

07-03-Algorithm

Created on 20220605.

Last modified on 2023 年 3 月 18 日.

目录

Chapter 1 计算模型合集

各种高效实用的计算模型

Chapter 2 数据结构

2.1 AAA

2.1.1 数组

$a[i] = a + i * \text{len}$; // i from 0; $a[i][j]$ 按行存: $a + (i * \text{len} + j) * \text{len}$; $a[i][j]$ 按列存: $a + (j * \text{len} + i) * \text{len}$;

5*5 的二维数组 a, 各元素 2 字节, $a[2][3]$ 行有限存, 地址? $2*5+3 = 13$, $13*2 = 26$, $a+26$;

2.1.2 线性表、链表

2.1.3 栈

2.1.4 队列

优先队列

2.1.5 哈希表

哈希函数

哈希函数: $y = H(x)$, 输出长度不变; 相同输入每次得到相同输出; 输入差距小也会导致输出差距大, 输入差距大也可能导致输出相同。x 求 y 容易, y 求 x 困难。

MD5, message digest algorithm 5 SHA-1, SHA-2, source hash algorithm. MD5, SHA-1 存在安全隐患。

2.1.6 堆

上浮和下沉, 用于实现 priority queues

2.1.7 树

二叉树

二叉树遍历：前序、后续、中序。反向构造二叉树：利用前 + 中，或后 + 中遍历结果，推出树的结构。只利用前 + 后不行。

树转二叉树：第一个孩子在左，兄弟都是右。

查找二叉树：左 < 根 < 右 1) 左子树的值 < 根的值 < 右子树的值；2) 一直向左达到最小值，一直向右达到最大值；3) 增加节点：从根开始，向末端方向，插入值更小就左转，否则右转，到达末端增加一个叶子节点；4) 删除节点 A：A 的左子树的最大节点替代删除的 A 的位置；5) 扩展：平衡二叉查找树；B 树 (m 个节点的形状平衡的)。

最优二叉树、哈夫曼树：带权路径长度最小。1,2,8,4 构造哈夫曼树：step1: 1,2->3; 3,8,4; step2: 3,4->7;7,8 so: 15 7 8 3 4 1 2 权值： $1*3+2*3+4*2+8*1=25$

线索二叉树：前序、后续、中序，列举各元素后，叶子 LR 指针指向前后元素。

平衡二叉树：任意结点左子树与右子树深度差不大于 1。平衡度 = 左子树深度 - 右子树深度。

2.1.8 图

有向图，无向图。完全图。

存储：邻接矩阵。n 个点， $n*n$ 。i 到 j 有邻接边， $R_{ij}=1$ ，否则为 0。邻接表，V1 --> [2,6,--] --> [4,1,--] --> [6,50,^] // V1 到 2 号结点距离 6，到 4 号结点距离 1，到 6 号结点距离 50

