第 6 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：{A274656}
2. 解题思路：
   这道题考察基本的队列操作。坑点很多，自己也不幸地跳了进去。add 是入队操作，delete q 是进行 q 次出队操作，get m 是从 front 加 m-1 就可以找到目标值，但是我用的是循环队列，所以还要对 maxSize 取余数。同时，我设置了 cntNum 来计算当前队列中的人数，用来判别操作是否合法，每次入队自增，每次出队自减。
3. 易错点：
   这道题易错点很多，所以在这里好好总结一下：
   ① 首先 q（delete 的操作数）和 m（get 的操作数）不能为负数，也不能超过 cntNum 的值，很明显此时操作是非法的。
   ② 最大的坑点是 delete 0 是合法的操作，虽然代表删除 0 个元素，但依然是合法的，因此不应该输出 GzhIsSoHandsome。我就错在这里，以后还要好好审题，认真考虑边角情况。

二、B 题

1. 题目代号与评测记录序号：{B273404}
2. 解题思路：
   删除线性表中的某一指定位置的元素，我是用链表来实现。

三、C 题

1. 题目代号与评测记录序号：{C273550}
2. 解题思路：
   汉明距离表示两个相同长度字对应位不同的数量。两整数对应位不同，很容易想到异或运算，对两个整数进行异或运算得出的值就是：对于两个数的二进制的每一位，对应相同为 1，不同为 0，所得的十进制数。也就是说，对于这个二进制的结果，计算它按位为 1 的个数即是汉明距离。例如，对于 110110，其 1 的个数为 4。那么现在如何求一个二进制数按位为 1 的个数？这是一道微软的面试题，思路就是每执行一次 tmp=tmp&(tmp-1)，会将 tmp 的二进制表示的最右边的 1 变为 0，对于 tmp=110110，tmp-1=110101，tmp&(tmp-1)=110100。一次类推，每次都把二进制数的最右边的 1 变为 0，并自增计数器，当 tmp 变为 0 的时候，即可跳出循环，输出计数器。

四、D 题

1. 题目代号与测评记录序号：{D273796}
2. 解题思路：

这道题看起来很难的样子，而且 n 开到很大的值。所以肯定只能用字符串来表示数。因此也很难用迭代的方式来求得第 n 个偶长度回文数是多少，所以只能从 n 和对应的偶长度回文数之间寻找对应关系。所以就试着写了几个数，有如下规律：f[1]=11, f[2]=22, f[3]=33, f[4]=44, f[5]=55, f[6]=66, f[7]=77, f[8]=88, f[9]=99, f[10]=1001, f[11]=1111…，因此有对于任意数 abcde，有 f[abcde]=abcdeedcba。因此就把字符串正着输出，接着倒置输出即可。

五、E 题

1. 题目代号与测评记录序号：{E276389}
2. 解题思路：

   这道题有点不好理解，就是从找规律入手。

   首先硬看题干，能看出序列是：a, ab, abc, abcd, abcde, abcdef, abcdefg, …，到了 a*之后就开始 aa, ab, ac, ad, ae, af, ag, ah, ai，所以我猜想后面会有 b*, ba, bb, bc, bd, …, bi，（也就是形如 x*, xa, xb, …, xi 的规律）但是到了 ii 之后我就不知道该怎么继续了，所以一开始我的输入就到 9045 为止，9044 和 9045 分别是 i 和 i，代表 ii。但是这道题的数据最大可以到达 int，显然这个序列不应该有尽头，不可能到 9045 就停了。所以 ii 之后肯定还要继续。之后我发现可取的字母有 9 个，另外还有个*，我就思考为什么不是 26 个字母，我感觉这些字母可能可以对应数字，也就是用 1~9 代替 a~i，0 代替*，再按规律写一遍，就发现在 ii 之后就可以写下去了，ii 代表 99，接下去就是 100,101,102,103…，那么再往下就是 a**,a*a,a*b,a*c…，至此终于读懂题了。

   然后，我就把它分为很多行，写成分行的形式，更清楚一些，就变成了：

   a
   ab
   abc
   abcd
   abcde
   abcdef
   abcdefg
   abcdefgh
   abcdefghi
   abcdefghia*
   abcdefghia*aa
   abcdefghia*aaab
   abcdefghia*aaabac
   abcdefghia*aaabacad
   abcdefghia*aaabacadae
   …
   …
   …

   对于求这个列表的第 m 个字母，首先要找到它在第几行（到底在第几行不重要，重要的是锁定到那一行去找它在第几个），再检索它在第几个，即可确定那个位置的字母是什么。接下来介绍一下算法。基本思想就是对于第 m 个字

符，每次减去每行的个数，直到它小于了当前行字符的个数，此时的 m 值即是即是所求字符的下标。m 每次减去某一行的个数之后，都还要在字符数组之后继续添加新的字符，使得这个数组能够作为下一行的字符串。每次都要添加新的字符进去，一直到最后，m 已经不够减当前行个数的值了，就可以去通过 m 作为下标寻找改行的字符了（比如，m=67，通过上述计算可知它在第 11 行的第 11 个，那么直接输出第十一个字符即可）。

