《数据分析与算法设计》

竝

1、前中后序遍历

preorder：根左右

inorder：左根右

postorder：左右根

2、DFS/BFS

dfs：递归

bfs：队列

3、排序/查找效率分析

待排序数据基本有序时，快排慢，冒泡/插入排序效率较高

堆排序适合记录数较大的文件

n较小时，冒泡、插入、选择。

n较大时，快排、归并、堆排。

排序算法	时间复杂度			空间复杂度	是否稳定
	最好情况	平均情况	最差情况		
直接插入排序	O(n)	O(n^2)	O(n^2)	O(1)	是
冒泡排序	O(n)	O(n^2)	O(n^2)	O(1)	是
简单选择排序	O(n^2)	O(n^2)	O(n^2)	O(1)	否
希尔排序				O(1)	否
快速排序	O(nlogn)	O(nlogn)	O(n^2)	O(logn)	否
堆排序	O(nlogn)	O(nlogn)	O(nlogn)	O(1)	否
二路归并排序	O(nlogn)	O(nlogn)	O(nlogn)	O(n)	是

折半查找效率 $log_2(n+1)$

4、AVL树/2-3树

AVL左旋右旋

树型	判定条件	调整方法
LL	BF(root) = 2, BF(root->lchild) = 1	对 root 进行右旋
LR	BF(root) = 2, BF(root->lchild) = –1	先对 root->lchild 进行左旋，再对 root 进行右旋
RR	BF(root) = –2, BF(root->rchild) = –1	对 root 进行左旋
RL	BF(root) = –2, BF(root->rchild) = 1	先对 root->rchild 进行右旋，再对 root 进行左旋

2-3树每个结点有2个值、3个子结点，结点值溢出要分裂到根结点/新根。

5、Kruskal/Prim

kruskal：按边权升序不断选取当前未被选取过且权值最小的边，若该边依附的顶点落在集合中不同连通分量上，则将边加入集合；否则舍弃，寻找下一条边，直至边数 = 顶点数 - 1。

Prim：选择一个起点加入最小生成树集合，每次都寻找与当前集合中顶点最近的那个顶点加入集合，直至所有点都已加入。

6、Dijkstra/Floyd

Dijkstra：单源最短路径。任一点，找与他距离最近的点作为中介点，更新与中介点相邻所有点到起点的最短距离，不断循环。

Floyd：全源最短路径。矩阵第i行第j列的元素为同一行元素与同一列元素中能够使它更小的和。

```
for(i=0;i<n;i++){
    for(j=0;j<n;j++){
        for(k=0;k<n;k++){
            dis[j][k]=min(dis[j][k],dis[j][i]+dis[i][k]);
        }
    }
}
```

7、最短增益路径

广度优先搜索：

① 将源点加入队列

② 当队列非空时，将队首结点的相邻点加入队列，（1）前向边：若未标记且容量 $u_i > 0$，则用 $l_i, j+$ 标记，l_i 为允许通过的最大流量，j 为上一结点。（2）后向边：若未标记且流量 $x_j > 0$ 则用 $l_j, i-$ 标记。

③ 从汇点往前遍历，找到增益路径。

④ 除了源点，去除所有顶点的标记，重新初始化。

⑤ 重复 ①~④ 直至汇点未被标记。

8、二分图最大匹配

广度优先搜索：

① 若 w 在集合 V 中，取队首结点 w，遍历相邻结点 u

（1）若是自由顶点，加入边集合，$v=w$，每当结点 v 已被 u 标记，边集合删除 (v,u)，并令 v 为 u 上的标记，重新将 (v,u) 加入边集合，重复循环，循环结束后删除所有顶点标记，用 V 中所有自由顶点重新初始化。

（2）若 u 已匹配，则用 w 标记 u ，将 u 加入队列。

② 若 w 在集合 U 中，用 w 标记其对偶顶点 v ，将 v 入队。

9、决策树

基于深度优先搜索

10、动态规划

第 i,j 步的状态由第 $i-1,j$ 与 $i,j-1$ 步决定。