【遍历】深度优先，广度优先。

【拓扑排序，AOV 网络】有向边表示活动之间开始的先后关系。

【图的最小生成树，普里姆算法】留下的权值最小。树没有环路，n 个节点的树边最多 n-1 个。染色红，逐个收集最短的一个元素进来。注意过程中不能形成环。

【图的最小生成树，克鲁斯卡尔算法】从最短的边开始选边。

Chapter 3 随机算法

3.1 随机数生成

3.1.1 Mersenne Twister

梅森旋转 (Mersenne Twister, MT) 算法, 常用是伪随机数生成算法。算法描述如算法 1 所示。

Algorithm 1: Mersenne Twister

Input: the index is noted as x_{in} , the seed number is noted as seed
Output: random number x_{out}

```
1 Initialization:  $[w, n, m, r], a, f, (u, d), (s, b), (t, c), l, MT_0 \leftarrow \text{seed};$   
2 for  $i \leftarrow 1$  to  $n - 1$  do  
3    $MT_i \leftarrow f \cdot \{MT_{i-1} \oplus [MT_{i-1} \gg (w - 2)] + i\}$   
4 end  
5 for  $i \leftarrow 0$  to  $n - 2$  do  
6    $M_c \leftarrow$  the composition of the highest  $w - r$  bits of  $MT_i$  and the lowest  $r$  bits of  
    $MT_{i+1};$   
7    $M_c \leftarrow M_c \gg 1;$   
8   if the lowest bit of  $M_c$  is 1 then  
9      $M_c \leftarrow M_c \oplus a$   
10  end  
11   $MT_i \leftarrow MT_{i+m} \oplus M_c$   
12 end  
13  $x \leftarrow MT_{x_{in}};$   
14  $x_{out} \leftarrow x \oplus [(x \gg u) \& d];$   
15  $x_{out} \leftarrow x \oplus [(x \ll s) \& b];$   
16  $x_{out} \leftarrow x \oplus [(x \ll t) \& c];$   
17  $x_{out} \leftarrow x \oplus (x \gg l);$   
18 return  $x_{out};$ 
```

Chapter 4 算法合集

一般难解问题的高效实用算法

有穷，确定，有效。

【复杂度】时间，空间

时间复杂度：1, $\log_2 n$, n , $n \log_2 n$, n^2 , n^3 , ..., 2^n

4.1 计算几何

4.2 分布式算法

4.3 并行算法

4.4 查找

4.4.1 顺序查找、线性查找

平均查找长度： $\frac{n+1}{2}$ time, $O(N)$

4.4.2 二分查找

有序排列。对于有序数组，每次甩掉一半可能区间。比较次数最多 $\lfloor \log_2 n \rfloor + 1$ time, $O(\log_2 n)$

4.4.3 散列表查找

例如，存储空间 10， $p=5$ ，散列函数 $h = key \% p$ ，存储 3,8,12,17,9：线性探测：3,4，2，5,6
冲突解决：线性探测，伪随机数。

4.5 排序

稳定、不稳定。【一样的数，保持原顺序，叫稳定】

4.5.1 插入式：直接插入

插入： $O(n^2)$ ，认为第 1 个已排序，剩余的依次插入到合适位置。

新的一个与已经排好的比，插入到位置

4.5.2 插入式：希尔

数据少时插入排序效率可以。

例如 10 个元素，先 $d=n/2=5$ ，隔 5 个一组，插入排序； $d = d/2 = 2$ ，取奇数是 3；隔 3 个一组，插入； $d = d/2 = 1$ ，全体插入排序。

4.5.3 选择式：直接选择

选择： $O(n^2)$ ，每次从剩余数组中挑最小的。

每次选剩余最小的。

4.5.4 选择式：堆排序

完全二叉树。堆： $O(n\log n)$ ，构造堆，不断取根-维护堆结构。小顶堆： $k_i \leq k_{2i}, k_i \leq k_{2i+1}$
大顶堆： $k_i \geq k_{2i}, k_i \geq k_{2i+1}$ 所有孩子都更小

从小到大排列：建小顶堆-》取顶-》建小顶堆-》。。。。

例如构造大顶堆：step1，数组顺序构造完全二叉树。step2，最后一个非叶子节点，与其 2 个孩子调整为大顶堆；倒数第 2 个非叶子节点，依次调整。如果有子树，要调整后继续调整子树。

4.5.5 交换式：冒泡

冒泡： $O(n^2)$ ，认为数组是从地板到天花板，每轮都从地板开始冒泡，每轮导致天花板降低；天地相接或某一轮的所有冒泡没有产生相邻的交换，认为排好序。

4.5.6 交换式：快速

快速： $O(n\log n)$ ，递归的分治法每次操作 F 是把当前处理的区间划分为 3 个部分：[小于基准的数区间] 基准数 [大于基准的数区间]。对左右 2 个区间递归执行操作 F。

