$f(n) = O(n^{\log_b^{a-\epsilon}})$, then $T(n) = \Theta(n^{\log_b^a})$
$f(n) = \Theta(n^{\log_b^a})$, then $T(n) = \Theta(n^{\log_b^a} \log n)$
$f(n) = \Omega(n^{\log_b^{a+\epsilon}})$ and $\exists c < 1$ s.t. $af(n/b) < cf(n)$ for large n, then $T(n) = \Theta(n^{\log_b^a})$

学号：

正确性：保持某性质，单调能中止

CH1 最大子段和——从后剪负和更新 公共子序列——DP 打表 渐进复杂度——够大，平均（期望复杂度）：独立考察、各种操作概率加权 & 分摊复杂度：连续足够多操作后，总体成本摊还给单次操作

CH2 二分 while(lo<hi){mi=/2; e<S[mi]?hi=mi:lo=mi+1; ret lo-1; 插值最慢 On 字长到 n^1/2 后为 loglogn Bubble 优化：大趟无误停止；大趟中表明 last（最后逆序对可能出现的位置，后面一定有序，hi 更新为 last

Bitmap 掩码位置 0x80>>(k&0x07) **校验环** 等长 intF[] 和一个 stack，使得 stack 添加的时候辅助 F 记录位置，del：交换为栈顶，懒惰删除 Read:k==T[F[k]]

列表去重：相邻对相同删除后者

选择排序稳定性:选择大于等于保证稳定性，后者优先循环节 A[k] A[r(k)] A[r(r(k))]回到 A[k]互不交 不超 n 交换法使 M 脱离原循环节自成且原来减小 1 无需交换次数等效于最初的循环节个数 **插入排序** 就地在线输入敏感 最坏 n^2 [3-10] **归并**排序 适于并行外部逆序对：上界 On^2 冒泡交换逆序减一 归并可计数

游标列表 一组数据等长游标，空数据指向下一个可插入位置，非空指向下一个可删除位置（头方向）

CH4 Stack 尾递归**进制转换**：while{模入栈，数/=进制}

D 括号匹配：规定通用，不能失配，最后为空

E 栈混洗：只允许原入辅助，辅助入目标。本质是等量 PopPush 序列，等价于括号匹配。混洗数卡特兰（S 第一次变空的时候——A 首元素 S 弹出）(2n)!/(n+1)!n!, SPn=sum(1~k~n)（SP（k-1）*SP(n-k)）

充要：没有 312，没有 j+1,i,j，模拟验证：从另一个栈底开始匹配，对上了就 pop，否则 push 成功则对。

F 中缀表达式求值：维护数和符号栈，读符号若读入高 push，栈顶高开算直到栈顶低，相等（右括号或\0）脱括号进入下一字符;左括号全< 右括号全>从来不入栈

G **RPN** 数压入，符直接算 生成：操作数直接接入 RPN，也放到栈中；运算符执行计算时接入 RPN。

J **双栈当队** 裹进两个底对底的栈，enq 压入 R，deq 弹出 F，若无 F 则把所有 R 都装过来。**分摊分析**针对每一个输入对他们的不同状态提出不同的开销，再根据假设情况计算上界最后总开销摊还给这种假设情况 d 次 deq 和 e 次 enq 之后。。势能：原开销+动态势能=定

K **Steap Queap** Def 辅助序列：每一个元素都是原序列中后缀最大者——从尾端扫描"顶"起非降。栈 push 新元素与辅助栈顶较大的。队列 push 要向前撑到够不到为止。优化：只计那几个撑起来的 index

直方图最大矩形 for<n{while(!empty&&H[top]>=H[r]) S.pop(); //until H[top]<H[r] S[r]=empty? 0:1+H[top]; push(r); }pop all..栈顶小于扫描就入栈，栈顶大时就一直 POP+计算更新直到栈顶小。继续扫描。分摊 O(n)?

