NANJING UNIVERSITY

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

# 变治法

南京大学软件学院
李传艺
lcy@nju.edu.cn
费彝民楼917

# 目录

- 变治法回顾
  - 概念
  - 预排序
- 高斯消去法
- 平衡查找树
- 堆和堆排序
- 霍纳法则和二进制幂
- 问题化简

# 变治法回顾

- 变治的三种方式
  - 一个更简单的实例——实例化简——预排序，将问题变为排序好的列表的问题
  - 一个实例的不同表现——改变表现——平衡查找树、堆——改变数据结构
  - 变为另一个问题的实例——问题化简——对NP难问题和NP完全问题的定义

- 预排序
  - 查找问题——折半查找、插值查找
  - 检验元素唯一性
  - 模式计算
  - 计算最近元素
  - 分治法解凸包问题

# 高斯消去法（1）

- 如何解n元一次方程组？
  - 利用一个表达式得到一个变量使用其它变量表达的方式，带入其它方程，消去一个变量
  - 变为n-1元一次方程组——什么设计策略？
  - 递归下去

- 过程太复杂

- 高斯消去法
  - 将方程组系数变为一个下三角全部是0的矩阵

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n
\end{aligned}
\Rightarrow
\begin{aligned}
a'_{11}x_1 + a'_{12}x_2 + \cdots + a'_{1n}x_n &= b'_1 \\
a'_{22}x_2 + \cdots + a'_{2n}x_n &= b'_2 \\
&\vdots \\
a'_{nn}x_n &= b'_n
\end{aligned}
$$

矩阵表达：$Ax = B$

$A'x = B'$

$$
A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}
\quad
B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}
\quad
A' = \begin{bmatrix} a'_{11} & a'_{12} & \cdots & a'_{1n} \\ 0 & a'_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \\ 0 & \cdots & 0 & a'_{nn} \end{bmatrix}
\quad
B' = \begin{bmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_n \end{bmatrix}
$$

# 高斯消去法（2）

- 变换成功后，$x_n$可以得到，然后逆向将其带回变换后的方程组，一一求解
- 化简为一个容易求解的实例

- 此时关键问题：如何变换得到下三角为0的系数矩阵？

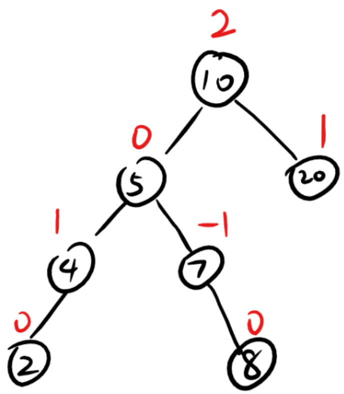
- 通过初等变换——不改变方程组的解
  - 交换方程组中两个方程的位置
  - 把一个方程替换为它的非零倍
  - 把一个方程替换为它和另一个方程倍数之间的和或差

  - 用第一个方程的一个倍数和第二个方程求差，将第二个方程中$x_1$系数变为0；同样与其它方程求差，将所有$x_1$系数变为0；
  - 再用第二个方程与其它方程作同样操作，将所有第二个方程后的所有$x_2$系数变为0；
  - 最终得到下三角为0的系数矩阵

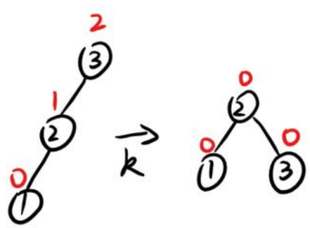
- LU分解
- 计算矩阵的逆

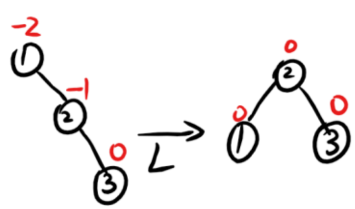
# 平衡查找树（1）

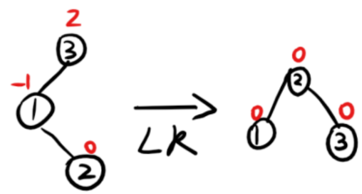
- 二叉查找树
  - 平均情况下，查找、插入和删除时间都是$O(logn)$
  - 最坏情况下是$O(n)$：极度不平衡，高度是n-1

