第 9 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：{A286878}
2. 解题思路：
   输入 n 个四位数，用 int 数组来存储。然后用 sort 函数从小到大来排序，接下来输出数组的第 m-1 个就是第 m 小的尾号。不过答案好像没有考虑前导零的情况，导致输出带有前导零的答案是错误的。

二、B 题

1. 题目代号与评测记录序号：{B288393}
2. 解题思路：
   这道题使用小顶堆来解决。首先理解题意，输出给定集合的丑数序列（从小到大排序）的第 m 个，丑数即为尽可以分解为给定集合中的元素的数。那么显然丑数序列中任意两个数相乘还是丑数，这样这个数还是仅可以分解为给定集合中的元素。使用小根堆的具体实现如下：首先把 1 入堆，虽然 1 不可为丑数，但是方便进行乘法运算，所以先把 1 放进堆中。之后进入循环，循环 n+1 次，每次循环的第一个的操作是，去除堆的第一个元素，也就是最小的元素，循环 n+1 次是因为第一次出堆的是 1，1 并不是丑数，所以要出堆 n+1 次才能得到结果，当然，出堆的就是答案 ans，每次出堆之后，要把 ans 乘以给定集合的每一个元素，并且在堆中没有这个元素的情况下把这个值放进堆中，有的话就不放了。也就是说，每次出堆一个值之后，最多在进堆 k（给定集合的元素数量）个，然后通过堆来排序。这里查重用到了 map，map 可以理解为映射，如果这个乘出来的元素 map 之后的值为 0，说明这个值没有在堆中，然后每次把这个值的 map 置为 1，之后再遇到这个值，进行 map 操作之后值就是 1 了，此时说明这个值已经处理并入堆过了，就不能再入堆了。这样就可以保证堆中没有两个相同的元素。最后输出 ans 即可。

三、C 题

1. 题目代号与评测记录序号：{C287367}
2. 解题思路：
   这道题读懂题就可以上手了。核心思想是对于每一个任务，它的运行时间和所求的时间间隔的交集，是这个任务的有效运行时间，因此求出两个区间下限的最大值，在求出两个区间上限的最小值，然后用最小值减去最大值再加一得到有效运行时间，如果说明值为负数，则说明两个区间没有交集，则说明有效运行时间为 0，每次用这个时间乘该任务单位时间消耗的资源，累加这个值，最后输出。

四、D 题

1. 题目代号与测评记录序号：{D287928}
2. 解题思路：

这道题考察二叉树的前序建树和遍历。首先用前序遍历建树，这里使用 string 比读入 char 更方便，接下来是遍历。基本是按照前序遍历的思路来的，首先设置 last 初始为一个很大很大的值，用来记录当前情况下，所遍历过的叶子结点的最小层数。一个 now 是当前叶子结点累计的层数。记下来，如果 root 为空，说明树为空，答案记 0，否则，进入前序遍历的算法：首先当此结点左右孩子都为空，且 now<last，说明这个结点是叶子结点，now<last 代表这个叶子结点坐在的层数比之前所记录的叶子结点的最小层数还要小，那么置 last 等于 now。如果不满足这个条件，就使得 now 自增，在左孩子不为空的情况字下，前序遍历递归左孩子，在右孩子不为空的情况下，前序遍历递归右孩子。保证后代不为空是因为不能递归到空结点，否则就会把 last 置为 0。最后输出 last 即可，也就是所有叶子结点的最小层数。

五、E 题

1. 题目代号与测评记录序号：{E289155}
2. 解题思路：

这道题分为两部分来考虑。对于一棵无穷的满 k 叉树，且每一个节点与它的子节点相连的边的权值分别为 1,2,…,k，要求在这棵树上有多少条从根出发不重复经过边的路径满足经过的边权值之和为 n 而且至少经过一条权值不小于 d 的边。对于这两个限制条件，可先求出对于满 k 叉树经过的路径的权重之和为 n 的路径条数，再求出对于满 d-1 叉树经过的路径的权重之和为 n 的路径条数，两值相减即为所求。原因如下：因为至少经过一条权值不小于 d 的边的反面就是经过的边的权重都是小于 d 的，也就等价于去求满 d-1 叉树经过的路径的权重之和为 n 的路径条数。因此这两个值相减就是所求。下面先求出对于满 k 叉树经过的路径的权重之和为 n 的路径条数。对于满 k 叉树，路径权重之和为 n 即可等价于权重和为 n-1 的加上 n-2 的加上 n-3 的……一直加到 n-k 的，写成公式就是 a[n]=a[n-1]+a[n-2]+…+a[n-k]，这就是一个递推的过程了，相当于：走到某处权值和为 n 的可以等价于“之前共累计了 n-1 然后选择走权值为 1”加上“之前共累计了 n-2 然后选择走权值为 2”加上“之前共累计了 n-3 然后选择走权值为 3”一直加到“之前共累计了 n-k 然后选择走权值为 k”。当然，还要考虑 n-k<=0 的情况，这种情况下公式就是 a[n]=a[0]+a[1]+…+a[n-1]，即可理解为：走到某处权值和为 n 的可以等价于“之前共累计了 n-1 然后选择走权值为 1”加上“之前共累计了 n-2 然后选择走权值为 2”加上“之前共累计了 n-3 然后选择走权值为 3”一直加到“之前共累计了 0 然后选择走权值为 n”。因此公式为：

$$\mathrm{ans}[n] = \mathrm{ans}[n-1] + \mathrm{ans}[n-2] + \cdots + \mathrm{ans}[n-k]\ , \qquad n > k$$

$$\mathrm{ans}[n] = \mathrm{ans}[n-1] + \mathrm{ans}[n-2] + \cdots + \mathrm{ans}[0]\ , \qquad n \le k$$

