素
3.改变右边第一个1的左边一个元素
4.重复2、3步直到所有的都生成

减常因子算法（1）

- 一般常因子为2
  - 折半查找
  - 问题：可以折三份查找吗？怎么做？

- 假币问题
  - 只有一个假币，如何快速找出
  - 蛮力法
  - 折半查找；折n份查找

- 俄式乘法
  - 两个数乘法变为：一个数除以2，另一个数乘以2的问题

n为偶数： $n \times m = \frac{n}{2} \times 2m$

n为奇数： $n \times m = \frac{n-1}{2} \times 2m + m$

n	m	
50	65	
25	130	130
12	260	
6	520	
3	1040	1040
1	2080	2080
		3250

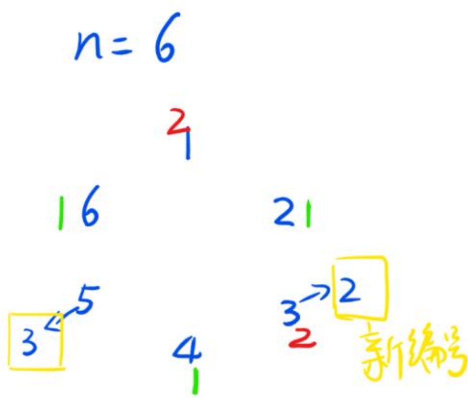
减常因子算法（2）

- 约瑟夫斯问题
  - 约瑟夫斯(Josephus): 约37--100 ,犹太历史学家和军人.原名约瑟夫.本.马赛厄斯.生于耶路撒冷.西元66年在反对罗马的犹太起义中他指挥一支加利利军队.在向罗马人投降时他施展手段获取优待,得以前往罗马,在那里写出几部关于犹太历史和宗教的著作,包括《犹太战争史》(History of the Jewish War,西元75--79年问世)和《犹太古事记》(Antiquities of the Jews,西元93年问世)卒于罗马
  - 约瑟夫斯的队伍在长达47天的殊死搏斗后，终因寡不敌众，加利利陷于罗马人之手。
  - 约瑟夫斯在40名士兵卫护下撤到一个山洞里，士兵们虽然打得精疲力尽，但依然坚贞不屈，他们发誓，决不让罗马人生俘，欲以自杀明志。在此生死攸关的时刻，约瑟夫斯却贪生怕死，惊恐不已。身为指挥官，他不敢拂逆军心提出投降，遂灵机一动，想出一个花招。他诡称自杀之举有违于犹太教的道德规范，如果要杀身成仁，最好的办法是让每个士兵按抽签方法决定顺序，依次由别人动手。他的建议得到士兵的一致赞成。但在抽签时，他略施小技，使自己抽到最后一号。当士兵们按次序一个挨一个魂归西天，只剩下约瑟夫斯和最后一个士兵时，约瑟夫斯先发制人，结果了士兵的性命，自己则跑出山洞向罗马人投降。
  - 其他版本：说服另一个人一起投降

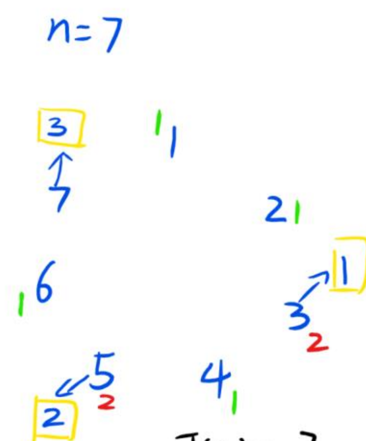
- N个人围成一圈，从1开始编号，然后1、2报数，每次消去报2的人；求最后一个人的编号？
  - 最后一个人的编号为J(n)
  - 递推关系：幸存者在每消去一次后仍然是幸存者；原来规模下他的编号是幸存者，新规模下他的编号变为仍然是幸存者，那么这两个编号之间的关系是什么？直到这个规模变为1，这个递推关系仍然成立

