CS2 基础算法串讲

洛谷网校
2023-08
Maxmilite

本次课程中出现「刷屏」等影响讨论区秩序行为的同学将被严厉处罚。

线性数据结构

链表（单链表、双向链表、循环链表）、栈、队列

链表

知识梳理

链表是一种存储数据的数据结构。

特点：插入与删除数据十分方便，但寻找与读取数据的表现欠佳。

与数组的区别：链表是链状结构，数组将所有元素按次序依次存储。因此：

- 随机访问数据中，链表删除、插入数据时间复杂度/操作次数 $O(1)$；寻找、读取数据时间复杂度/操作次数 $O(n)$。
- 随机访问数据中，数组寻找、读取数据时间复杂度 $O(1)$，删除、插入的操作次数/时间复杂度 $O(n)$。

其空间复杂度为 $O(n)$。

显然可以动态扩展。

链表

知识梳理

单向链表：指针域用来连接当前结点和下一结点。

双向链表：指针域有左右（或上一个、下一个）之分，用来连接上一个结点、当前结点、下一个结点。

循环链表：最后一个结点的下一个结点是第一个结点。

插入/修改数据：修改某结点及其前后结点的 prev，next。初赛阶段直接模拟，只需考虑信息覆盖情况（即，要修改的信息的来源，是否已经被修改过）。

例题

题目来源 CSP-J 2022 初赛 | 第 4 题

链表和数组的区别包括（　　）。

A．数组不能排序，链表可以。

B．链表比数组能存储更多信息。

C．数组大小固定，链表大小可动态调整。

D．以上均正确。

例题

题目来源 CSP-J 2022 初赛 | 第 11 题

以下哪组操作能完成在双向循环链表结点 p 之后插入结点 s 的效果（其其中前驱后继/前后结点使用 prev, next 表示）：（　　）。

A. `p->next->prev = s; s->prev = p; p->next = s; s->next = p->next;`

B. `p->next->prev = s; p->next = s; s->prev = p; s->next = p->next;`

C. `s->prev = p; s->next = p->next; p->next = s; p->next->prev = s;`

D. `s->next = p->next; p->next->prev = s; s->prev = p; p->next = s;`

队列、栈

知识梳理

队列是一种具有「先进入队列的元素一定先出队列」性质的表，又称先进先出（FIFO）表。

特点：先进先出（FIFO）

空间复杂度 $O(n)$，单次进出时间复杂度 $O(1)$。

栈是一种具有「先进入栈的元素一定先出栈」性质的表，又称先进后出（FILO）表。

特点：先进后出（FILO）

空间复杂度 $O(n)$，单次进出时间复杂度 $O(1)$。

可以用栈实现队列。

例题

题目来源 CSP-J 2022 初赛 | 第 2 题

有 6 个元素，按照 6，5，4，3，2，1 的顺序进入栈 S。请问下面哪个出栈序列是非法的。

A. 5 4 3 6 1 2

B. 4 5 3 1 2 6

C. 3 4 6 5 2 1

D. 2 3 4 1 5 6

例题

题目来源 CSP-J 2022 初赛 | 第 5 题

假设栈 S 和队列 Q 的初始状态为空。存在 1 ~ 6 六个互不相同的数据，每个数据按照进栈 S、出栈 S、进队列 Q、出队列 Q 的顺序操作，不同数据间的操作可能会交错。已知栈 S 中依次有数据 1，2，3，4，5，6 进栈，队列 Q 依次有数据 2，4，3，6，5，1 出队列。则栈 S 的容量至少是（ ）个数据。