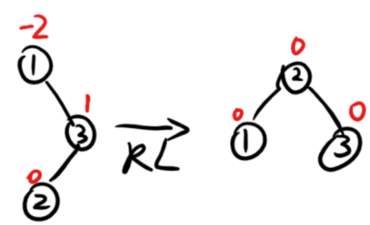


- 两种避免二叉树退化到最差情况的方案　AVL树，红黑树、分裂树
  - 实例化简：将不平衡二叉树转变为平衡的状态，使得问题变得简单
  - 改变表现：允许单个节点中不止包含一个元素　2-3树、3-4树、B树
- AVL树（Adelson-Velsky and Landis Tree）——平衡查找树
  - 每个节点的左右子树高度差不大于1的二叉查找树；左右子树高度差称为**平衡因子**
  - 如果每次查找、插入和删除操作都保证是在AVL树上进行，则最坏情况也是$O(logn)$
  - 如果插入新的元素后发现不再平衡，通过旋转操作变为平衡
    - 右单旋转
    - 左单旋转
    - 左右双转
    - 右左双转



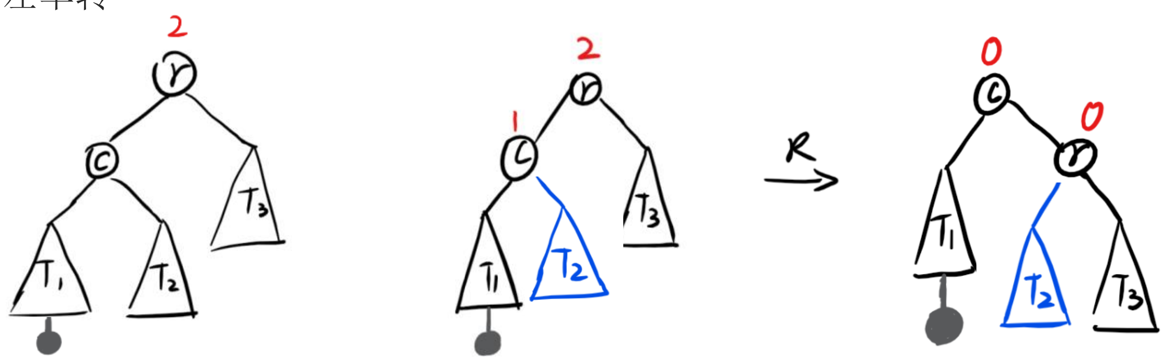



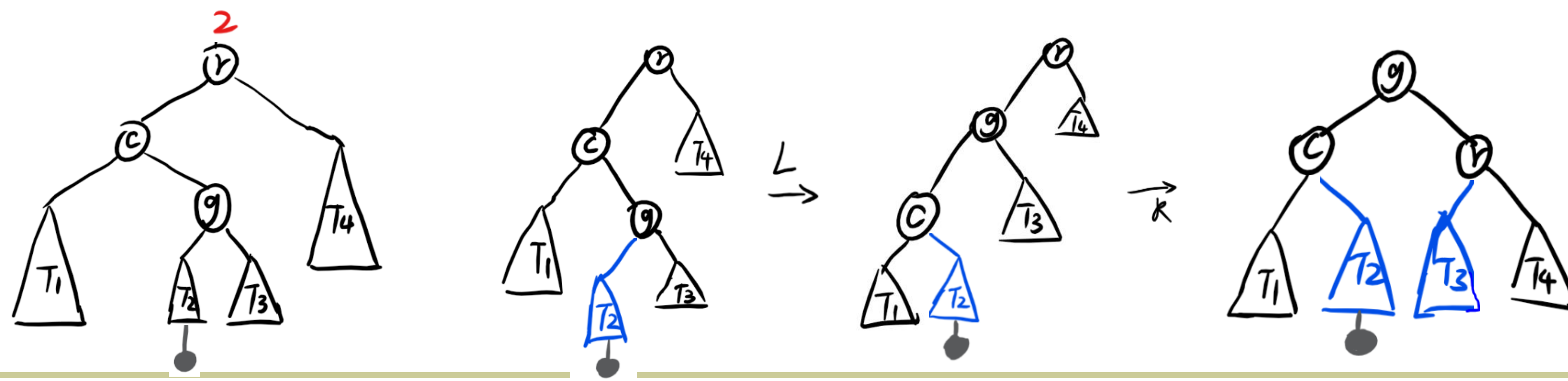



# 平衡查找树（2）

- 一般形式下的各种旋转操作