减常因子算法（3）

- J(n)和J(n/2)的关系？
- 区分n为偶数还是奇数



$J(6)=5$, $J(3)=3$



$J(7)=7$ $J(3)=3$

- n为偶数：J(n)=2J(n/2)-1
- n为奇数：J(n)=2J((n-1)/2)+1

- 对n的二进制做一次向左的循环位移得到解

$J(6)=J(110_2)=101_2=5$

$J(7)=J(111_2)=111_2=7$

减可变规模算法（1）

- 欧几里得最大公约数算法

- 计算中值和选择问题
  - 选择问题：求一个n个数列表的第k个最小元素的问题
  - 如果k =「n/2˥，则找到的是中值

  - 可以使用排序：$O(n\log n)$

  全部排序太浪费时间，如何将数据量变小呢？——减治法

  - 借鉴快速排序的做法
    - 对数据进行分区
    - 只处理一个分区

算法复杂度：$C(n) = C(n/2) + (n+1)$

$a=1,\ b=2,\ d=1 \Rightarrow a<b^d \Rightarrow \Theta(n)$

  - 例子：找出下列9个数中第5小的元素：4,1,10,9,7,12,8,2,15

$k=5$

4 | 1 10 9 7 12 8 2 15 ←两个指针，快速交换

中轴

S=3<5 1 2 | 9 | 7 12 8 10 15

中轴

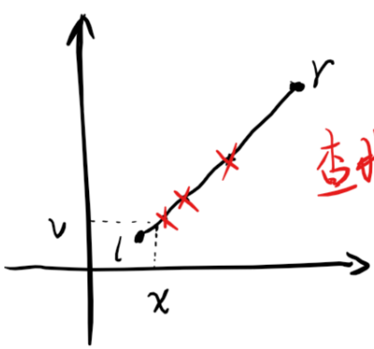
s=6>5 7 8 | 12 10 15

k=5

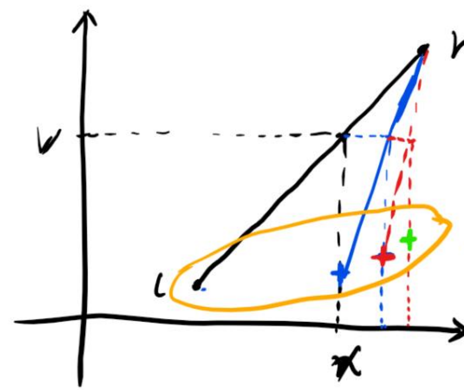
减可变规模算法（2）

- 插值查找：有序数组的查找
  - 折半查找：将数组中间元素与查找键对比——每次减少一半
  - 假设数组值是线性增长的，估计查找键的下标，然后比较——每次减少的是不定的规模

  - 如果是线性的，则一次命中
  - 最坏的情况是什么？



查找v要4次折半
1次插值



多数元素分布在
(l, v)之间
每次只能减少1个元素

- 分布不均匀则效率不一定高

减可变规模算法（3）

- 二叉树的查找和插入
  - 左子树上的值都小于根，右子树的值都大于根
  - 查找与插入类似
  - 平均查找效率是$O(logn)$

- 拈游戏
  - 现有一堆棋子，两个玩家轮流从中拿走1到m个，假设每个玩家都做出了最佳选择，那么哪个玩家能够拿到最后一颗棋子？先走的还是后走的？

  - 这一次如何拿能够让下一个人必输？
  - 保证最后剩下m+1个棋子，则必赢→保证拿完剩下m+1
  - 可以拿的范围是m+2到2m+1，就是说如果剩下2(m+1)那肯定是要输了
  - 结论
    - 在处于x*(m+1)+1到(x+1)*(m+1)-1之间的一个数出手的那个人肯定赢
    - 在y*(m+1)时出手的人肯定输
    - 先出手的人把剩下的构造成(m+1)的倍数就行，即拿走n mod (m+1)；且每次都拿走那么多

- 思考题：m*n方格的巧克力，有一个1*1的方格是坏的，两个人轮流每次顺着边界沿直线掰开，吃掉不包括坏的那半，最后遇到坏的不能再掰的人要吃掉坏的；先掰还是后掰好，怎么掰？

总结

- 减治法
  - 建立原问题与较小规模问题解之间的关系

- 减小规模的角度
  - 减常量：插入排序、深度优先、广度优先遍历
  - 减常因子：折半查找、三份查找
  - 减可变规模：欧几里得最大公约数、选择问题、插值查找

- 拓扑排序的解法
  - 有向无环图的定义

- 下一次课内容：变治法和时空权衡

谢 谢!