一、调试成功程序及说明

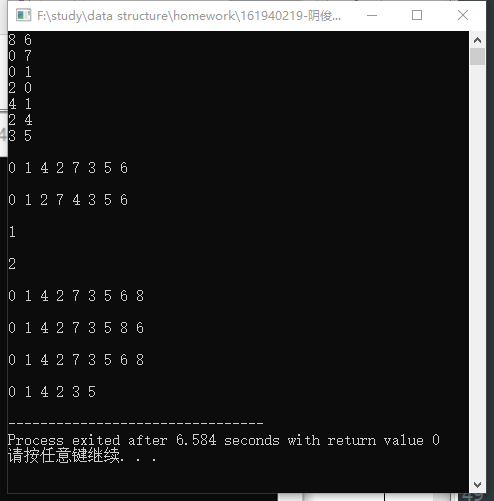
1. 编程实现书P96 ADT Graph 基本操作11个，用邻接矩阵存储结构实现；

算法思想

DFS：类似树的先序遍历，利用递归实现。函数起初先访问该结点，再标记该结点已被访问，然后对于该结点的每一个邻接点，若还未被访问过，再次递归调用DFS访问它。

BFS：类似树的层序遍历，利用队列实现。先令起始结点入队，并标记已访问。只要队不空就出队，并访问它，对于其每一个未被访问过的邻接点，再次入队，并在结点入队时标记其已访问。

运行结果：



结果分析：

T(n) = O()

S(n) = O()

附源程序。

2. 输入N个权值（1-100正整数），建立哈夫曼树。

算法思想：

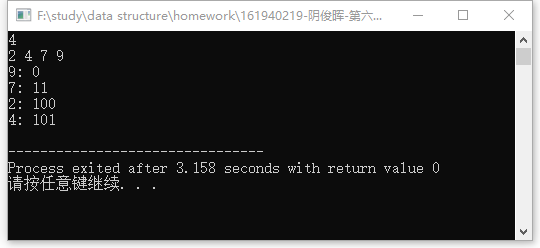
1. 建立哈夫曼树

将n个权值放入最小堆中，一次循环使两个权值出队，作为一棵新的树的左右孩子。直至堆中元素个数为1，完成建树。

1. 为哈夫曼树编码

基于层序遍历，预先为每个结点分配一段用于编码的空间。入队时若是左孩子，编码末尾加0，若是右结点，编码末尾加1，直到遍历至叶结点，完成编码。

运行结果：



结果分析：

T(n) = O(n\*log(n))

S(n) = O(n)

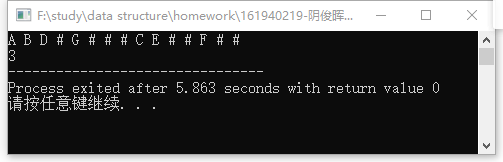
附源程序。

3. 编写函数，对二叉链表结构的二叉树，求宽度。（树的宽度指具有最多结点数量的那一层上的结点数）（书P94 4）

算法思想：

利用levelRear记录该层末尾结点，levelNum记录该层结点数，max记录最多结点数量。利用层序遍历访问结点，每访问一个结点levelNum加1。若当前访问的结点是该层末尾结点，实时更新max，levelNum，levelRear，具体操作如下：若levelNum大于当前max，将levelNum赋值给max，levelNum重置为0，若该树还有下一层结点，更新levelRear，反之循环终止。

运行结果：



结果分析：

T(n) = O(n)

S(n) = O(n)

附源程序。

4. 编写函数，对一棵以孩子-兄弟链表表示的树，输出第i层的所有元素。

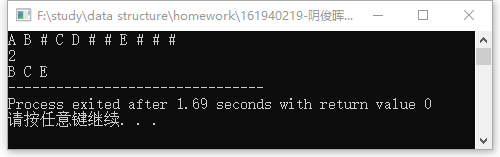
算法思想：

先让让根结点入队，然后执行以下循环i-1次。

一层循环：先让所有长子依次入队，再让长子的所有兄弟们依次入队。

循环结束后，队中剩余结点即第i层的所有元素，依次出队并输出即可。

运行结果：



结果分析：

T(n) = O(i\*levelCount)

S(n) = O(n)

附源程序。

5． CSP题目

问题描述：俄罗斯方块是俄罗斯人阿列克谢·帕基特诺夫发明的一款休闲游戏。

　　游戏在一个15行10列的方格图上进行，方格图上的每一个格子可能已经放置了方块，或者没有放置方块。每一轮，都会有一个新的由4个小方块组成的板块从方格图的上方落下，玩家可以操作板块左右移动放到合适的位置，当板块中某一个方块的下边缘与方格图上的方块上边缘重合或者达到下边界时，板块不再移动，如果此时方格图的某一行全放满了方块，则该行被消除并得分。

　　在这个问题中，你需要写一个程序来模拟板块下落，你不需要处理玩家的操作，也不需要处理消行和得分。

　　具体的，给定一个初始的方格图，以及一个板块的形状和它下落的初始位置，你要给出最终的方格图。

输入格式：

　　输入的前15行包含初始的方格图，每行包含10个数字，相邻的数字用空格分隔。如果一个数字是0，表示对应的方格中没有方块，如果数字是1，则表示初始的时候有方块。输入保证前4行中的数字都是0。

