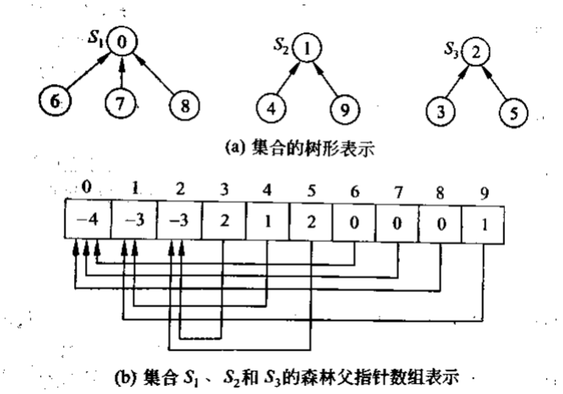
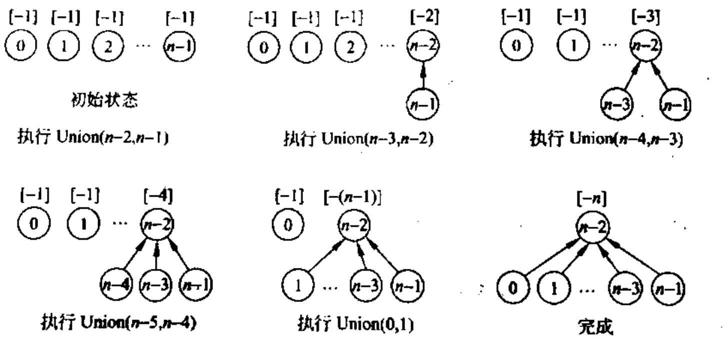
数据结构期末复习

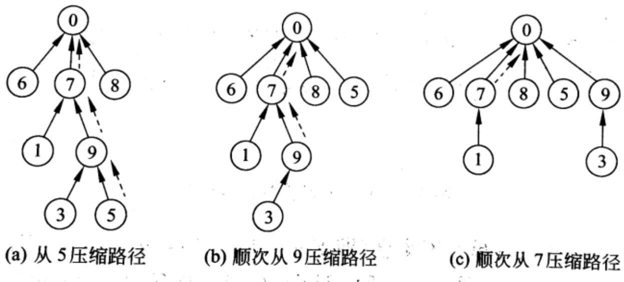
1. 集合的两种表示：树形表示和父指针数组表示

   a)
2. 并查集 WeightedUnion 过程（节点个数多者作为双亲）

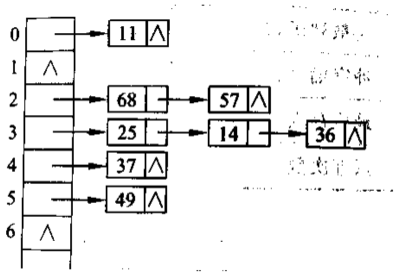
   a)
3. 并查集 CollapsingFind 过程

   a)
4. 衡量散列方法的搜索性能
   a) ASL(succ)：搜索到表中已有表项的平均探查次数
   b) ASL(unsucc)：搜索不到表中表项，但找到插入位置的平均探查次数
   c) 注意，如果散列表大小为 12，采用的散列函数为 x%11，那也意味着最多只能插入前 11 个表项，第 12 个表项始终为空，计算 ASL(unsucc)时多项式只能有 11 项相加，分母只能是 11，而不能是 12（因为 12 始终不会被搜索到！始终不会作为可能被探查的位置！）。亦即 unsucc 计算时分母只能是取余号后的数字！
5. 线性探查法找下一个位置公式：$(H_0 + \mathrm{i})\%\mathrm{m}$，即此时把数组当成循环数组使用
6. 散列表若经常变动，使用开散列方法处理冲突，因为闭散列方法在删除时只能逻辑删除表中元素，容易造成大量空间被浪费
7. 二次探查法找下一个位置公式：$(H_0 \pm i^2)\%\mathrm{m}, \mathrm{i} = 1,2,3,\dots,(\mathrm{m}-1)/2$，先加后减，序列应该固定，散列表大小 m 必须是满足 4k+3 的质数
8. 闭散列线性探查法结果绘图举例：