接下来讲讲向数组添加新的字符的算法。先举个栗子，对于当前数组，最后两位值为 di（49），那么按照题干规律，下一个要向数组添加的字符是 e*（50），也就是通过 di 模拟加一的计算来达到 e*，那么这里对字符串 di 来模拟加一的运算就可以了，这个“新的加法运算”有如下运算法则要格外注意：a~h 就是 1~8，直接加一就可以达到对应的字符，如 c+1=d（3+1=4）。对于 i+1（9+1）这个运算，得到的结果是 j，但是 j 不是所用的字母，要把 j 换成*（也就是 0），并且进位。最后把得到的字符放在数组的后面即可。

那么设当前要添加的字符串的位数为 digitNum，当前行的最后 digitNum 个字符就是要新添加的字符，就把最低位向前回溯 digitNum 个元素，把最低位加一（也就是找到 di 的 i，对于 di 就是 i+1，按照运算法则，得到*并且进位），把进位保留在 int 类型变量 c 中。接下来，如果前一位进位了，那么这一位要加一，否则，就直接照抄过来（对于 di 来说，最低位加一的过程中进位了，那么 d 需要加一来得到 e，最终就得到了 e*，放在数组后面）。

每次达到 digitNum 位数的末尾，就要给 digitNum 加一，比如，当前位数为 2，最后两个字符是 ii（99），那么按照规律，接下来是 a**（100），此时 digitNum（当前字符的位数）要自增。我的方法是从最后一位向前遍历 digitNum 个，如果都是 i（9），那么说明到达了 digitNum 位数的最末尾，位数需要加一。接下来就是 a**（100）了，再用上面的“向数组后面添加新的字符串的算法”把它放到字符数组中即可。

最后输出累减后的 m 指向的数组的元素即可。

3. 易错点：
   ① 加一的时候要注意进位
   ② 回溯 digitNum 个元素要注意一个细节，比如，当前 digitNum 为 3，数组的最后四位为 ihii（9899），回溯之前这三个位置已经留了出来，但是还没有值，要把这三个位置变成 a**（100）。也就是依次回溯 digitNum=3 个元素，那么回溯过程如下：最后一个元素回溯到 iihi 中的最后一个 i，加一变成*，并且进位；倒数第二个元素回溯到 ihii 中的倒数第二个 i，接受前一位的进位加一变成*，并且进位；倒数第三个元素回溯到 h，加上进位变成 i，这就出问题了，那个 h 并不属于 ii（99），此时的 a**变成了 i**，这显然是不对的，所以要额外讨论，每次位数（digitNum）增加的时候，要手动把最高位置成 a 即可。这样，就可以把 a**放在 ii 的后面了。

六、G 题

1. 题目代号与测评记录序号：{G274890}
2. 解题思路：

这道题的难点在于理解题意。对于一个 01 串，它有自己的最长交替子串，现

在要对这个子串进行一个操作，然后再求新串的最长交替子串。这个操作就是：对这个 01 串的某非空连续子串进行取反。以上是题意。

那么这道题的重要结论是，若操作前的 01 串的最长交替子串的长度为 cnt，总长度为 len，那么操作后的串最长交替子串的长度为：cnt+2 或 len (cnt+2>len)。后者好理解，串的最长交替子串是显然不能超过 len（串的总长），下面讲一下 cnt+2 是怎么来的。

① 对于一个 01 串，要对一个连续交替子串取反，最多可以使这个连续交替子串的左一个和右一个加入到这个连续交替子串中，使得总的最长交替子串（cnt）的大小加上 2（也就是刚刚提到的左一个和右一个）。举个栗子，某串 0101000001010101010111111，操作前它的最长交替子串是红色部分，对红色部分的一连续子串（注意这个子串不能在头部和尾部，其两头一定要有数字）取反，变为 0101000010101010101011111，那么此时串中两个黄色的字符也加入到这个最长交替子串中，此时 cnt+2。

② 对于任意 01 串，令某连续交替子串之后或之前的第二位取反，可使长度增 2。例如：还是某串，0101000001010101010111111，把黄色的 0 变为 1，则串变为 0101010001010101010111111，此时，黄色也是交替子串的一部分，此时 cnt+2。

那么现在来讲一下什么时候操作后的串的最长交替子串为 len，结论是操作前的 cnt=len-1 或 cnt=len。对于前一种情况 cnt=len-1，可以考察串 1010010101010，红色部分为操作前的最长交替子串，对于前部的 1010 取反，得到 0101010101010，黄色的字符加入最长交替子串的队伍中。此时操作后的 cnt 为操作前的 cnt 加一（也可解释为串的总长）。对于 cnt=len，就很好理解了，串本身就是连续交替子串，对于 0101010101，直接对全体成员取反，变为 1010101010，cnt 不变，大小仍为 len。

可以总结出一个公式，cnt1 位操作前的最长交替子串的长度，cnt2 位操作后的最长交替子串的长度，len 为串的总长度，可以有公式：

$$cnt2=cnt1+2\ (cnt1<=len-2)$$

$$cnt2=len\ (cnt>len-2)$$

最后再讲一下 cnt 怎么求得，把 flag 设为字符串的第一位，遍历字符串的每一位，只要和 flag 不同的值，就计入最长交替子串的序列中，同时 cnt++，并且把 flag 设置为 01 串当前位的值。即可记录最长交替子串的长度。

3. 易错点：

+2 的思路比较不好想到，多列几个例子就能发现操作后的串并不会比操作前的交替子串长很多。