CH5 A 空树 H 取-1，RB 取 0 **真**:全二度 **完全**只剩最后一层不满 **满**为全满 长子左孩与兄弟右孩—**多叉树**

E 先序：维护栈 while: pop 访问+push rc lc while{访问左藤（loop: visit x+push x->rc+x=x->lc until !x），栈空退出，否则弹出子树循环.}

F 中序：深入左藤（从上到下依次入栈），栈空退出，否则 pop 一个节点 visit 后转向其右子树..节点出栈时左子树不存在或完成，右子树未入栈。是否 O（n）

Succ()：有右孩子，右孩子一路向左深入。否则一路作为右孩子(向左上)找爹，停止后再找一个爹。

G 后序：若栈顶不是 x 的爹，就去找其子树中最左的叶子（while 栈顶 x 非空(if lc，(if rc,push rc) push lc) else(push rc).返回前 pop()）。Pop and visit 每个节点出栈时以它为根的子树已经遍历，且右兄弟如果存在一定在栈顶。正可以遍历右兄弟。验证复杂度?

等价于表达式树的计算~RPN

H 层次遍历：deq visit enq lc rc，其中队列最大规模 n/2 的 ceiling。前面都出 1 入 2.向量紧凑表示完全树

I 重构 中序+先序|后序 or 先序+后序&&真 增强序列 allNULL 输出 **PFC** 文件长度正比于平均带权深度 wald。必有叶子在底两层 不断贪心合并最小树直到只剩一棵 正确：内节点必双子、兄弟可换、层次 归纳 定差

CH6 图 A 简单图：无自环重边 简单路径没有重点带权网络即各边带权重的图

B **邻接矩阵** n*n 记录边 关联矩阵 n*e structV 中有出入度、状态、dftime、遍历树 parent、priority structE 中有 type（undetermined、tree、cross\forward\backward）实现：线性 Vs、双 vec 嵌套构成二维边指针集合。枚举所有邻居 O(n) 邻接表为 Ooutdeg+1 **addedge**: new

姓名：

表并更新关联点的出入度。Deledge 类似 addvertex: Vs push 新、每个已有的外层 vec 都要 push 一个 null 并且加一个整个的外层 vec。删除：del 出边（入度）和那一列 vec，删除顶点后再删除所有的入边（出度）。

性能分析 适稠密图 θn^2 空间 用 vec 后透明地处理空间溢出等 平面图有点-边+区域-连通支=1

C **邻接表** 每个点作表头连接自己所有邻居（所有后继），有向图只记录作为前驱\后继的边，适稀疏

空间 On+e 平面图 On 时间：构造 n+e 枚举所有以 v 为尾（头）的弧 1+degv（n+e，可建立逆邻接表改进）

维护后查 deg+-都是 O1 但 exists 需要搜索 Oe，选用散列可期望 O1 适遍历、顶点数目不定，算 deg 相关

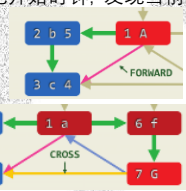
D bfs 树的层次遍历 维护 Queue，每次 dTime=++clock，对于所有邻居开始处理{如果尚未发现则改为发现并加入 Q 中，标记为 Tree 边和 parent，如果已经被发现标记为跨边}标记访问完毕

用于：寻找连通支和可达分量，对连通支的遍历包一层小写的 bfs，reset 并重置 clock，对所有节点做遍历一旦未发现就做 BFS(v,clock)。

复杂度：无向图 reset n+e 每个顶点内枚举邻居需要 1+deg，共需要 n+2e

性质应用：最短路径即为深度 周星驰数

E **dfs** 先序遍历支撑树 DFS{标记开始时钟；发现当前 v 考察每个邻居{未被发现：标



树，递归 DFS 它；dis—指向了祖先，backward；visited—比较发现时间，自己早则 forward 否则 cross} v 已经访问完毕；标记结束时钟}

无向图中只有 TREE 和 BACKWARD 有向图 也包装遍历树所有边取反向 **括号引理**：后代活跃期包含于祖先，无关则不相交，没有别的情况。Backward 一定有回路但！=回路 forward 是我的孩子，cross 和我无关。