A．2

B．3

C．4

D．6

递推、递归

递归、递推

知识梳理

递归，是指在函数的定义中使用函数自身的方法，在计算机科学中还额外指一种通过重复将问题分解为同类的子问题而解决问题的方法。

递归的基本思想是某个函数直接或者间接地调用自身。

在程序执行中，递归是利用堆栈来实现的。

递推算法通常是通过计算前面的一些项来得出序列中的指定项的值。从已知的规律一步步推向未知的信息。通常可以找到前后之间的数量关系（递推式）。

例题

题目来源 CSP-J 2022 初赛 | 第 15 题

以下对递归方法的描述中，正确的是：（　　）

A．递归是允许使用多组参数调用函数的编程技术

B．递归是通过调用自身来求解问题的编程技术

C．递归是面向对象和数据而不是功能和逻辑的编程语言模型

D．递归是将用某种高级语言转换为机器代码的编程技术

例题

题目来源 CSP-S 2021 初赛 | 第 3 题

在程序运行过程中，如果递归调用的层数过多，可能会由于（ ）引发错误。

A．系统分配的栈空间溢出

B．系统分配的队列空间溢出

C．系统分配的链表空间溢出

D．系统分配的堆空间溢出

简单图论

图的定义与相关概念
图的表示与存储：邻接矩阵、邻接表

图的基本概念

知识梳理

图是由若干点以及连接点与点的边构成的。

若干点构成的点集 V 与若干边构成的边集 E 构成图 $G = (V, E)$。

无向边，有向边：边 $e = (u, v)$ 是否区分先后顺序（$u \to v$）。

无向图/有向图/混合图：仅含无向边/有向边的图叫做无向图/有向图，二者均有的图叫做混合图。

带权图/正权图：图中每条边 e 都被赋予了一个数字（权值 w），$e = (u, v, w)$。如果所有边的权值 w 都是正实数，称图是正权图。

度数/入度/出度：与一个顶点 v 直接连接的边的条数叫做顶点的度数 $d(v)$。以一个顶点 v 为起点/终点的边的条数称为该顶点的出度 $d^+(v)$/入度 $d^-(v)$。

图的基本概念

知识梳理

自环：对边 $e=(u,v)$，若 $u=v$，称 e 为一个自环。

重边：若边集合中有两条边 e_1,e_2 完全相同，则它们被称作一组重边。

简单图：没有重边和自环的图。

连通：无向图中，任意两点间存在一条途径（一连串边首尾相连），称图是连通的；有向图中，任意两点 u,v 间存在两条有向途径使得 u,v 相互可以到达对方，称作强连通，将有向边换成无向边可以连通称为弱连通。无向连通图至少有 $n-1$ 条边，有向图强连通至少有 n 条边（n 是点数）。

稠密图/稀疏图：一张图的边数接近点数的平方，称为稠密图；一张图的边数远小于点数的平方，称为稀疏图。二者没有严格的定义。

完全图：任意不同两点间均有边（无向）/均有两条方向不同的边（有向）

图的存储

知识梳理

初赛需要掌握两类图存储方式：邻接矩阵、邻接表。

邻接矩阵：使用二维数组 e 存边，$e[u][v]=1$ 代表 u,v 之间有连边，$=0$ 代表没有。如果是带边权的图，1 可以替换为边权。优势是可以 $O(1)$ 查询某条边是否存在，劣势是在稀疏图上效率很低，所以一般只会在稠密图上使用邻接矩阵。对于 n 个点 m 条边的图，空间复杂度 $O(n^2)$。

邻接表：使用一个支持动态添加元素的数据结构构成的数组来存边，例如 `vector`。存储内容为与一个点的所有出边的相关信息（终点、边权等）。对于 n 个点 m 条边的图，空间复杂度 $O(m)$。

例题

题目来源 CSP-J 2022 初赛 | 第 9 题、CSP-J 2020 初赛 | 第 8 题

考虑由 N 个顶点构成的有向连通图，采用邻接矩阵的数据结构表示时，该矩阵中至少存在（ ）个非零元素。/ 有 N 个顶点的无向图至少应该有（ ）条边才能确保是一个连通图。

A. $N-1$

B. N

C. $N+1$

D. N^2

简单树

树的定义与表示、二叉树的定义与基本性质、二叉树的表示与存储
二叉树的遍历：前序、中序、后序
特殊树：完全二叉树、Huffman 树与 Huffman 编码、二叉搜索树