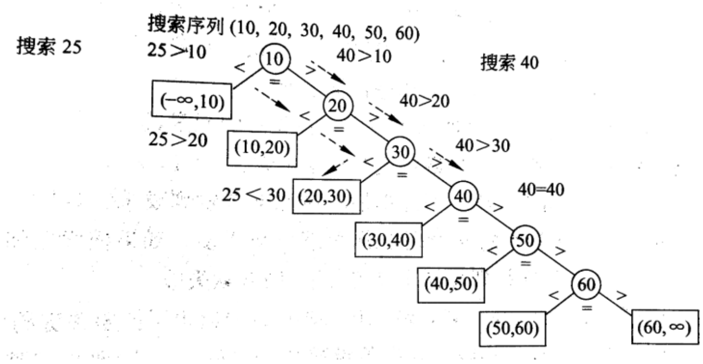
0	1	2	3	4	5	6	7	8	9	10	11
11		68	25	37	14	36	49	57			
(1)		(1)	(1)	(1)	(3)	(4)	(3)	(7)			

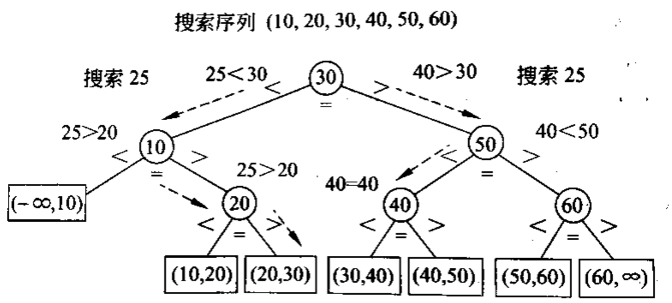
9. 开散列法得到的链表结构绘图举例：



   a)

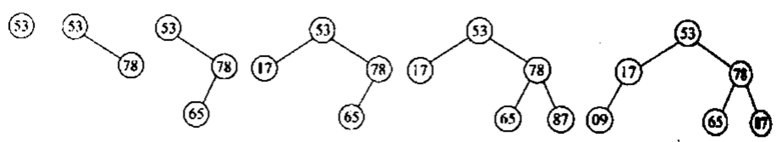
10. 开散列法比闭散列法节省空间，除留取余法作散列函数更好
11. 有序顺序表的判定树
   a) 顺序搜索：

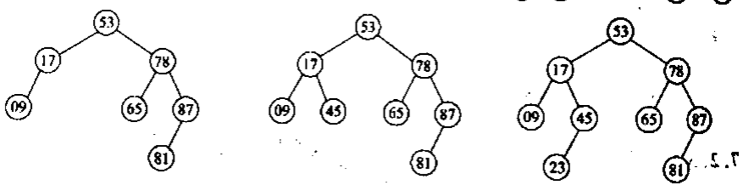




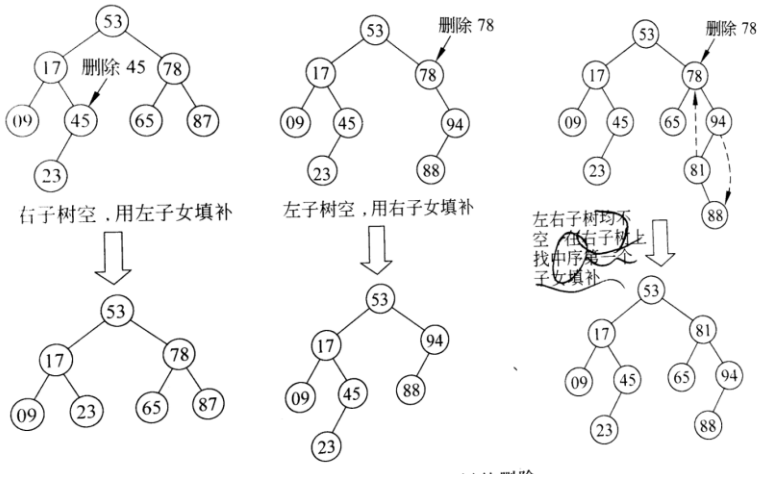
   b) 折半搜索：

12. 有序顺序表顺序搜索和折半搜索的 ASL_{succ} 和 ASL_{unsucc} 计算？ 【请补充】
13. n 个结点的二叉树个数/n 个数进栈，出栈序列个数？（卡特兰数）$\frac{1}{n+1} * C_{2n}^{n}$
14. 二叉搜索树的插入、删除、查找？
   a) 插入（插入只能插在叶节点！）