4.5.7 归并排序

归并： $O(n\log n)$ ，递归的分治法 1) 自顶而下：不断细分，然后归并。归并要保证两帧有序。

4.5.8 基数排序

4.6 图的搜索

4.6.1 广度优先

广度优先：FIFO（先入先出），用队列。

4.6.2 深度优先

深度优先：LIFO（后入先出），用栈。

4.6.3 Bellman-Ford

Bellman-Ford：无向图中最短路径问题。从 A 节点到 B 节点，节点间度量为正。设置初始权重起点为 0，其余点无穷大。广度优先，更新各个节点的权重值，有更小的时更新权重值。

4.6.4 Dijkstra

Dijkstra：无向图中最短路径问题。从 A 节点到 B 节点，节点间度量为正。设置初始权重起点为 0，其余点无穷大。计算与 A 连接是边中最短的节点 K1，然后计算与 K1 连接是边中最短的节点 K2，持续下去直到 B。

4.6.5 A-star

A-star：Dijkstra 并没有一个指向性保证一次性走到终点。增加一种引导，如当前点与终点的估计距离，引导每次对最短节点 K1 的选择。这样的算法称为“启发式算法”。

4.7 安全算法

A 向 B 数据传输 4 个问题：1) 窃听：C 听到了；【加密】2) 假冒：A 或 B 是假的；【消息认证、数字签名】3) 篡改：B 收到的是 C 修改后的 A 发送的消息；【消息认证、数字签名】4) 事后否认：A 事后不承认消息了。【数字签名】数字签名中，为了确认公开密钥的制作者，使用“数字证书”技术。

4.7.1 利用密钥加密

共享密钥加密

【共享密钥加密。对称加密】DES, AES, 凯撒密码, 动态口令。1) 加密和解密用相同的密钥。密钥从 A 发送到 B 的过程中, 也可能被窃听。密钥分配问题。2) 密钥分配问题的解决方案: 密钥交换协议, 公开密钥加密。

公开密钥加密

RSA 算法、椭圆曲线加密算法。1) B 生成公钥 P 和私钥 S, B 把 P 发给 A, A 用 P 加密后把密文发给 B, B 用 S 解密。2) 假设 n 个人互相传输, 需要 $n(n-1)/2$ 对密钥。改进是想 B 对所有的人都保持 1 个 P 和 1 个 S。3) 安全问题: 【中间人攻击】在 B 把 P 发给 A 时, C 截获 P, 把 Q 发给 A, A 用 Q 加密后的密文再次被 C 截获, C 解密后修改, 再用 P 加密传给 B。问题是由于 A 不知道收到的密钥是否来自 B。用数字证书解决。4) 加密解密时间长, 不适合持续发送小数据的情形。用混合加密解决。5) RSA 算法中利用 Miller-Rabin 质数测试。

混合加密

SSL/TLS 协议。数据用共享密钥加密, 发送的密钥信息用公开密钥加密。

Diffie-Hellman 密钥交换

1) 构造密钥合成算法 $F = [P, S]$, 具有特点: 1) 可合成不可分解; 合成后可继续合成; 合成结果与合成顺序无关。2) A 和 B 公开密钥 P, A 准备 SA, A 传输 $[P, SA]$ 给 B; B 准备 SB, A 传输 $[P, SB]$ 给 A; A 和 B 各自组合出 $[P, SA, SB]$ 用于加密和解密。窃听者无法组合出。2) A 和 B 公开大质数 P 和另外一个数 G; A 选一个数 x, A 发送 $(G^x) \bmod P$ 给 B; B 选一个数 y, B 发送 $(G^y) \bmod P$ 给 A; A 和 B 都用密钥 $(G^{xy}) \bmod P$ 。根据质数 P、生成元 G、 $(G^x) \bmod P$, 求 x 的问题, 称为离散对数问题, 该问题至今没找到解法。

