第 10 次上机解题报告

15081070 张雨任

一、A 题

1. 题目代号与评测记录序号：{A289963}
2. 解题思路：
   考察最基本的树的操作，前序遍历建立树，然后后序遍历来计算树的高度，递归取左右子树中更高的高度即可。

二、B 题

1. 题目代号与评测记录序号：{B290274}
2. 解题思路：
   这道题考察数组的操作，读懂题即可。循环条件是在当前下标不等于目标下标，先按照传送规则计算 cnt+=arr[cnt]，接下来如果 cnt>=n（到头了）或者 cnt>t（超过预期了），说明失败了，跳出。跳出后，如果 cnt==t 说明成功了，否则失败了。

三、C 题

1. 题目代号与评测记录序号：{C292716}
2. 解题思路：
   这道题考察图的存储和深度优先遍历。存储选择了二维数组形式的邻接表来实现，深度优先遍历通过递归可以解决。一开始选择了邻接矩阵，但是这样会超时，因为题目中的点的编号是 0-5000 的，所以需要开一个 5000*5000 的二维数组，但是由于点并不是连续的，并且只有 1000 个点，所以这个矩阵肯定会比较稀疏，这样遍历 5000*5000，会超时。所以我用了邻接表来实现，因为只有 1000 个点，所以开一个 5000*1000 的二维数组叫做 adjlist。值为 n 的点，与他相连的点在二维数组的第 n 行，二维数组的每一行的第一个数记录这一行元素的个数。也就是说，对于值为 n 的点，adjlist[n][0]记录与 n 相连的点的个数，adjlist[n][1]、adjlist[n][2]…（如果存在的话）记录与这个点相连的点，这样就可以形成一个形如邻接表的结构（实为一个二维数组）。因此还需要一个检查的数组，用来检测这个点是否存在，因为点的范围为 0 到 5000，但并不是所有值都是这个图中的一个结点。所以 check_appear[i]=true 代表值为 i 的点是这个图的一个结点，=false 代表不是。然后要把每一行的点从小到大排序，因为题目中说优先遍历小的点，不使用邻接表的链表的结构是因为链表不能很方便的排序，这里使用数组，直接调用 sort 函数就可以了。最后遍历每行依次进行 dfs，对于结点值为 v 的点，dfs 的时候只需要遍历到二维数组这一行的第 adjlist[v][0]个（也就是最后一个），而之前使用邻接矩阵则是要遍历到 5000，因此使用邻接表能够大大降低时间复杂度。

四、D 题

1. 题目代号与测评记录序号：{D291958}
2. 解题思路：

   首先理解题意。对于每个数，可以乘二，也可以减一，也就是一个父结点可以产生两个子结点，那么就等价于求值为 m 且距离根节点（值为 n）最近的结点的层数。有点类似于二叉树的结构，但是不用二叉树来实现。

   这道题使用了广度优先搜索。设立一个结构体名为 node，含有两个数据元素，val 代表某一个点的值，level 代表这个点的层数，也就是从 n 到这个点所进行的操作次数。既然是广度优先搜索，就要使用队列。当然还要注意重复的点就不要再放进队列了，因为已经有这个点以及它能够产生的点了，再放的话就重复了，而且还进行了很多的相同操作，很耗费资源，所以要设立查重数组，对于值为 x 的点，没处理过的值记为 check[x]=0，处理过的记为 check[x]=1。接下来就是广度优先搜索的算法了。首先把 val 为 n 且 level 为 0 的点先入队，接下来在队列不为空的情况下进行循环。首先取队首元素 tmp 并出队，把 check 数组中下标为 tmp.val 置为 1，代表处理过值为 tmp.val 的点了，再遇到 tmp.val 的点就不用管了，接下来要把值为 2*tmp.val 的点和 tmp.val-1 的点入队，但是这里要注意很多细节。首先是操作，新建一个类型为 node 的对象，把 val 值置为 tmp.val*2，把 level 置为 tmp.level+1，原因是这个 node 是 tmp 这个节点产生的出来的，所以操作次数（也就是层数）比 tmp 要多 1，再把 check 数组中下标为 tmp*2 的置为 1，代表这个点操作过了，然后把新节点入队。这里最重要的是入队的条件，首先要查重，操作过的话就不要在入队了，也就是 check[x]==0 的点才有资格入队，还要注意 tmp.val 是要小于等于 10000 的，因为如果某数经过变换已经达到 10000 以上，而 m 又一定是小于等于一万的，所以就没有必要对它进行乘二的操作了，再乘二就更大了，只能减一才能达到目标的值，所以这里入队的条件是 tmp.val<=10000 且没重复过。然后再入队 tmp.val-1 的情况，入队的操作是类似的，区别在于入队的条件，首先还是不能是重复过的值，然后要保证减一之后不能为负数，原因是生成了负数的话，这个数不管乘二还是减一都还是负数，而目标值 m 一定是正数，所以要剪掉这一部分的情况。这样就处理好了对于队首的值所可能产生的两个值的入队操作。接下来如果这个队首的值（tmp.val）等于 m，说明达到了目标值 m，即可跳出并输出这个点（tmp）的 level 就可以了，否则就重复之前的操作，这就是这道题的广度优先搜索的算法。
3. 易错点：