   b)
   c) 删除（三种情况都要会）



d)

e) 查找

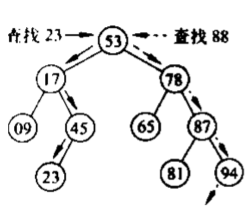


图 7.7 二叉搜索树的搜索

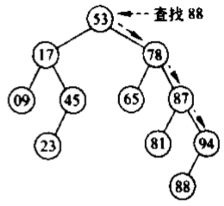
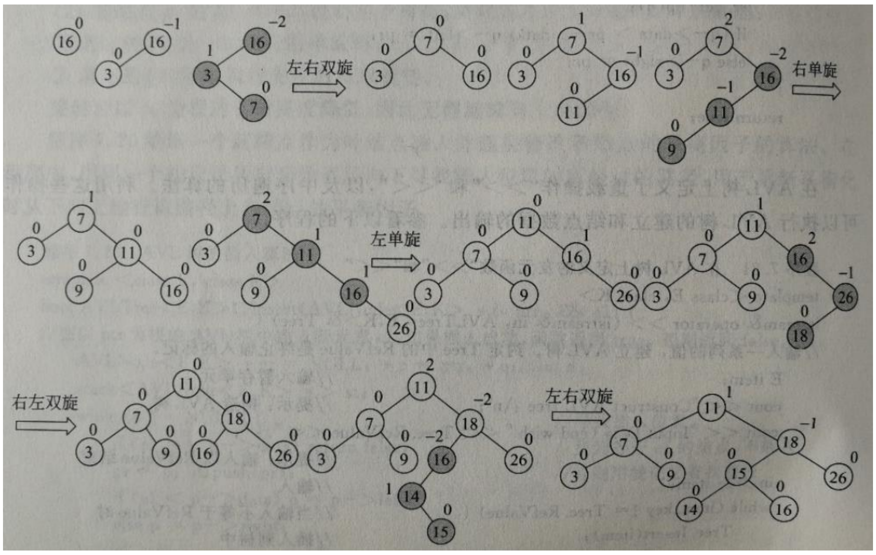


图 7.8 插入新结点 88

f)

g) 注意删除结点，若左右子树均非空，则选择该节点的右子树的中序遍历下第一个结点填补这个位置，这样仍然可以保持树的结构和搜索性能

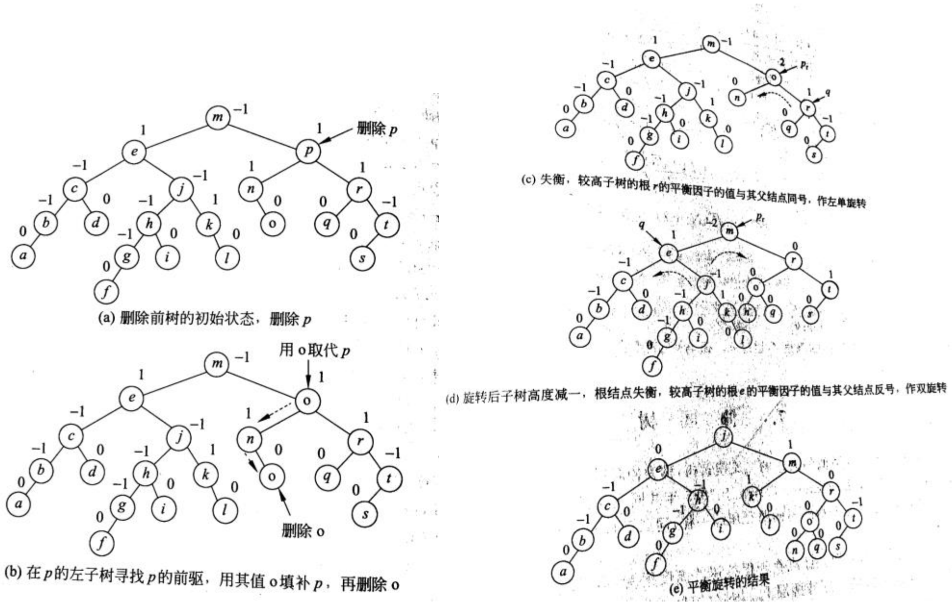
15. AVL 树（重点！）
16. 平衡因子：右子树-左子树
17. 平衡化旋转：插入结点到根回溯，如果有平衡因子绝对值>1 情况出现，则该节点及路径下两个结点进行平衡化旋转，直到路径上每个节点平衡因子<=1
18. AVL 树的插入：