  - 左单转




  - 左右双转

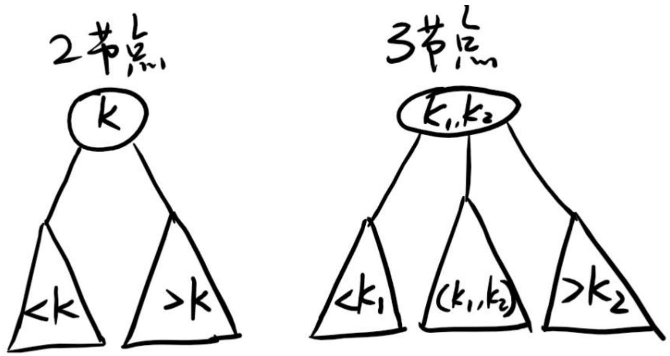


# 平衡查找树（3）

- 通过插入操作构建一个AVL树
  - 例：5，6，8，3，2，4，7
  - 具体过程
    - 搜索插入位置
    - 插入
    - 检查平衡因子，如果不符合则旋转；直到 满足平衡
    - 直到所有元素都插入
- 删除操作
  - 查找到元素
  - 删除节点，检查平衡因子；不满足则旋转直到满足
- 优点：最差情况下，查找、插入和删除操作都是$O(logn)$
- 缺点：每一次插入、删除都需要检查和维护平衡型，操作复杂，阻碍了AVL树成为实现字典的标准结构
- 课后作业：没有实现过AVL树的同学，用任意一种语言实现AVL树的结构和操作

# 平衡查找树（4）

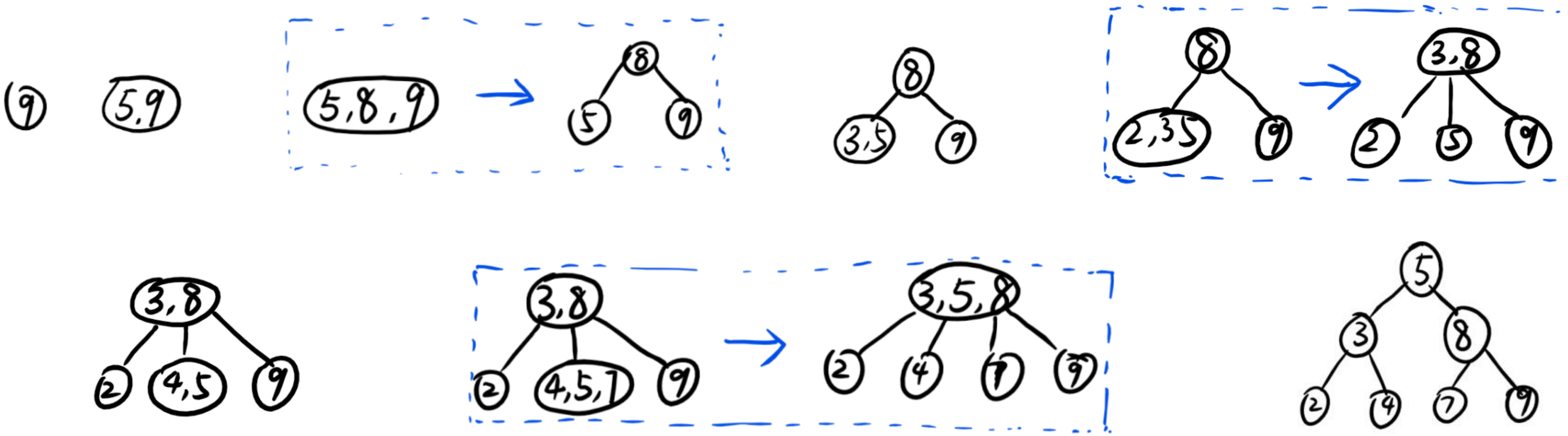
- 2-3树
  - 改变了树的表现形式
  - 允许两种节点：一种有两个子树一个键值的2节点，一种有三个子树两个键值的3节点



