

- ☐ 在某节点被删除后AVL树的高度即便下降了, 这次操作期间也未必做过旋转调整。
- ☐
- ☐ 有向图经DFS后若共有 k 条边被标记为BACKWARD, 则它应恰有 k 个环路。
- ☐
- ☐ 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DFS树, 但 s 在树中的度数必然固定。
- ☐
- ☐ 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 $\lfloor M/2 \rfloor$ 个。

-
- ☐
- ☐ PFS过程中, 尽管每一步迭代都可能多次调用prioUpdater(), 但累计不过 $\mathcal{O}(e)$ 次。
- ☐
- ☐ 相对于KMP算法而言, BM算法更适用于大字符集的应用场合。
- ☐
- ☐ 在存有 n 个词条的跳转表中, 各塔高度的期望值为 $\Theta(\log n)$ 。
- ☐

-
- ☐ 红黑树的插入或删除操作, 都有可能导致 $\Omega(\log n)$ 个节点的颜色反转。
- ☐
- ☐ 将 $\{0, 1, 2, \dots, 2018\}$ 插入一棵空的伸展树后若树高为2018, 则上述词条必是按单调次序插入的。
- ☐
- ☐ 在插入操作后若红黑树黑高度增加, 则在双红修复过程中仅做过重染色, 而无任何结构调整。
- ☐
- ☐ 若输入序列包含 $\Omega(n^2)$ 个逆序对, 则快速排序算法 (LUG版) 至少需要执行 $\Omega(n \log n)$ 元素交换操作。

-
- ☐
- ☐ 采用12-C节中介绍的任何一种增量序列, shellSort()最后的1-sorting都只需要 $\mathcal{O}(n)$ 时间。
- ☐
- ☐ 无论是单独借助BC[]表或GS[]表, BM算法在最好情况下都只需要 $\mathcal{O}(|T|/|P|) = \mathcal{O}(n/m)$ 时间。
- ☐
- ☐ 对规模为 n 的AVL树做一次插入操作, 最坏情况下可能引发 $\Omega(\log n)$ 次局部重构。
- ☐

