|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 学号： |  | 姓名： |  | | 班级： | |  |
| Name | 插入 | 删除 | | 寻找m次 | | 说明 | |
| AVL | O(1)次调整 | O(logn)次调整,平均0.28 | | O(mlogn) | | 平衡树 | |
| SPLAY | O(n)次调整 | O(n)次调整 | | O(mlogk + nlogn) | | 非平衡树 | |
| 红黑 | 双红修正 1次重构，O(log n)染色 | 双黑修正，1重构1单旋O(logn)重染色 | |  | | 平衡树 | |
| B-tree | 次分裂 | 次缝合 | | ,高度只有倍 | | | |
| Kd-tree | 构建，用quickSelect |  | |  | | O(n) 存储空间 | |
| 锦标赛 | 胜者树根是第一，败者树根是第二(更高效) 排序渐进上等于小顶堆，但常数大 | | | | | | |

文本

描述已自动生成文本

中度可信度描述已自动生成

|  |  |
| --- | --- |
| Tarjan | 有向图强连通分量算法、splay树 |
| Knuth | 自后向前交换shuffle法、栈混洗没有312形、AVL平均旋转次数、KMP、希尔排序H-g sorted |
| Pratt | KMP、希尔排序Pratt序列 |

1. 图片包含 图形用户界面

   描述已自动生成**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*树**
2. 若AVL树插入元素的过程中发生了旋转操作，则树高必不变。
3. 在某节点被删除后AVL树高即便下降了，这次操作期间也未必做过旋转调整。
4. 将 {0,1,\*,2018}插入一棵空的伸展树后若树高为2018，则上述词条未必是按单调次序插入的。
5. 若调用remove将节点x删除，则所需的时间为被删除之前树的高度。
6. 胜者树需要在重赛过程中反复将节点与其兄弟进行比较，败者树不需要。
7. 将N个关键码按随机次序插入B树，则期望的分裂次数为O(log^2 N)。(连续分裂h次才能涨，O(h^2))
8. B树的任一非叶节点内，每个关键码都存在直接后继，且必然来自某个叶节点。
9. 图示

   描述已自动生成BTree:solveOverflow()和BTree:solveUnderflow()在最坏情况下均需下界(logn)的时间，然而在B-树任一足够长的生命期内，就分摊意义而言二者都仅需要O(1)时间。
10. 在任何情况下，伸展树总能保持每次操作O(logn)的均摊复杂度。(而不是平均复杂度)
11. 最底局的叶节点一旦被访问(并做splay)之后，伸展树的高度未必然随即下降。
12. 在kd-search中，查找区间R与任一节点的4个**孙节点**(假设存在)对应区域最多有两个相交。（\sqrt{n} 来历）
13. 在BST中删除两个节点，无论先删除哪个节点，最终BST的拓扑结构均相同。
14. 由同一组共n个词条构成的任意两棵BST，经2n-2次zig或zag旋转之后，必定可以相互转换。
15. 文本

    描述已自动生成若红黑树插入一个元素后**黑高度**增加，则双红修正过程中没有拓扑结构变换，只有重染色操作。 （u为红）
16. 文本

    描述已自动生成如果元素理想随机，那么对二叉搜索树做平衡化处理，对改进其**渐进时间复杂度**没有什么作用。
17. Kd-tree 、interval、segment树空间复杂度是线性，2Drangetree是O(n logn) 空间。
18. 右左子树规模之比x的最大值X：左式堆 < AVL树 < 红黑树。（红黑树条件宽松，可能碰到大数）
19. 由5个互异节点构成的不同的BST共有 42种。(catalan:1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796)
20. **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*图**
21. 在DFS算法中的default分支，将 dTime(v) < dTime(u) 改为 dTime(v) < fTime(u) 可行。
22. 因同一顶点的邻居被枚举的次序不同，同一有向图G所对应的DFS 森林未必唯一。然而只要起始于G中某顶点s的**某次DFS**所生成的是一棵树， 则起始于s的**任何一次DFS**都将生成一棵树。
23. 文本

