Tip

时间复杂度不仅仅依赖于问题的规模

若算法所需的辅助空间相对输入数据量是个常数，则称算法为原地工作

线性表

1. 取值插入删除时注意位置的合法性
2. 插入时注意`listsize`的溢出
3. 双向链表插入删除的时候有4个指针要操作

栈

1. `Push()`时注意栈溢出
2. `S.top`指向即将push进入的位置
3. `realloc()`可能会将分配新的空间而非拓展，**返回的指针值可能会变**

队列

1. 循环队列中`Q.rear`指向下一个将入队的位置
2. 两种方式处理循环队列
    1. 设标志判断队空队满
    2. 空一个位置，此时分配的空间要比最大队长多1，即如果数组为A[m]，则队列的最长长度为m-1
3. **链队列有头结点**，即front指向头结点而非队头

串

定长顺序存储

1. 串连接需要考虑截断
2. 求子串考虑起始位置和长度的合法性

堆分配存储表示

1. 注意要覆盖原串，需要将原串给free

模式匹配

KMP

数组

随机存储结构

$$LOC(j_1,j_2,...,j_n)=LOC(0,0,...,0)+L\sum_1^n j_nc_n$$

$$c_n=c_{n-1}b_{n-1},c_n=1$$

矩阵

稀疏矩阵稀疏因子$\delta = \frac{t}{m \times n}$,$t$为非零元，$mn$为矩阵大小

稀疏矩阵表示方法

1. 三元组{i, j, e}

三元组的矩阵转置

1. 按顺序查找的方式将在j列的元素放置新表中，然后j += 1
2. 对每列第一个元素的恰当位置进行计算，然后以行序为主序遍历原表，放入相应的位置并更新每一列的恰当位置
3. 行逻辑链接的顺序表，即加入每行第一个非零元的位置表

矩阵相乘MN

M的每行非零元(i,k)乘N的第k行j列元素加至Q的(i, j)位置，全部加完之后压缩存储

时间复杂度$O(M.tu * N.tu/N.mu)$

3. 十字链表right,down,rhead,chead

十字链表的初始化与相加

广义表

广义表表长为元素个数，表看成一个整体，一个元素

头尾链表空表为null，拓展线性链表还有一个表节点，只是hp，tp都是null

```
原广义表:[2, 3, [4, 5], 6]
表头:2 头表:[2] 表尾:[3, [4, 5], 6]
```

树

二叉树

二叉树不是度不大于2的有序树

度不大于2的有序树没有左右孩子之分

$n_0 = n_2 + 1$

n个节点的完全二叉树深度为$[log_2 n] + 1$

完全二叉树中度为1的节点最多只有一个

顺序存储编号i的节点存在i-1的位置

链式存储结构

n个节点有n+1个空链域

先序+中序，后序+中序，层次+中序 可复原二叉树

后序遍历时，T的左孩子的直接后继为

1. 如果T->rchild == NULL,T->lchild->next = T;
2. 如果T->rchild != NULL, 见下代码

```
p = T->rchild;
while (p->lchild || p->rchild)
{
    if (p->lchild) p = p->lchild;
    else p = p->rchild;
}
```

存储结构

1. 双亲表示法，结构体包含双亲
2. 孩子表示法，包含孩子链表的头指针
3. 孩子兄弟表示法

霍夫曼树

n个叶子的霍夫曼树有2n-1个节点

通过回溯和试探进行图的遍历

含n个节点的不相似的二叉树共$\frac{1}{n+1}C_{2n}^{n}$棵

图

基本概念

网，带权图

a--->b,a弧尾，b弧头

路径P(a,b)

回路P(a, a)

简单路径即图论中的圈

连通分量即图论中的连通片

一个有向图的生成森林由若干棵有向树组成

存储结构

存储结构	创建时间复杂度
邻接矩阵	$O(n^2+en)$
邻接表	$O(n+e)$,若输入为顶点编号；$O(ne)$,若输入为顶点，需要查找
十字链表	$O(n+e)$,若输入为顶点编号；$O(ne)$,若输入为顶点，需要查找
邻接多重表	?

图遍历

DFS用邻接矩阵遍历，查找每个顶点的邻接点所需的总时间是$O(n*n)$

DFS用邻接表遍历，查找邻接点所需时间为$O(e)$，查找顶点需要$O(n)$，总时间$O(n+e)$

连通性

最小生成树

任何一个无向连通图的最小生成树有一棵或多棵

Prim

找$(U,\bar{U})$中代价最小的边，让边代价为0，并且将边的另一个端点加入U中，对这个点的邻边更新代价

时间复杂度$O(n^2)$，适合边稠密的图

Kruskal

初始条件是n个孤立顶点

选择两个顶点落在不同连通分量上的代价最小的边

时间复杂度$O(eloge)$，适合边稀疏

关节点和重连通风量

连通度k，至少删去k个顶点才能破坏图的连通性

关节点：

1. 有两棵子树的根
2. 非叶子节点v点的子树中所有节点都没有指向v的祖先的边，v为关节点

$low(v)=min\{visited(v),visited(v->parent,v->parent->parent,...),low(v->child,v->child->child,...)\}$