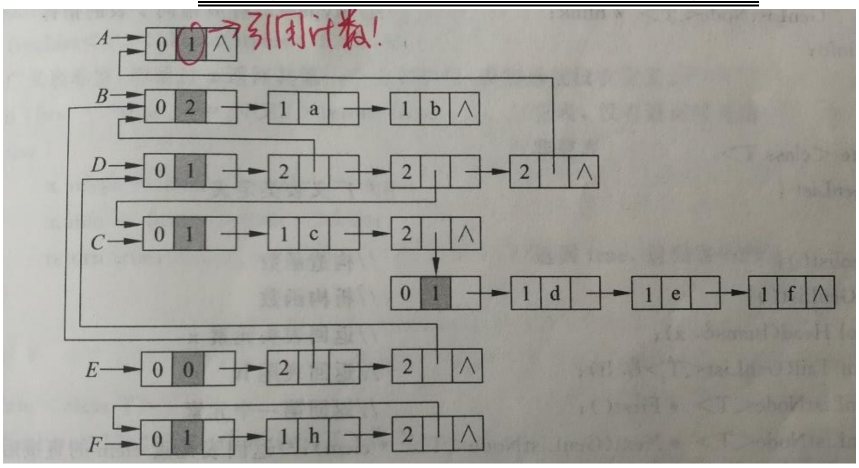


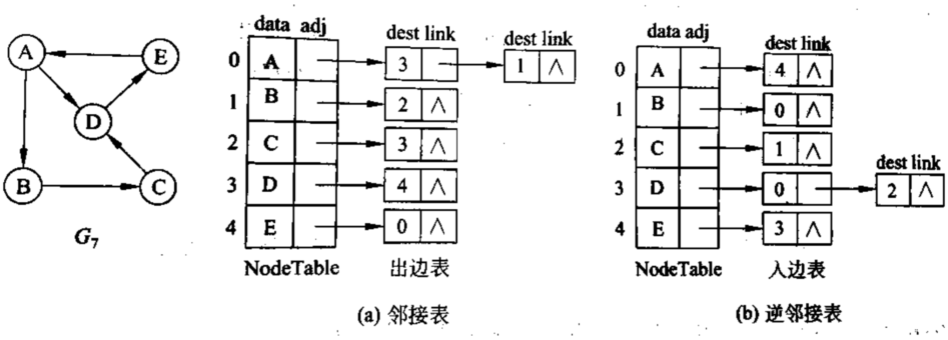
a)

19. AVL 树的删除
    a) 如果有失衡，那么考察失衡结点 p_r 和较高子树的根 q，异号双旋，否则单旋（含 0）
    b) 如果被删除结点有两个子女，则找中序下的直接前驱取代被删除结点的位置（为什

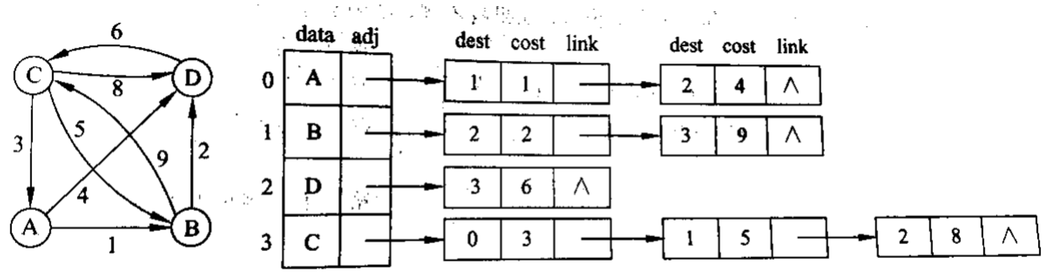
么？因为这样效率更高，更快一点!）

c) 