   这道题很容易超时，所以要做好剪枝，首先是查重能够节省很多时间，然后有些注定不会产生答案的结点就不要入队了，也就是要注意结点入队的条件。然后就是慎用 map，一开始犯懒查重用的 map 来查的，结果 t 到怀疑人生，所以能用基本的结构（这里用的数组）处理的事情就不要用 STL。

五、E 题

1. 题目代号与测评记录序号：{E290815}
2. 解题思路：

   这道题考察树的建立和递归遍历。前序遍历建树就不在赘述。这里需要用到前序遍历来识别叶子结点。遍历时前提是当前结点不为空，在这个情况下，

如果该结点左右孩子皆空，说明是叶子结点，就把累计的层数放在数组中，不是叶子结点的话，则自增 n，在左孩子不空的情况下，递归遍历左孩子，在右孩子不空的情况下，递归遍历右孩子。最后遍历数组每一个值，累计出和，再除以个数算出来期望，输出注意.2lf 即可。

六、F 题

1. 题目代号与测评记录序号：{F290896}
2. 解题思路：

   此题和 E 很类似，只不过改成了每个结点都要算，所以直接在该结点不为空的情况下，把该结点的的层数放进数组，然后自增 n，在左孩子不空的情况下，递归遍历左孩子，在右孩子不空的情况下，递归遍历右孩子。最后进行一样的操作，遍历数组每一个值，累计出和，再除以个数算出来期望，输出注意.2lf 即可。

七、G 题

1. 题目代号与测评记录序号：{G292108}
2. 解题思路：

   这道题用到了并查集的思想。首先要考虑，所有的点中，有一部分点是可以相互连通的，因此也就没有必要为他们设立中转的点了。在所有点中，有一些点可以互相连通，但是他们不能和另外的一些点连通，因此建立新的点就是为了连通这些本不连通的点的。两个点，只要是横坐标相等或者纵坐标相等就说明这两个点是连通的。这样就可以得到几个集合，集合元素是点。如果有 n 个集合，说明需要至少新建 n-1 个点，来连通他们，可以保证任意两个集合都可以有连通的路径可以走。

   下面来说说如何使用并查集来把本身孤立的点连接成集合。首先构建一个结构体数组，数据成员是横坐标 x、纵坐标 y 和父结点的下标 father。初始化的时候要输入结点的横纵坐标，还要把 father 置为-1，-1 代表没有父结点，也就是根节点，输入结束之后是合并集合的算法。对于任意两个点，记下标为 i 和 j。这两个点可以合并的前提是：结点 i 的横坐标等于结点 j 的横坐标或者结点 i 的纵坐标等于 j 的纵坐标，因为这说明这两个点是可以直线相连的，可以直接滑冰过去的。合并的规则是如果这两个结点的根结点相同，则他们不用合并，因为已经在一个集合中了；如果根结点不同，就需要进行合并。需要合并的话，就把 i 结点对应的根节点的父指针（本来为-1）等于 j 结点对应的根节点的下标，也就是把 j 所在的树并到 i 所在的树中，成为他的一个子树。这里涉及到如何求某一结点的根结点的下标，就是沿着父指针链寻找这个结点的根，循环跳出条件是当这个结点的 father=-1，说明遍历到了根节点了。用这样的方法就可以进行集合的合并了。

   最后只需要遍历所有的点，如果 father==-1，说明这个点是根节点，总共有 n 个根结点也就说明有 n 棵树，也就是说有 n 个集合，说明需要新建 n-1 个新点才能连通这些集合，最后输出 n-1 即可。