4.7.2 消息认证码

认证、检测篡改 A 把密钥 P 安全发给 B; A 用密文和密钥 P 生成消息认证码如 ab12, 称为 MAC(Message authentication code); B 收到密文和 MAC 后, 用密文和密钥生成一份 MAC 和 A 发来的比较是否一样。MAC 算法: HMAC, OMAC, CMAC 缺点: 无法保证密文是 A 生成的还是 B 生成的。问题原因是两方都有相同的密钥, 不能确定 MAC 是谁生成的。解决方案: 数字签名。

4.7.3 数字签名

希望：A 的签名发送给 B，B 可以验证签名，B 不能生成签名。1) 公开密钥加密是 P 加密 S 解密；数字签名是 S 加密 P 解密。2) A 准备发签名，A 准备好 P 和 S；A 用 S 加密得到签名，公开 P。能够用 A 发布的 P 解密的，一定是 A 的 S 加密生成的。3) 求消息的哈希值 X，对 X 加密得到签名。4) 问题：需要知道公钥 P 的制作者，防止 C 用自己的公钥冒充 A 的。解决方法：数字证书。

4.7.4 数字证书

A 和 B 之间的事，找一个双方承认的中间人。A 把要公开的密钥 PA 和自己的个人信息提交给认证中心 (CA, Certification Authority)；CA 确认后利用 CA 的私钥将 PA 和个人信息生成签名作为 A 的证书。B 收到证书后，利用 CA 的公钥 PC 检测证书。1) 问题：检测证书的公钥 PC 是来自 CA 的吗？CA 的 PC 是以数字证书的形式交付的，有更高级别的 CA 署名。2) 根认证中心，其正当性由自身证明，如大型企业。3) 网站的证书称为“服务器证书”，与域名信息对应。可确认域名和存储网站本身的服务器由同一个组织管理。4) PKI, public key infrastructure, 公钥基础设施。

4.8 聚类算法

4.8.1 k-means

随机选择凝聚中心，得到 n 个集合；利用集合重心作为新的凝聚中心，计算新一轮的簇。重复下去得到最终的 n 个集合。

4.8.2 层次聚类

初始时每个对象为 1 类；每次将最近的 2 类合并为 1 类，持续下去。

4.9 其他算法

4.9.1 欧几里得算法（又称辗转相除法）

1) $A = k_1 * \gcd(A, B)$, $B = k_2 * \gcd(A, B)$, A 和 B 看做相同刻度的不同数量的尺子，不断把长的重新赋值为长的减去短的，直到最后剩下长度之比为 1:2，得到了刻度。2) 令 $L_0 > R_0$; $L_1 = R_0$, $R_1 = L_0 \bmod R_0$ ；一直到 $L_k, R_k, R_k = 0, L_k = \gcd(L_0, R_0)$

4.9.2 质数判断

1) 根据定义枚举: 计算 A 的平方根 n , $i:[2,n]$, $r[i] = A \bmod i$; $r[i]$ 中有 0 表示有公因数, 即不是质数。2) 费马测试: 对于质数 p , 任意小于 p 的数 c , 有 $(c^p) \bmod(p) = c$ 。测试 A , 随机找几个小于 A 的数, 判断通过费马测试, 大概率认为是质数。3) 存在满足费马测试的合数, 称为 Carmichael Numbers, 绝对伪质数, 如 561。4) AKS 算法, 多项式时间内进行质数测试。

4.9.3 PageRank

1) 利用网页间的链接关系判断网页的价值。2) A 链接指向 x 个网页, x 个网页评分 A 的权重; A 被 y 个网页指向, A 的评分等于来的各个网页的权重之和; 为了解决循环链接, 引入随机游走, 即有 a 的概率跳到其他的节点, 有 $1-a$ 的概率沿着链接关系走。

4.9.4 汉诺塔问题

递归 1) 移动方法 F 满足: $F(n) = F(F(n-1))$

Chapter 5 算法综合案例

面向应用的大尺度难解问题的工程实用算法

Chapter 6 工程算法集成和相应软件 体系结构

Chapter 7 工程算法分析和评价体系