　　输入的第16至第19行包含新加入的板块的形状，每行包含4个数字，组成了板块图案，同样0表示没方块，1表示有方块。输入保证板块的图案中正好包含4个方块，且4个方块是连在一起的（准确的说，4个方块是四连通的，即给定的板块是俄罗斯方块的标准板块）。

　　第20行包含一个1到7之间的整数，表示板块图案最左边开始的时候是在方格图的哪一列中。注意，这里的板块图案指的是16至19行所输入的板块图案，如果板块图案的最左边一列全是0，则它的左边和实际所表示的板块的左边是不一致的（见样例）

输出格式：

输出15行，每行10个数字，相邻的数字之间用一个空格分隔，表示板块下落后的方格图。注意，你不需要处理最终的消行。

样例输入：

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1 0 0

0 0 0 0 0 0 1 0 0 0

0 0 0 0 0 0 1 0 0 0

1 1 1 0 0 0 1 1 1 1

0 0 0 0 1 0 0 0 0 0

0 0 0 0

0 1 1 1

0 0 0 1

0 0 0 0

3

样例输出：

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1 0 0

0 0 0 0 0 0 1 0 0 0

0 0 0 0 0 0 1 0 0 0

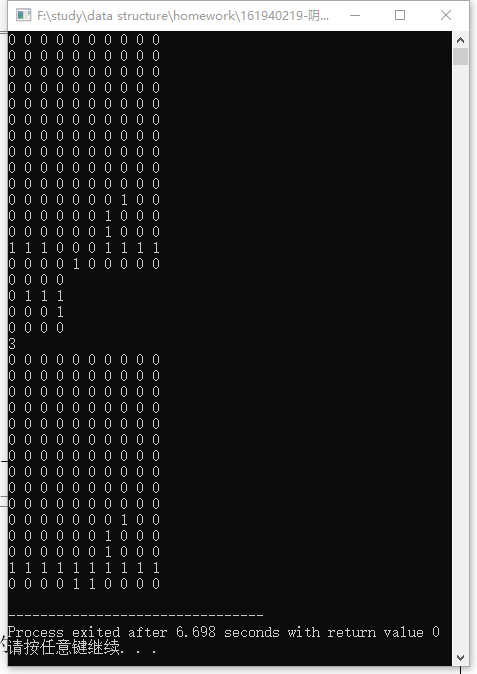
1 1 1 1 1 1 1 1 1 1

0 0 0 0 1 1 0 0 0 0

算法思想：

1. 读入数据：将原始放格图和下落板块分别读入两个二维数组中。
2. 先找到下落板块的初始位置，然后下落，直到不能继续下落时，将1填充到原始放格图的二维数组当中。
3. 输出现在放格图的二维数组。

运行结果：



结果分析：

T(n) = O()

S(n) = O()

附源程序。

-------------------------------------------------------------------------

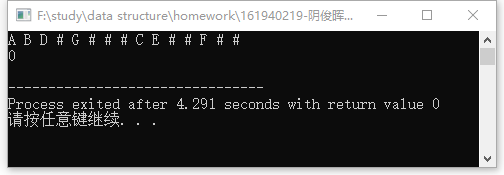
选做题**6.49 编写算法判别给定二叉树是否为完全二叉树**

算法思想：

基于层序遍历。

层序遍历所有非空结点的指针，并将它们的左右指针（无论空或非空）都入队，若在队列中存在非空指针在空指针之后的情况，则该树不是完全二叉树，若遍历结束之后，所有空指针都在非空指针之后，则该树是完全二叉树。

运行结果：



结果分析：

T(n) = O(n)

S(n) = O(n)

附源程序。

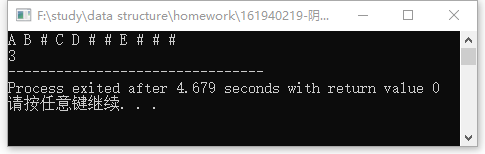
选做题**6.60 试编写算法，对一棵以孩子-兄弟链表表示的树统计叶子的个数。**

算法思想：

若一棵以孩子-兄弟链表表示的树中该结点没有左孩子，表明原树中该结点是叶结点。

所以只要遍历一次树，统计该树中没有左孩子结点的总数即可。

运行结果：



结果分析：

T(n) = O(n)

S(n) = O(n)

附源程序。

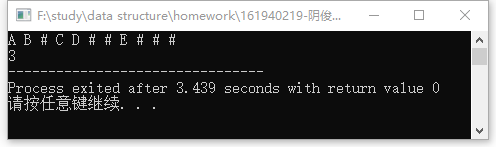
选做题**6.62 对以孩子-兄弟链表表示的树编写计算树的深度的算法。**

算法思想：

同第4题。

利用第4题的思路，可在以孩子-兄弟链表表示的树的基础上实现层序遍历，只要知道遍历到哪一层时队列为空（即该层没有结点），上一层的层数即为树高。

运行结果：



结果分析：

T(n) = O(i\*levelCount)

S(n) = O(n)

附源程序。

二、代码行数及小结

|  |  |
| --- | --- |
| 题目 | 代码行数 |
| mygraph.h | 29 |
| mygraph.cpp | 151 |
| binaryTree.h | 45 |
| binaryTree.cpp | 386 |
| 1 | 37 |
| 2 | 10 |
| 3 | 9 |
| 4 | 12 |
| 5 | 99 |
| 选6.49 | 9 |
| 选6.60 | 9 |
| 选6.62 | 9 |
| 总计 | 805 |