树的定义与表示

知识梳理

一个没有固定根结点的树称为无根树。无根树指定一个节点为根，即为有根树。

性质：

1．边数 = 点数 - 1

2．无向无环的连通图，在任意不同两点间添加一条边后所得图有唯一一个环

3．任意两个结点之间仅有一条简单路径

父亲：除根外的结点，从该结点到根的路径上的第二个结点。

祖先：从该结点到根的路径上的除它本身之外的结点。

子结点/后代：u 是 v 的父亲/祖先 $\Rightarrow$ v 是 u 的子结点/后代；

结点深度：结点到根结点路径上的边数；树的高度：所有结点深度的最大值。

子树：某个结点和它所有的后代和边构成的图叫做这个节点的子树。

常见特殊树

知识梳理

链：满足与任一结点相连的边不超过 2 条的树。

菊花：满足存在结点 u 使所有除 u 外的结点均与 u 相连的树。

二叉树：每个结点最多只有两个儿子的有根树。

二叉树

知识梳理

二叉树：每个结点最多只有两个儿子的有根树。

完整二叉树：每个结点子结点均为 0 或者 2。

完全二叉树：只有最下面两层节点度数可以小于 2，且最下面一层节点都集中在该层最左边的连续位置上。

满二叉树/完美二叉树：所有叶节点的深度都相同的二叉树。

前中后序遍历：根左右，左根右，左右根。

序列反推：已知中序遍历序列和另外一个序列可以求第三个序列。

完全二叉树

知识梳理

1. 除最后一层以外第 i 层节点数量：2^{i-1}
2. 具有 n 个节点二叉树深度 $k=\log_2 n+1$
3. 最后一层节点数 $n-2^k+1$
4. 数组表示：根节点为 1，接下来每层从左至右依次标号
5. 对某节点 i，左子节点 $i\times 2$，右子节点 $i\times 2+1$，父节点 $\left\lfloor\frac{i}{2}\right\rfloor$，兄弟节点 $\begin{cases}i-1, i\text{ 是偶数}\\ i+1, i\text{ 是奇数}\end{cases}$

Huffman 树

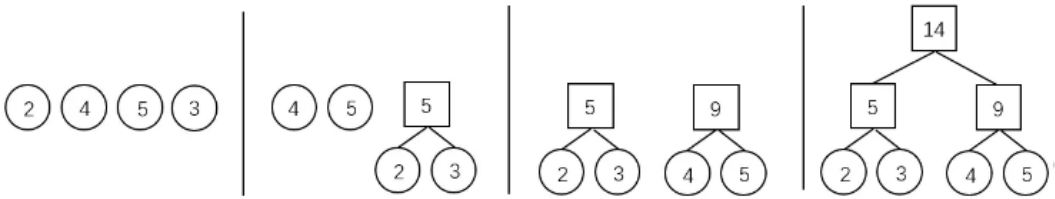
知识梳理

树的带权路径长度（WPL）：设二叉树具有 n 个带权叶结点，从根结点到各叶结点的路径长度与相应叶节点权值的乘积之和。

哈夫曼树：对于给定一组具有确定权值的叶结点，可以构造出不同的二叉树，其中，WPL 最小的二叉树称为哈夫曼树（Huffman Tree）。

哈夫曼算法：

1. **初始化**：由给定的 n 个权值构造 n 棵只有一个根节点的二叉树，得到一个二叉树集合 F。
2. **选取与合并**：从二叉树集合 F 中选取根节点权值 **最小的两棵** 二叉树分别作为左右子树构造一棵新的二叉树，这棵新二叉树的根节点的权值为其左、右子树根结点的权值和。
3. **删除与加入**：从 F 中删除作为左、右子树的两棵二叉树，并将新建立的二叉树加入到 F 中。
4. 重复 2、3 步，当集合中只剩下一棵二叉树时，这棵二叉树就是霍夫曼树。



Huffman 编码

知识梳理

在进行程序设计时，通常给每一个字符标记一个单独的代码来表示一组字符，即编码。如果字符出现的频率不同，则可以让频率高的字符采用尽可能短的编码，频率低的字符采用尽可能长的编码，来构造出一种不等长编码，从而获得更好的空间效率。

前缀编码：任一编码都不是其他任何一个编码的前缀

Huffman 编码：将字符集及出现频率作为叶节点插入构造 Huffman 树。规定哈夫曼编码树的左分支代表 0，右分支代表 1，则从根结点到每个叶结点所经过的路径组成的 0,1 序列即为该叶结点对应字符的编码。

二叉搜索树

知识梳理

定义：

1. 空树是二叉搜索树。
2. 若二叉搜索树的左子树不为空，则其左子树上所有点的附加权值均小于其根节点的值。
3. 若二叉搜索树的右子树不为空，则其右子树上所有点的附加权值均大于其根节点的值。
4. 二叉搜索树的左右子树均为二叉搜索树。

例题

题目来源 CSP-J 2020 初赛 | 第 12 题

独根树的高度为 1。具有 61 个结点的完全二叉树的高度为（ ）。

A. 7

B. 8

C. 5

D. 6

例题

题目来源 CSP-J 2022 初赛 | 第 7 题

假设字母表 {a，b，c，d，e} 在字符串中出现的频率分别为 $10\%, 15\%, 30\%, 16\%, 29\%$。若采用哈夫曼编码的方式对字母进行不定长的二进制编码，字母 d 的编码长度为（　）位。