有向图环路检测 dfs 欧拉回路拓扑排序双联通分量 dfs

F **拓扑排序** DAG 有向无环图 构造与偏序相容的全序可以拓扑排序一定无环，无环一定可拓扑排序。因为"有向无环图中必有零入度节点不唯一 m"若去掉 m 可排序则加入 m 的排序为 m 在最前 m。

顺序：所有零入度入 Stack，取空 Q，while S 不空{栈顶入 Q，如果 v 邻接 u 入度 1 则入栈，并删除 v 及其关联}最后 G 删空当且仅当成功拓扑排序

逆序：DFS 中每当 visited 加入 S，有 backward not dag。

结束后顺序弹出 S 已经 ftime 由大到小 **CH7APP**

A 双连通分量 关节点删了之后连通分量增多，无关节点为双连通图，极大的双连通子图为双连通分量 BCC 找关节点？—通过 dfs 根至少两子树，内部节点有某个孩子 u 且这颗子树不能由任何 backward 连接到 v 的真祖先（可以 backward 到 v）。这时 BCC 父亲和 BCCu 的关节点。记录 Highest Connected Ancestor：hca(v) = subtree(v) 经过 backward 连接的最高祖先。DFS 中一旦发现 backward v,u，取 hca v=min(hca(v),dtime(u)) 最后标记的是 dtime。如果 dfsu 完成返回 v 时若有 hcau<dtimev 则 hca(v)=min(hca(v),hca(u))。否则 u 子树的最高祖先不超过 v 即可断定 v 是关节点。而 v+子树 u 为一个 BCC

Dfs 找 bcc：额外围护 Stack，将当前 v 入栈。枚举邻居{未发现 Evu—标树边，递归遍历 u、讨论 hca 关系确定关节点后持续弹出 S 直到 u 弹出这些就是一个 BCC；见上}同 O(n+e) 推广到强联通分量？618

B **pfs** 存放顶点并维护 priority(v)越小越先 框架{枚举邻居更新优先级和父节点；从 undis 节点中挑出 pri 最小的点作为下一个 s；标记为访问以及标记好树边（特别的，s 的爹已确定）

C Dijkstra 均正最短路径树!=MST 更新时发现比较路径长度松弛长的（by set new parent）

D prim—MST 允负 0 Cayley 完全图有 n^(n-2)

uv 是该割的一条极短跨边则一定存在包含 uv 的 MST。反证证明。任何 MST 都通过极短跨边连接割。若支撑树添一条边的环路中存在其他边为极长边。则原 T-f+e 为新图的 MST。构造中把当前树看成割。正确性：每一条都属于某一 MST 6-28?

实现：比较老 pri 和权重，后者小则更新 pri 和 parent

E Kruskal MST 每个节点都是一棵树，按照边权排序依次尝试使森林变为一棵树。找 e 来自不同树时可以合并。Proof 引入的每条边都属于某颗 MST e 一定是割

班级：

T 的极短跨边。归纳证明。。

并查集 union-find 问题：维护 group[]所属子集快找慢合并。维护 parent 法：合并时指向新爹即可 680?

F Floydwarshall 允负边但不能负环路。For k for u for v A[u][v]=min(A[u][v],A[u][k]+A[k][v]) **8 二叉搜索树**

A 中序有序 归纳树高可证中序遍历单调

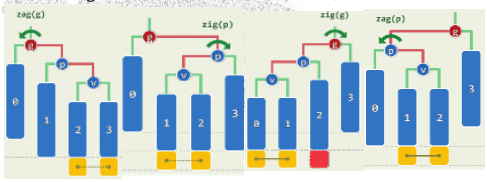
B 查找—返回引用，可能为空节点表示该插入的位置 插入—查找确定位置，插入后向上更新高度 删除—分类删除{单分支时删除自身让孩子替换，双分支与直接后继交换数值！ 删除其直接后继}向上更新高度

C 期望树高 全随机排列的平均高度θlognBST 越低的权重越大 n 个互异节点的不同 BST 共有 catalan(n)棵 =sum(S(k-1)*S(n-k)=(2n)!/(n+1)!n! BST 等概率θsqrtn 理想平衡：n 个节点高度在 $\log_2 n$ 渐进平衡—渐进不超过 Ologn 即为 BBST 等价 BST 可 On 互转