  - 所有叶子节点都在同一层：必须是一个绝对平衡的树，每个非叶子节点的平衡因子都是0；每个节点最多可以有2个键值
  - 高为h的2-3树包含的节点数大于等于高度为h的满二叉树的节点数，即至少有2^h-1个节点
- 查找
  - 如果是2节点；如果是3节点；
- 插入
  - 一定插入叶子节点；
  - 如果插入后有两个元素，则完成；
  - 如果插入后有三个则分裂，最小的和最大的放到两个叶子节点，中间的提到父节点里；
  - 直到没有包含三个键值的节点。

# 平衡查找树（5）

- 2-3树例子：构造9，5，8，3，2，4，7数列的2-3树



- 删除
  - 把上面的列表中9删掉，如何操作？
  - 把5删掉如何操作？
  - 比较复杂的过程

  - 课后作业：选择一种语言实现2-3树的结构和操作
- 优点：相对与平衡二叉树可能降低了树的高度，提高操作效率

# 堆和堆排序

- 课后作业：实现堆数据结构
  - root
  - max()
  - get(i)
  - insert(i)
  - remove(i)

# 霍纳法则和二进制幂（1）

- 蛮力求多项式值

$$p(x)=2x^4-x^3-3x^2+x-5$$

4次乘　2次乘　2次乘　4次加减

- 霍纳法则
  - 改变计算方式，减少计算次数

$$= x(2x^3-x^2-3x+1)-5$$

$$= x\cdot(x\cdot(2x^2-x-3)+1)-5$$

$$= \underbrace{x\cdot(\underbrace{x\cdot(\underbrace{x\cdot(\underbrace{2x-1}_{1})-3}_{2})+1}_{3})}_{4}\ \underbrace{-5}_{-5}$$

| | | | | | |
|---|---|---|---|---|---|
| 系数 | 2 | -1 | -3 | +1 | -5 |
| | $(((2x$ | $-1)\cdot x$ | $-3)\cdot x$ | $+1)\cdot x$ | $-5$ |
| $x=3$ | 6 | 5 | 12 | 37 | 76 |

# 霍纳法则和二进制幂（2）

- 单纯计算$a^n$时，霍纳法则有用吗？
- 两种基于改变表现的求幂的算法
- 指数n的数位表示

$$a^n \quad n = p(x) = b_I x^I + \cdots + b_i x^i + \cdots + b_0 = x\cdot(b_I x^{I-1} + \cdots + b_i x^{i-1} + \cdots b_1) + b_0$$

$$= x\cdot(x\cdot(b_I x^{I-2} + \cdots + b_2) + b_1) + b_0$$

$$a^n = a^{p(x)} = a^{x\cdot(x\cdot(\cdots)+b_1)+b_0}$$

$$= a^{x\cdot(x\cdot(\cdots)+b_1)} \cdot a^{b_0}$$

$$\left(a^{x\cdot(\cdots)+b_1}\right)^x \cdot a^{b_0}$$

$$\downarrow$$

$$\left(\left(a^{(\cdots)}\right)^x \cdot a^{b_1}\right)^x \cdot a^{b_0}$$

$$\left(\left(\left(a^{b_I}\right)^x \cdot a^{b_{I-1}}\right)^x \cdot a^{b_{I-2}}\right)^x \cdots\cdots$$

| | | | | |
|---|---|---|---|---|
| $b_i$ | $b_I$ | $b_{I-1}$ | $b_{I-2}$ | $\cdots$ $b_0$ |
| 结果 | $a^{b_I}$ | $\underbrace{(a^{b_I})^x \cdot a^{b_{I-1}}}_{a_1}$ | $\underbrace{(a_1)^x \cdot a^{b_{I-2}}}_{a_2}$ | $\cdots (a_{I-1})^x \cdot a^{b_0}$ |