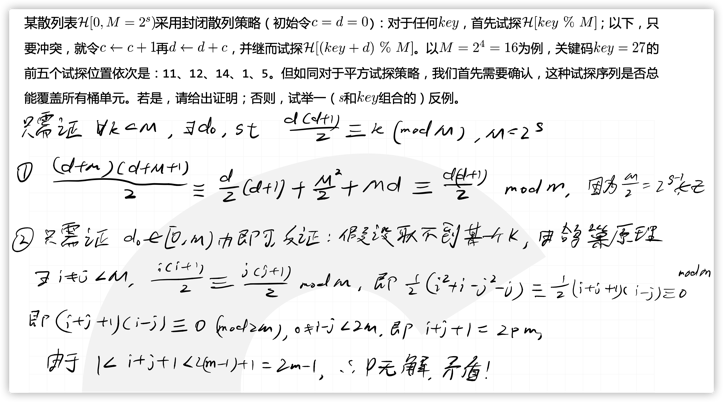
    描述已自动生成设在有向图G中，存在一条自顶点v通往u的路径。于是，若在某次DFS中有 dTime(v) < dTime(u)，则这次DFS所生成的DFS森林中，v不一定是u的祖先。
24. 由于起点和边选取顺序不同，DFS森林的树边、向前边、向后边、跨边数量可能都不同。
25. 有向图经DFS后若共有 k 条边被标记为 backward+forward+cross，则它应恰有k个独立环路。
26. 对于同一无向图，起始于顶点s的DFS尽管可能得到结构不同的树，但s在树中的度数必然固定。
27. 在图的优先级搜索中，每次可能调用多次prioUpdater，但累计调用次数仍为O(e)。
28. 如果把朋友圈视为一无向图，那么即使A君看不到你给B君点的赞，你们仍可能属于同一个双连通分量。
29. **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*串**
30. 若KMP算法不使用改进版的next表，最坏情况下时间复杂度可能达到O(m+n)。
31. 相对于KMP算法而言，BM算法更适合于大字符集的应用场合。
32. 图片包含 游戏机, 铅笔, 汽车, 充满

    描述已自动生成借助BC表GS表，BM算法在最好情况下都需要O(m+n)\O(n/m) 时间。
33. 在BM算法中，对于任一模式串P，0 < gs(j) <= j 对于**每个**0 <= j < |P| 都成立。
34. 随机生成的二进制串，gs表中gs[0] = 1的概率为 1/ 2^(m-1)
35. 对小写字母集的串匹配，KMP算法与蛮力算法在最好和平均情况下渐进时间复杂度相同。
36. **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*堆**
37. 完全二叉堆删除元素在最坏情况下时间复杂度为O(logn)，平均情况为O(logn)。 (上滤平均O(1))
38. 图片包含 信件

    描述已自动生成有n个节点的左式堆，左子堆最小规模为1。
39. 在使用Heapify批量建堆的过程中，改变同层节点的下滤次序对算法的正确性和**时间数常数**无影响。
40. 与二叉堆相比，多叉堆delMax()操作时间复杂度高。但3叉堆比二叉堆更快。e叉堆最快（
41. 用完全二叉堆来实现PFS，则各顶点在出堆之前，深度只可能逐步减少或持平而不致增加。
42. **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*散列**
43. 采用双向平方试探策略时，将散列表长度取作素数M=4k+3，可以杜绝位置冲突。
44. 如果元素理想随机，那么对二叉搜索树做平衡化处理，对改进其**渐进时间复杂度**没有什么作⽤用。
45. 若元素理想随机，则用除余法作为散列函数时，即使区间长度不是素数，也不会影响数据的均匀性。
46. 采用单向平方策略的散列表，只要长度M不是素数，则每一组同义词在表中都不会超过[M/2]个。错
47. 图片包含 图表

    描述已自动生成将n个词条逐个插入一个容量为M、采用线性试探策略、初始为空的散列表，n < M，则无论它们的插入次序如何，最终的平均成功查找长度都必然一样。
48. **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*排序**
49. 若序列中逆序对个数为O(n^2)，则使用快速排序进行的交换次数为 O(n)。(可构造)
50. 经快速划分( LGU版)之后，后缀G中的雷同元素可能调换相对次序，其余部分的雷同元素也会。
51. 希尔排序每按照某个增量做过逐列排序，序列中逆序对的总数都会减少或持平，不会增加。
52. 只要底层的排序算法是正确且稳定的，则 radixSort基数排序也必然是正确且稳定的。
53. 采用任何一种增量序列，希尔排序1-sorting不一定只需要O(n)时间。(2^n肯定不行)
54. 表格

    描述已自动生成文本, 信件

    描述已自动生成在n个节点的跳转表中，塔高的平均值为2。

文本

描述已自动生成图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成文本

描述已自动生成手机屏幕的截图

中度可信度描述已自动生成图示

描述已自动生成文本

描述已自动生成图片包含 图示

描述已自动生成图片包含 图示

描述已自动生成图片包含 图示

描述已自动生成图示

描述已自动生成图示

描述已自动生成