D AVL 树渐进平衡 定义平衡因子=H(lc)-H(rc)<=1 h 高度至少包含 S(h)=fib(h+3)-1 个节点此为最瘦的情况 S(h)=1+S(h-1)+S(h-2)配成 fib $(1)a_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$

插入可能从祖父向上一整路失衡，找到最低失衡节点(一边 updH)和儿孙考虑。同向一次 zigzag 用 34 实现，异向两次等价于 34 重构。高度复原，全树复衡。

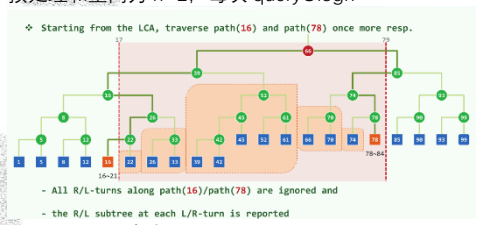
删除：单旋、双旋情况可能祖先继续失衡高度减小，可能需要 logn 次调整、可能全树高度下降 kunth: 0.21 次



1D 查区间 二分查右边界后向前检查+输出 Or+logn

2D：利用容斥原理 n 方时间的预处理，矩阵每个点中包含其左下的点的个数，答案正对角线-反对角线

预处理和空间为 n^2，每次 queryOlogn



B **kdtree**1D—完全二叉 each key=Minkey in Rsubtree，即右边区间的左端点。仅最底两层的叶节点是 1D 点集

2D 每个 region 左下开右上闭 等价于四叉树划分

递归建树 P：如果只剩一个，创造叶节点返回；Root=createKdNode();splitDirection=even(d)? ver|hori; root->splitline=findmedian(direction,P);

分隔开子点集，并左右递归建树 所属 region 往左下移

Query：如果是叶区间，判断是否在；左右分别{整个 region 都在搜索中，report，否则如果有交集则继续递归搜索}

复杂度：预处理 Onlogn 储存 On 查询 Or+sqrtn Q(1)=O1; Q(n)=2+2Q(n/4) 节点的四个孙子不超过两个会继续递归。更高维度？O(r+n^(1-1/d))

C MultiLevelSearchTree 按维度搜索范围，最坏——x 搜出了所有点但 y 拒绝了所有点 On

嵌套树实现 O(r+(logn)^2) 二维建树时空 Onlogn

二维以上建树储存都是 nlog（d-1）n 查询为 r+logdn

A 逐层**伸展** 对 v 的父亲做 zigzag，最坏时分摊Ωn

双层伸展 向上追溯两层， 在双 zig 或 zag 时先转祖父再转父亲。使最坏情况的单条路径也能收缩一半。

【8-2】分摊 Ologn 每次旋转之后都要对三代 updH

搜索：标准搜索+splay+返回根

插入：搜索后决定在根的左上还是右上插入新根

删除：搜索后删除，在 R 里面做失败查找 Rmin 来当根

评价：无需高度和平衡，分摊 Ologn 近 AVL 且反复顺序访问子集分摊仅常数

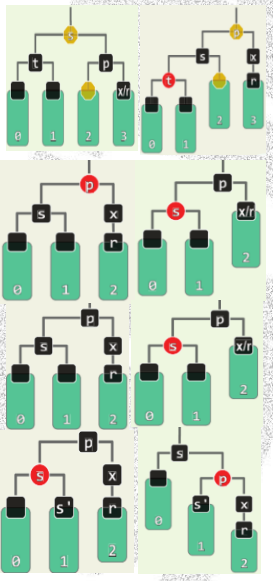
分摊分析：定义势能φT=sum(log(sizev))平衡小倾侧大 单链 logn!=Onlogn 满树 On 不妨仅考察 search 连续 m>>n 次访问，记 Ak=Tk+△φk