20. 广义表的存储表示，<u>引用计数不要再漏掉了！子表也有引用计数！</u>
    a) 

    b) utype=0（表头）/1（元素）/2（子表）；广义表（包括子表）都需要一个附加头结点
    c) ref（引用数）/value（值）/hlink（头指针）；
    d) tlink（尾指针）
21. 图的存储结构
    a) 邻接矩阵：行表示出，列表示入
    b) 邻接表：
        i. 有向图的邻接表和逆邻接表表示：
        ii. 

iii. 带权（网络）的邻接表表示：（多了一个 cost 域代表边的权重）



G_9 NodeTable Edge Linked List （出边表）

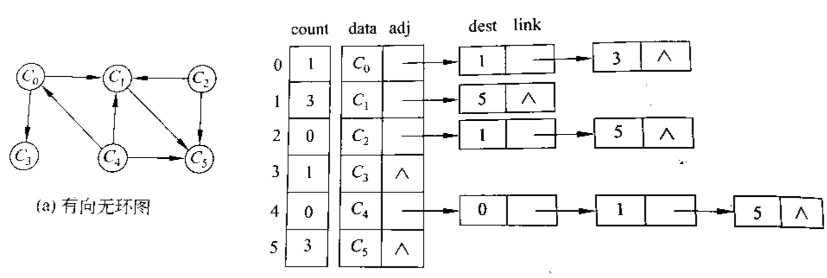
iv.

22. 求最小生成树的两个算法
    a) Kruskal 算法：每次选所有边的最小权边
    b) Prim 算法：每次选当前已经连好的连通分量的最小割
23. 求非负权值的单源最短路的算法
    a) Dijkstra 算法：S[]、dist[]、path[]辅助数组的变化？【请补充例题】

表 8.2 Dijkstra 算法中各辅助数组的变化

选取终点	顶点 1			顶点 2			顶点 3			顶点 4		
	S[1]	dist[1]	path[1]	S[2]	dist[2]	path[2]	S[3]	dist[3]	path[3]	S[4]	dist[4]	path[4]
初始	0	10	0	0	∞	−1	0	30	0	0	100	0
1	1	10	0	0	60	1	0	30	0	0	100	0
3	1	10	0	0	50	3	1	30	0	0	90	3
2	1	10	0	1	50	3	1	30	0	0	60	2
4	1	10	0	1	50	3	1	30	0	1	60	2

    b)
24. AOV 拓扑有序序列的输出：每次有新入度=0 的顶点就进栈，再把入度=0 的顶点弹栈
    a) 进栈操作：count[w]=top;top=w;//top 指向新栈顶，原栈顶元素放在 count[w]
    b) 弹栈操作：v=top;top=count[top];//栈顶位置送 v，top 退到次栈顶
    c) count 数组即可作为栈使用，减小空间复杂度
    d) 图和对应的邻接表（注意点：count 数组也要绘制出来！）



(a) 有向无环图

(b) NodeTable

    e)
    f) 栈顶指针变化：注意-1、还有拓扑排序终止后又重新初始化栈顶-1 的问题！还 count 数组元素意思有时候是指向次栈顶的位置，要记得改变！如本题中的“2”指向 count[2]!！，不能错了

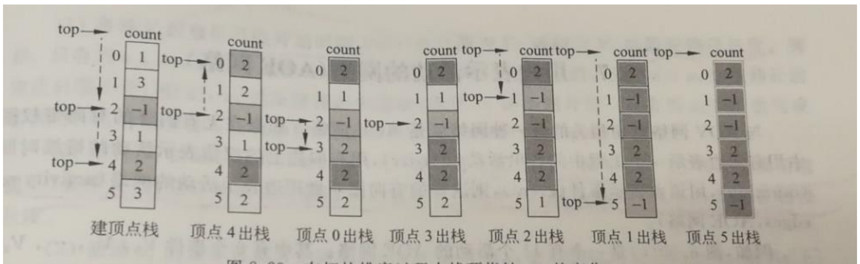
g) 