实现的话，先按照 n<=k 的情况来写，累积地求出 ans[i]，累加就可以了，接下来当 i 超过了 k 之后，就减去第一项，接下来的每次循环都减去第 i-k-1 项来保障当前 ans 是从 ans[n-k]开始累积的。还要记得每轮循环要对 1000000007 取余，对于累计的值小于 0 的情况，要加 1000000007 让他变成正数。对于第二种情况求满 d-1 叉树和求上面求满 k 叉树没有区别，改个数就可以了（把

k 都改成 d-1 即可），这样就可以得到一个 limit 数组。把它们相减就可以了，就是 ans[n]-limit[n]。减之前还要注意一些细节，首先要保证答案是正数，所以 ans[n]小于 limit[n]的时候，结果为负数，所以要加上 1000000007。还要注意 d=1 的时候其实第二个情况（求满 d-1 叉树）的结果是不存在的，所以仅仅输出 ans[n]也就是仅求满 k 叉树的情况就可以了。

3. 易错点：
   取余不要忘掉，还有要判断 tmp 为负数的情况，都要加上 1000000007，注意 d=1 的情况。

六、G 题

1. 题目代号与测评记录序号：{G288440}
2. 解题思路：
   这道题可以使用树的结构。题意是：邮件从 A 发到 B，前提是 A 有这封邮件，B 没有这封邮件。由于一个人可以发给多个人，所以这是一颗树，要求的就是这棵树的高度。首先，输入姓名不区分大小写，所以要把输入的姓名都变为小写字母来存储。这道题我是用结构体数组来存储每个人，每个结点有三个变量，一个 vector 类型的 next 来记录子女的编号，一个 string 类型的 name 来记录名字，一个 int 类型的 lengthOfPre 来记录此结点的父结点的高度，这个结构体数组名为 people，再创建一个 cntPeople 来记录结点的编号，从 1 开始来避免冲突，因为 map 映射到 int 的话默认值为 0，最后可以得出总人数。
   首先读入 name1（收件人）和 name2（发件人），收件人 name1 是从来没有收到过邮件的，所以他一定没在 people 数组中出现过，所以把名为 name1 的结点放在 people 数组的末尾，编号为 cntPeople。这里设立了两个 map 类型的变量，check 来记录名字为 xxx 的结点，其编号是 check[xxx]，另一个是 chk1，来记录名字为 xxx 的人是否做过收件人，如果 chk1[xxx]=1，说明这个人做过收件人，那么这个人一定不是初始发件人，也就是说它不是这棵树的根。然后把 name1 的 check 置为当前的 cntPeople 的值，把 name1 的 chk1 置为 1，说明这个人做过收件人了。这样就处理好了收件人 name1。接下来分类讨论处理发件人，如果之前存储过名为 name2 的结点，也就是 check[name2]!=0，那么用 check[name2]来定位到这个结点的下标，在向量 next 中 push_back 名为 name1 的结点的编号，也就是 cntPeople-1，如果之前没有存储过名为 name2 的结点，则要在 people 数组末尾新建一个名为 name2 的结点，把 name2 的 check 值置为 cntPeople，把 name1 的编号（也就是 cntPeople-1）放到 next 向量的末尾，然后累加 cntPeople，就完成了收件人 name2 的处理。循环 c 次，这样就完成了初始化，建立了一个暂时未知根结点的树，每个结点的孩子的编号用 next 向量表示。
   接下来，建立一个栈，先把根结点放进去，寻找根结点的方法如下：遍历整个 people 数组的 name，也就是所有人的名字，如果这个名字的 chk1 不等于 1，即 chk1[xxx]!=1，说明这个人从来没做过收件人，说明这个人是最初的邮件传播者，记下此时的下标，就是根结点的下标 start。之后把 start 入栈，设置 now 和 last，now 来记录当前路径的累计的长度，last 来记录在此之前的叶子结点的历史最大高度。在栈不为空的情况下，弹出栈顶的数，令当前高度 now 就等于这个弹出的数指向的结点的 lengthOfPre 加一，lengthOfPre 代

表某结点的父结点的高度。之后遍历该结点的所有孩子（也就是 next 的所有元素），把它们的 lengthOfPre 置为 now，也就是当前高度，然后再依次进栈。如果某一结点的 next 向量为空（.size()为 0），说明这个结点的是叶子结点，则要和 last 相比，如果比 last 大，说明 now 是新的最大的叶子结点的高度，则把 now 赋给 last。最后遍历整个树之后输出 last。这道题和上次上机的最后一道题有点像。