则显然 T=A-△φ在 A+-Onlogn 区间内若能证明

A=Omlogn 则一定有 T=Omlogn
讨论 A 在单 zig/zag 中 Ai<1+rankiV-randi-1V

$$A_i^{(k)} = T_i^{(k)} + \Delta\Phi(S_i^{(k)}) = 2 + \Delta rank_i(g) + \Delta rank_i(p) + \Delta rank_i(v)$$
$$= 2 + [rank_i(g) - \underline{rank_{i-1}(g)}] + [rank_i(p) - \underline{rank_{i-1}(p)}] + [\underline{rank_i(v)} - rank_{i-1}(v)]$$
$$< 2 + rank_i(g) + \underline{rank_i(p)} - 2\cdot rank_{i-1}(v) \quad (\because rank_{i-1}(p) > rank_{i-1}(v))$$
$$< 2 + rank_i(g) + rank_i(v) - 2\cdot rank_{i-1}(v) \quad (\because rank_i(p) < rank_i(v))$$
$$< 3\cdot(rank_i(v) - rank_{i-1}(v)) \quad (\because \frac{\log G_i + \log V_{i-1}}{2} \le \log\frac{G_i + V_{i-1}}{2} < \log\frac{V_i}{2})$$
$$A_i^{(k)} = T_i^{(k)} + \Delta\Phi(S_i^{(k)}) = 2 + \Delta rank_i(g) + \Delta rank_i(p) + \Delta rank_i(v)$$
$$= 2 + [rank_i(g) - \underline{rank_{i-1}(g)}] + [rank_i(p) - \underline{rank_{i-1}(p)}] + [\underline{rank_i(v)} - rank_{i-1}(v)]$$
$$< 2 + \underline{rank_i(g) + rank_i(p)} - 2\cdot rank_{i-1}(v) \quad (\because rank_{i-1}(p) > rank_{i-1}(v))$$
$$< 2 + 2\cdot rank_i(v) - 2 - 2\cdot rank_{i-1}(v) \quad (\because \frac{\log G_i + \log P_i}{2} \le \log\frac{G_i + P_i}{2} < \log\frac{V_i}{2})$$
$$= 2\cdot(rank_i(v) - rank_{i-1}(v))$$

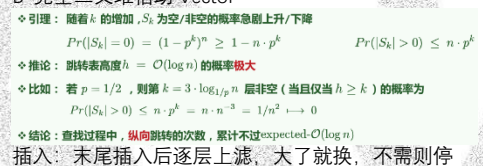
B BTree d 代合并 m=2^d 出路和 2^d-1 个关键码 多级存储中减少外部查找的 IO，批量访问一组关键码
外部节点深度相等为树高，叶节点为 h-1，内部节点关键码 <=m-1&& 分支出路：ceiling m/2<=n+1<=m（root>=2）成为(m/2，m)-树 典型(2,4)-树
内部都是 vec 装关键码和孩子
查找 vec，不成功标记_hot，v=v->child[r+1]每一深度最多一次 IO 运行时间 Ologn
插入上溢 在_hot 中插关键码和空指针，检查上溢：上溢节点中位数归入父节点的子树位置并分裂指针分别指向中位数左边右边，仍然可继续分裂。。到根仍上溢则提升中位数为新根，是 B 树增高唯一途径 O（h）
删除下溢：如果不是叶子，在右子树中找直接后继交换 1.若有左兄弟且够大，则 p 节点分界 key 作为最小，左兄弟最小作为 p 节点分界。2.右兄弟借父亲旋转修复 3.左右兄弟必有其一但不够用则把父亲的分解 key 降下来粘接自己与兄弟。但父亲仍继续下溢，不超过 Oh Max Height 时内部节点尽量瘦 **logmN+1 <= h<= 1+log (ceiling(m/2), grounding ((N+1)/2))**
C **红黑树** 减少结构变化，相邻版本差不超过 O1
1 树根和外节点为黑 2.红节点只有黑孩子父亲 3.外部节点黑深度相等 即根的黑高度 等价于 2,4 B 树证明其平衡 h<=R+H<=2H 黑高度上界 log2,n+1 故 Ologn
插入：红节点黑高度 0 修双红{爹是红则一定有黑祖父，1.叔父黑，则局部 34 重构+染色 RBR 2.叔父红，pu 转黑（BH++）g 转红。分裂节点 g 上移但 g 仍可能双红若已经抵达树根则强行转黑、黑高度++}
删除：仅分析实际删除的，单分支被接替的情况。删除者和接替者一个红时，接替者染黑即可。
删除者接替者都黑需修正 如果删除根则新根置黑，更新高度即可。若父亲平衡则好，替代节点为红染黑即可，接替和删除均黑（此时接替者为 null）{接替者父亲 p 和兄弟 s
BB-1 兄弟黑且至少有一个红孩子 t{ s 继承 pcolor, tp to black}等效于关键码旋转 消除下溢 转下 p 转上 s
BB2-R s 黑且两个孩子都黑 p 红 下溢节点与兄弟合并顺便向父节点借了一个红色
BB2-B s 黑两个孩子都黑 p 黑 合并兄弟和空空的自己和爹，父节点变空继续上溢 solve（p）
BB-3S 红孩子必黑 绕 p 单旋继续 solve（r）下一步必恢复 1~2 旋转 3 染色/2 染色/1 染色上升一层/1 转 2 染色