图 8.29 在拓扑排序过程中栈顶指针 top 的变化

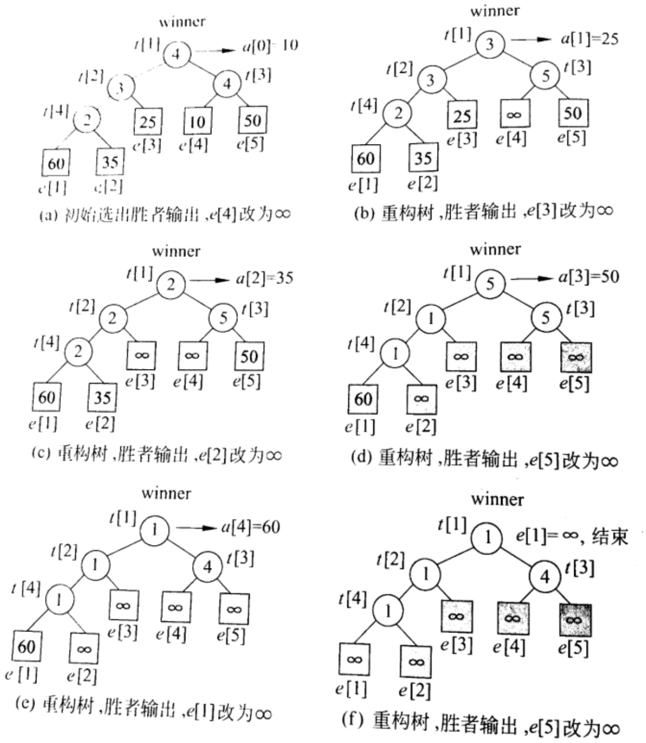
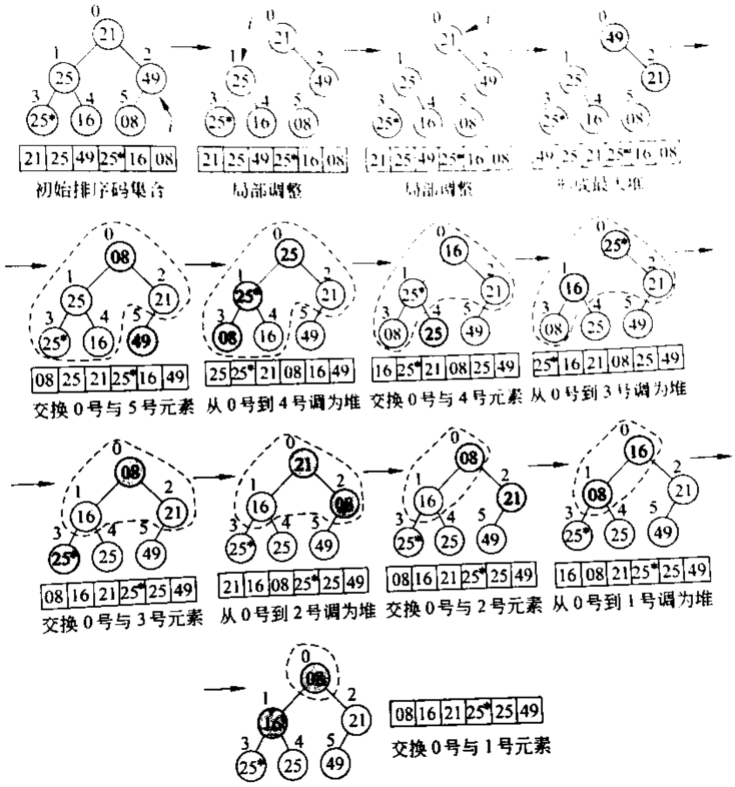
25. AOE 网络
    a) 事件 Vi 的最早可能开始时间：Ve[i]=源点->Vi 的最长路径长度
    b) 事件 Vi 的最迟允许开始时间：Vl[i]=终点减去 Vi->终点的最长路径长度
    c) 活动 ak:<Vi,Vj>的最早可能开始时间：源点->Vi 的最长路径长度，Ae[k]=Ve[i]
    d) 活动 ak:<Vi,Vj>的最迟允许开始事件：Al[k]=Vl[j]-dur(<i,j>)
    e) 关键活动：组成关键路径，Al[k]=Ae[k]！是活动的最早和最迟相同！
26. 排序（具体看练习！）
    a) 稳定的直接插入排序：大循环从前往后，temp 从后往前比，直到找到自己的插入位置
    b) 稳定的希尔排序：依次做增量为 gap 的直接插入排序，最后有序
    c) 稳定的冒泡排序：大循环从前往后走，小循环从后往前换，依次交换元素至最终有序
    d) 不稳定的快速排序：pivot 基准元素选取，划分，递归
    e) 不稳定的选择排序：选当前序列最值，放到头头，以此类推
    f) 稳定的锦标赛排序：可以用完全二叉树定义，如图所示为一颗胜者树的排序过程
       i. 