$x=2 \quad b_i = 0$或$1$

$$a^{13} = a^{1101_2}$$

| $b_i$ | $b_I$ | $b_{I-1}$ | $b_{I-2}$ | $\cdots\cdots$ | $b_0$ |
|---|---|---|---|---|---|
| 结果 | $a^{b_I}$ | $\underbrace{(a^{b_I})^x \cdot a^{b_{I-1}}}_{a_1}$ | $\underbrace{(a_1)^x \cdot a^{b_{I-2}}}_{a_2}$ | $\cdots$ | $(a_{I-1})^x \cdot a^{b_0}$ |

| | | | |
|---|---|---|---|
| $1$ | $1$ | $0$ | $1$ |
| $a$ | $a^2 \cdot a^1 = a^3$ | $(a^3)^2 \cdot a^0 = a^6$ | $(a^6)^2 \cdot a^1 = a^{13}$ |

- 优点：减少乘法次数
- 从左向右计算
- 从右向左计算

| | | | |
|---|---|---|---|
| $1$ | $1$ | $0$ | $1$ |
| $(a^4)^2 \cdot a^5$ | $(a^2)^2 \cdot a$ | $a^2$ | $a$ |
| $a^{13}$ | $a^5$ | | |

# 问题化简（1）

- 把一个要解决的问题化简为一个我们知道如何求解的问题

- 例子
  - 求最小公倍数
  - 计算图中路径数量
  - 优化问题的化简
  - 线性规划
  - 化简为图的问题

- 求最小公倍数
  - 24和60的最小公倍数：lcm(24,60)
  - 24=2*2*2*3; 60=2*2*3*5
  - lcm(24,60) = (2*2*3)*3*5 = 24* 60 / (2*2*3) = 24 * 60 / gcd(24, 60)

- 计算图中路径的数量
  - 计算给定图中两个顶点间路径的数量
  - 如何判断两点之间是否存在路径？
  - 邻接矩阵的幂代表什么？

# 问题化简（2）

- 优化问题的化简
  - 求函数的最小值，理解为求负函数的最大值
  - 拉格朗日乘子法：将带约束的求极值问题转化为简单的求极值问题
    - max f(x,y) subject to g(x,y)=c
    - Max F(x,y,a) = f(x,y) + a*(g(x,y)-c)
  - 求函数极值问题，化简为求导数为0的解方程问题

- 线性规划问题
  - 多变量线性函数的最优化问题，变量满足的约束是以线性等式或者不等式出现的
  - 例：一家具厂生产桌子和椅子，桌子售价50元、椅子售价30元，生产桌子需要木工4小时，油漆工2小时，生产椅子需要木工3小时，油漆工1小时。该厂每月可用木工工时120小时，油漆工工时50小时。问该厂如何生产才能使每月销售收入最大？

| |
|---|
| max: $z=50X_1+30X_2$ |
| s.t. $4X_1+3X_2\le 120$ |
| $2X_1+X_2\le 50$ |
| $X_1\ge 0\quad X_2\ge 0$ |

  - 单纯形法，Karmarkar算法
  - 将问题转化为线性规划问题：背包问题，给定承重为W的背包和n个重量为$w_1,w_2,\ldots,w_n$价值为$v_1,v_2,\ldots v_n$的物品，求物品中价值最大的一个子集，且要能够放到背包中

# 问题化简（3）

- 化简为图问题
  - 很多问题都能够化简为标准的图问题
  - Spark GraphX的应用
    - 推荐技术
    - 分类技术

  - 游戏类题目转化为图问题时，节点表示可能的状态，边则表示状态之间可能的转变：状态空间图

  - 例：一农夫带着一头狼，一只羊和一个白菜过河，小船只能一次装载农夫和一样货物，狼会吃羊，羊会吃白菜，只有农夫在时才安全。现欲让所有物品包括农夫都安全过道河对岸，求最佳答案。

# 总结

- 变治法的三种体现
  - 变为一种简单的实例求相同问题
  - 改变问题的表现形式——数据结构：平衡二叉树、堆
  - 化简问题——转换为一个已知解法的问题

- 平衡二叉树
- 2-3树
- 堆

- 霍纳法则和在二进制幂中的应用

- 下次内容——时空权衡

谢 谢!