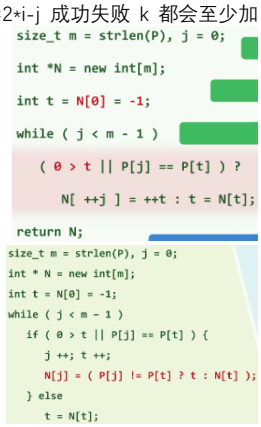
A **hash**() key->&entry 期望 O1 不是 O1
B 散列函数 确定快速(期望 O1)满射均匀少聚集取质

$$S(n) = \sum_{k=1}^{h} k\cdot 2^{h-k} = \sum_{k=1}^{h}\sum_{i=1}^{k} 2^{h-k} = \sum_{i=1}^{h}\sum_{k=i}^{h} 2^{h-k}$$
$$= \sum_{i=1}^{h}\sum_{k=0}^{h-i} 2^k = \sum_{i=1}^{h}\{2^{h-i+1}-1\} = \sum_{i=1}^{h} 2^{h-i+1} - h$$
$$= \sum_{i=1}^{h} 2^i - h = 2^{h+1} - 2 - h = \mathcal{O}(n)$$

数模：不动点 0，相关性(相邻映射也相邻) MAD (ax+b)%primeM 还有数字分析、平方取中间几位，折叠取和、按位异或、伪随机数、多项式{循环移位异或之后相加，最后模表长}
C 冲突排解 开放散列 多槽位、独立链空间不连续无法缓存、公共溢出区处理冲突正比于溢出区规模
封闭散列+开放定址可出借
线性试探：非同义词也会冲突，数据堆积严重、局部性良好。标记以懒惰删除—后查找时为不匹配的非空桶，插入时为空桶
平方试探：表长素数且装填因子小于 0.5 一定能找到出空桶——n 方%M 取值为 ceiling(M/2)且前面就能取遍
双向平方试探 表长取 4n+3 素数就不重不漏
素数 p 表示为平方和等价 4k+1 自然数表示成平方和等价于每一个 4k+3 素因子都是偶次方
再散列：约定冲突后再 hash 一次为每次的偏移增量
重散列：懒惰标记过多或者装填因子太大
桶排序 初始化置 0，扫描标记 Om、顺序输出 On。允许重复时桶里维护链表和表长空间变 Om+n
N 个互异点在实轴上的最大缝隙？1.找出最左最有并划分 n-1 个桶 2.模余归入对应的桶并在每个桶中动态记录最左最右点（可相同可没有） 3.计算相邻非空桶的距离 Proof Maxgap 至少跨过两桶 但 mingap?
基数排序 从低位开始分别桶排序 Proof 会保持前 i-1 趟的顺序复杂度为 O(t*(m+n))m 为取值范围，t 为数域位数 且有稳定性。。二进制时 1 扔到尾巴 0 不管
整数排序 n 进制 d 位 比如六十四位整数的 n>2^16 即可 在 0~n^d 中 n 个数转换进制后排序时间为 Odn 转换进制的复杂度。。？
计数排序 大数据小取值——已经按数字排序的牌再按花色排序：归到四个花色桶中，得到各个桶内的最大 rank（向后累加桶内个数）。从后向前扫描原序列并相应桶的--最大 rank 即为位置。。
跳转表 期望塔高 2 解决空间 On
查找：每一层向后到更大的 key or 溢出，命中则返回否则向下一层继续。。纵向跳转不超过期望 **Ologn**
横向跳转一定抵达塔顶 Thm Ey=(1-p)/p=1 次，同层不超过两次，每次查找都在期望 2h 即 Ologn 内
插入：查找位置（雷同降落）插入底层后后扔硬币长高 删除时从高到低拆塔
A 支持取删插(sorted) vector list BBST 都可实现
B 完全二叉堆借助 Vector