如果$low(v->child)>=visited(v)$，v为割点，如果$count<G.vexnum$，根为割点

DAG, AOV-N, AOE

DAG:有向无环图

AOV-N:顶点活动网络，表示顶点活动的先后顺序

AOE-N:边活动网络，顶点为事件，表示边活动的开始或者结束

将偏序集变为全序即拓扑排序

拓扑排序可检测AOV-N是否有环，通过不断删去入度为0的顶点和更新其他顶点的入度，如果没有剩余，则没有环，否则有环

时间复杂度$O(e+n+2*n+e)$

关键路径

先按照拓扑排序的方式计算各顶点的最早发生时间，如果拓扑排序失败，说明有环，并将拓扑排序的序列压入栈中，在对逆拓扑排序序列求最迟发生时间

计算弧活动的最早开始时间和最晚开始时间为$O(e)$，计算顶点事件的最早和最晚发生时间为$O(n)$，总时间复杂度为$O(n+e)$

最短路径

Dijkstra算法(一点到其他点的最短路径)：$S_0=\{v_0\}$,取$(S,\bar{S})$最短的路径的另一个顶点加入S

时间复杂度$O(n^2)$

Floyd(每一对顶点之间的最短路径):找到中间顶点序号<=u的路径，并将其与之前<=u-1的最短路径相比较，然后赋值，最后中间顶点序号<=n-1就是ij之间最短路径

查找

静态查找表：查询，检索

动态查找表：查询，检索，插入，删除

唯一标识某记录的关键字叫主关键字

平均查找长度ASL：需要和给定关键字进行比较的关键字个数的期望

查找方式	ASL
顺序查找	$\frac{n+1}{2}$
折半查找(有序表)	$\frac{n+1}{n}\log_2 n-1$
次优查找树	构造时间$O(n\log n)$
索引查找表	L_b+L_w，L_b为确定所在块的ASL，L_w为在块中查找的ASL，顺序查找$\frac{b+1}{2}+\frac{s+1}{2}$，折半查找块$\log(\frac{n}{s}+1)+\frac{s}{2}$
二叉排序树	平均性能$\leq 2(1+\frac{1}{n})\ \ln n$
平衡二叉树	$\log n$

斐波那契查找：以斐波那契数列进行分割

插值查找：按照比例查找

次优查找树：取使$\Delta P=|\Sigma_{i左边}w-\Sigma_{i右边}w|=|sw_h+sw_{l-1}-sw_i-sw_{i-1}|$最小的i,然后分治

索引顺序表(分块查找)

二叉排序树(二叉查找树)

插入

在搜索的时候将父亲也给传进去，这样当查找不到(p == null)时可以让p指向父亲，方便插入

在搜索的时候传入父亲null，如果该树为空树，则p=null，需要让T直接指向新创建的节点

删除

1. 右子树为空，直接让左子树的根接替该点
2. 左子树为空，直接让右子树的根接替该点

3. 若左右子树都不空，则让左子树的最右边那个点(该点的直接前驱)代替该点，并且删除左子树最右边的那个点，让那个点的左子树的根代替那个点，如果那个点就是该点左子树的根，则让该点的左孩子为那个点的左子树

平衡二叉树(AVL)

平衡因子$BF = Depth(T->lchild) - Depth(T->rchild) = -1, 0, 1$

类型	判定	操作
LL	插入左子树，左子树长高，根bf=1，左孩子bf=1	单次右旋
LR	插入左子树，左子树长高，根bf=1，左孩子bf=-1	左旋右旋
RR	插入右子树，右子树长高，根bf=-1，右孩子bf=-1	单次左旋
RL	插入右子树，右子树长高，根bf=-1，右孩子bf=1	右旋左旋

B树

m阶树

1. 每个节点最多m棵子树
2. 根节点至少两棵子树
3. 除根外，所有非终端节点至少$\lceil \frac{m}{2} \rceil$棵子树
4. 非终端节点$(n, A_0, K_1, A_1, ..., K_n, A_n)$
5. 所有叶子节点都在同一层，不带信息

B+树

m阶B+树

1. n棵子树的节点含n个关键字
2. 所有叶子节点包含所有关键字信息

键树（数字查找树）

哈希

构造哈希函数

1. 直接定址法
2. 除留余数法,最好去素数

产生冲突的两个关键字称为**同义词**

处理冲突

1. $H_i = (H(key) + d_i) mod m$，
   d_i线性探测再散列：$1, 2, 3, ..., m-1$;二次探测再散列：$1^2, -1^2, 2^2, ..., k^2, -k^2, (k \leq \frac{m}{2})$;伪随机数列
2. 再哈希
3. 链地址：将所有冲突归到一个链表上

装填因子$\alpha = \frac{\text{装入记录数}}{\text{哈希表长度}}$