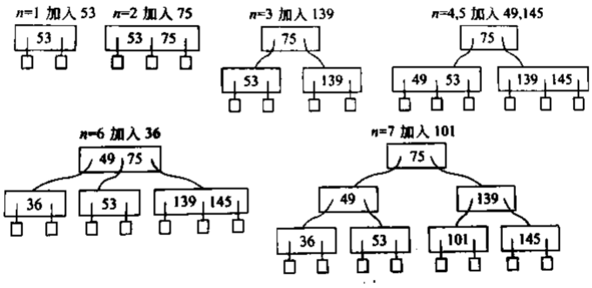
图 9.16 利用胜者树进行排序
    g) 不稳定的堆排序：如果要非降序排列：建最大堆，交换堆顶和堆尾元素，然后调整回排除堆尾元素的最大堆，以此类推



i.

h) 稳定的归并排序：划分划分，归并的时候对两个序列添加指针往后走，将更小的填入数组当中，归并后最终得到的就是有序的序列了

27. 索引顺序文件搜索：一般两级搜索，
    a) 确定满足 ID[i-1].max_key < K ≤ ID[i].max_key 后根据指针指向地址查对应子表
    b) 搜索成功的平均搜索长度=ASL_{Index} + $ASL_{SubList}$
28. 动态的 m 路搜索树：
    a) 结构：n, P_0, (K_1,P_1), (K_2, P_2), ..., (K_n, P_n)，其中 Pi 的 0≤i≤n < m，Ki 的 1≤i≤n < m，n 即为当前节点存在的关键码的个数且 n_{max}=m-1
    b) 举例【请补充】
    c) m 路搜索树最大结点个数：$\frac{1}{m-1}(m^h-1)$
    d) 高度为 h 的 m 路搜索树关键码个数在 h 和m^h-1之间
    e) 一棵有 n 个关键码的 m 路搜索树高度为$\log_m(n+1)$到 n 之间
29. B-tree
    a) 高度 h≤$\log_{\lceil m/2\rceil}\left(\frac{N+1}{2}\right)+1$，例如阶数 m=199，关键码总数 N=1,999,999，B 树高度不超过$\log_{100}1000000+1=4$
    b) 反之，N $\geq 2*\left\lceil\frac{m}{2}\right\rceil^{h-1}-1$，例如阶数 m=4，高度 h=3，则关键码总数至少为$2*2^2-1=7$
    c) B 树的搜索：一旦找到就停止；区分 B+树随机搜索：即使非叶节点找到，也要一直走到叶节点才可以停止
    d) B 树的插入：如果超出 B 树的当前结点关键码个数，则分裂左右，中间的$\lceil m/2\rceil$关键码则放到父节点当中（自底向上分裂节点）记住一定要分裂结点!!!

i. 

e) B 树的删除：精髓：左右兄弟和父亲进行填补，删非叶节点则拿该结点指示的子树中的最小关键码 x 来代替被删除的位置，然后删除关键码 x

30. B+树
    a) 本书：最大关键码复写原则写非叶节点
    b) B+树的插入：一定插在叶节点，然后向上更新非叶节点，结点分裂时左边$\left\lceil\frac{m+1}{2}\right\rceil$个元素，右边$\left\lfloor\frac{m+1}{2}\right\rfloor$个元素
    c) B+树的删除：精髓：删除时本叶未达$\left\lceil\frac{m}{2}\right\rceil$，则只删除叶，索引项不管；本叶达$\left\lceil\frac{m}{2}\right\rceil$，右叶不达，右叶调入本叶并更新索引项；本叶右叶均达$\left\lceil\frac{m}{2}\right\rceil$，合并二者，调父节点（无论如何调，只要分解关键码不会导致搜索问题就可以不用改它）
    d) B+树的叶节点最好标上指针
31. 注意区分“逻辑结构”与“存储结构”的区别！
32. 二叉搜索树删除有左右子女结点的结点：找中序下右子树第一个子女填补
33. AVL 树删除有左右子女结点的结点：找中序下直接前驱