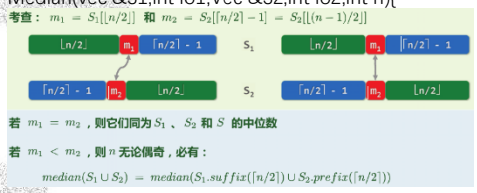
插入：末尾插入后逐层上滤，大了就换，不需则停
删除：删头补脚后逐层下滤，找堪为父者交换或停止
建堆：按个上滤时最坏 Onlogn->自下而上的下滤 Floyd 从 n/2-1 开始向上的下滤
C **堆排序** 改进 selecttion 的发动机选最大值，就地算法{建堆后 swap 头元素和 elem[n--]，下滤堆头}
D **胜者树** 叶节点待排序内部为胜者 createOn removeOlogn insertOlogn 树根是全局冠军。用于排序时建堆后排一个数就下去删除然后向上重赛。。Onlogn 选取最小的 k 个 n+klogn
败者树：根的父节点是冠军，重构时和败者比赢得向上输的留下。构造时相当于 n 次重构
E **优先级队列** 堆装满顶点 delmax nlogn increase 更新关联顶点距离需要 elogn**d 又** par:k-1/d，child=kd+i 上滤 logdn 下滤 dlogdn 运行时间为 ndlogdn+elogdn 取 d=e/n+2 时达到 O(elog((e/n+2),n))自适应达到最优左式堆后 O(e+nlogn) 插入合并 inc 接口分摊 O1
F **左式堆**(knuth 修订命名) 沿藤合并 基于二叉树
引入 all 外部节点 定义 NPL 为到外节点的**最近距离**且规定 npl 左孩子>=npl 右孩子 自己 npl=1+npl(rc)
根右侧链(长度 d)的终点为最浅外部节点，存在以 r 为根高度为 d 的满子树 至少有 2^d-1 个内部节点和 2^(d+1)-1 个节点 反之 n 节点左式堆一定小于 grounding（log(2n+1)）=Ologn
合并：{把较大的堆的右子堆与另一堆合并后连接到较大的堆顶，并根据 npl 可能交换左右子堆，更新 npl1116 插入 merge 原树和新节点 删除 merge 左右子树
B 暴力 i(原串) j 都要回退
C **KMP**(Knuth+Morris+Parrt) i 永不回退 失败时 j 变小并继续，记录 j 回到的 next 表：前缀=后缀的最大前缀右点 while(j<m&&i<n){ if (j<0||T[i]==p[j]) i,j++ else j=next[j]} **分摊分析**：K=2*i-j 成功失败 k 都会至少加 1，结束时 k<=2n+1
Next 表：失败前的后缀=原串前缀 len
mamammia-10012310 -1 0 -1 1 0 -1 3 1 0
Boyer & Moore 算法
BC 从末字符开始比对，BC 怀字符，遍历 Pstr 标记字符的最右出现
偏好概率小 P 长字母表大 **策略**移动到那里/没出现整个移过去/出现太靠右移动一个 **最好** n/m **最坏** n*m worst: 100in000000?