A．1

B．2

C．2 或 3

D．3

搜索

深度优先搜索、广度优先搜索
深度优先遍历、广度优先遍历

深度/广度优先搜索/遍历

概念复习

深度优先搜索

使用递归函数进行暴力枚举。

需要用到栈。

广度优先搜索

使用队列进行暴力枚举。

需要用到队列。

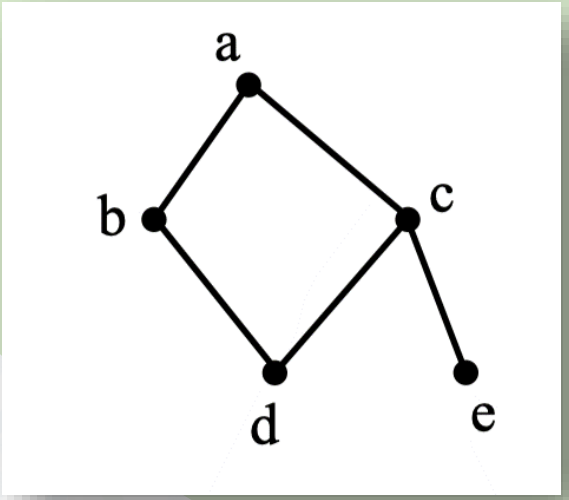
深度优先遍历：每次都尝试向更深的节点走。

广度优先遍历：每次都尝试访问同一层的节点。如果同一层都访问完了，再访问下一层。

例题

题目来源 CSP-J 2021 初赛 | 第 14 题

以 a 为起点，对下边的无向图进行深度优先遍历，则 b, c, d, e 四个点中有可能作为最后一个遍历到的点的个数为（ ）。



A．1

B．2

C．3

D．4

排序

冒泡排序、选择排序、插入排序、计数排序

排序

概念复习

稳定性是指相等的元素经过排序之后相对顺序是否发生了改变。

算法名称	最好时间复杂度	最坏时间复杂度	平均时间复杂度	是否稳定
冒泡排序	$O(n)$	$O(n^2)$	$O(n^2)$	是
选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	否
插入排序	$O(n)$	$O(n^2)$	$O(n^2)$	是
计数排序	$O(n+w)$，w 为值域	$O(n+w)$，w 为值域	$O(n+w)$，w 为值域	是
归并排序	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	是
快速排序	$O(n \log n)$	$O(n^2)$	$O(n \log n)$	否

例题

题目来源 CSP-J 2022 初赛 | 第 12 题

以下排序算法的常见实现中，哪个选项说法是错误的（　）。

A．冒泡排序算法是稳定的

B．简单选择排序是稳定的

C．简单插入排序是稳定的

D．归并排序算法是稳定的

杂项

前后缀表达式

逆波兰表达式

概念复习

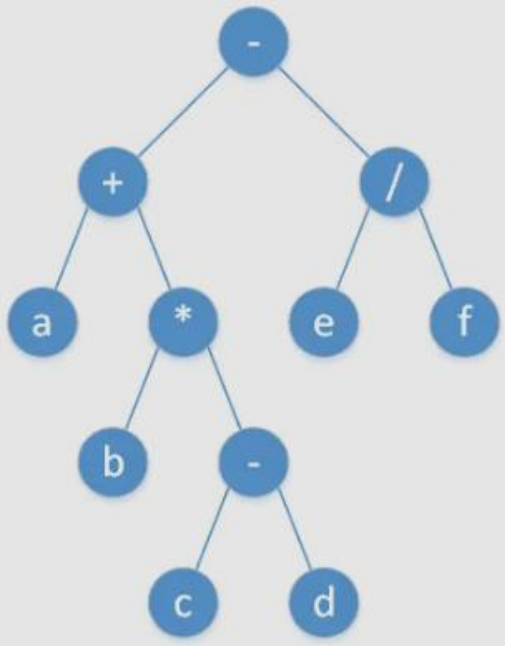
表达式树上进行树的遍历可以得到不同类型的表达式。

算术表达式分为三种，分别是前缀表达式、中缀表达式、后缀表达式。

中缀表达式是日常生活中最常用的表达式;

后缀表达式是计算机容易理解的表达式。

前后缀表达式转中缀表达式：栈



例题

题目来源 CSP-J 2022 初赛 | 第 6 题

对表达式 `a + (b - c) * d` 的前缀表达式为（）。

A. `*+a-bcd`

B. `+a*-bcd`

C. `abc-d*+`

D. `abc-+d`

限时练习