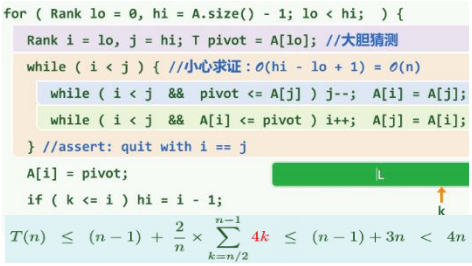
Gs 表好后缀 最好 n/m 最差 n+m 改良 kmp 反过来找 P[j:m]=P[k:m-j]把 j-k 记录在 j 里
构造：MS/ss=P[0:j]的某个后缀==P 的某个后缀<=j+1 Ssj=j+1: gs[i]=m-j-1 任 i<m-j-1/gs[m-ss[j]-1]=m-j-1 ms[n-1] = n; int lower = n-1, upper = n-1;
KarpRabin d 进制数编号 匹配则一定子串相等 将指纹用散列压缩（相邻两次散列，O1 内获得下一个指纹）哈希值相等后仍需精确比对
A **快排** mi=Partition(lo,hi),递归快排两边
LUG 版 随机选一个轴点 swap 到 lo 并备份，向内交替移动 hi lo 把有效的指向空的后变为空的。最后轴点归位 期望 logn 取消递归时维护栈小贪心，大任务优先入栈。保证递归深度不超过 logn **落在λn 试试**，一条递归路径最多 log(2/(1+λ),n)个准居中，λ>1/3 时有 1-1/pown 的概率深度不超 3log3/2,n **比较次数** 1.386
LUG'解决重复，if(lo<hi)elem[lo++]=elem[hi];

$$T(n) = (n-1) + \frac{1}{n}\cdot\sum_{k=0}^{n-1}[T(k)+T(n-k-1)] = (n-1) + \frac{2}{n}\cdot\sum_{k=0}^{n-1} T(k)$$
$$\frac{T(n)}{n+1} = \frac{T(n)}{n+1} - \frac{T(1)}{2} = 4\cdot\sum_{k=2}^{n}\frac{1}{k+1} - 2\cdot\sum_{k=2}^{n}\frac{1}{k} = 2\cdot\sum_{k=1}^{n+1}\frac{1}{k} + \frac{2}{n+1} - 4 \approx 2\cdot\ln n$$

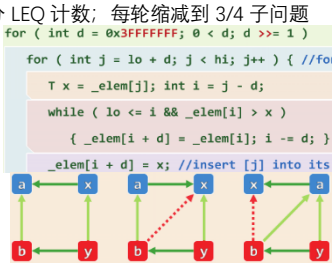
DUP'版 松 LUG'的交换条件，多交换轴点居中 DUP
LGU 稳定：从第二个往后如果小于轴点就与++mi 交换
众数 for{ifc==0 maj=A[i];c=1;else maj==A[i]?c++:c--}
Median(vec &S1,int lo1,Vec &S2,int lo2,int n){



If(n<3)return trivialmedian(~~~~~)
Int mi1=lo1+n/2;mi2=lo2+(n-1)/2;
If(S1[mi1]<S2[mi2])return median(mi1,lo2,n+lo1-mi1)
Else if{} else return S1[mi1] 任意子向量 Ologmin(n1,n2)
第 k+1 小 选 pivot 进行 LUG 后缩小范围继续猜



Linearselect:Q 是小常量，如果序列小于 Q 为 trivial，划分子序列；各自 On*n 排序找到中位数；递归找到全局中位数；划分 LEQ 计数；每轮缩减到 3/4 子问题
希尔排序 矩阵变窄，每次对各列排序。对希尔序列的最坏情况 6 7 8 52 3 1 4 g 有序在 h 排序后仍 g 有序当 gh 互质时，



最小组不出来为 gh-g+h 则 gh 有序后其线性组合依然有序 gh 互质时 for all elems: i-j>x(g,h) only if S[j]<=S[i] 不超过 n*x(g,h)个 inversions