第 5 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：{A269880}
2. 解题思路：
   这道题我的思路是使用结构体数组，记录每个队员的初始编号、实力和积分，完全模拟每次对战，每次对战之后排序即可。数组的下标代表排名减一，最后输出所求排名的选手的编号即可。Debug 浪费了很多时间，cmp 的 a.score==b.score 写成了 a.score==a.score。

二、B 题

1. 题目代号与评测记录序号：{B270654}
2. 解题思路：
   这道题对我来说读题和理解题是最大的困难。列车分为两种操作，进栈和出栈，每次进栈输出一个 A，每次出栈输出一个 B。进栈是按照编号从小到大的顺序的，需要出栈的时候就不进栈。把所要的编组序列存储在数组（变量名为 shuzu）中，并用一个指针 j 指向数组中的值，出栈的顺序一定是按照这个数组的顺序排列的。当且仅当栈顶的元素和指针 j 指向 shuzu 的值（shuzu[j]）相等的时候，出栈，并把元素放在一个想象中的栈，这个栈中的元素就是按照编组顺序排列的，但是并不需要真正建立这个栈，因为输出的仅仅是进栈和出栈的过程。每次出栈之后，指针 j 自增，以指向 shuzu 的下一个值，也就是下一个要编组的列车。如果 shuzu[j]不等于栈顶元素，说明无法出栈，那么就要进栈了。循环的控制条件是如果出栈的次数达到了列车的个数，说明编组完毕，循环结束。
3. 易错点：
   要格外注意栈空的时候，栈空的时候 pop 不出来任何元素，但是还是会打印一个 B 出来，所以每当栈空的时候不能执行出栈的操作。

三、C 题

1. 题目代号与评测记录序号：{C269669}
2. 解题思路：
   这道题不要想复杂，直接用字符串就行，用下标从两头遍历，相等或者两边有一个为 ”?” 为符合，否则为不符合。

四、D 题

1. 题目代号与测评记录序号：{D269309}
2. 解题思路：
   之前练习赛出过几乎一样的，这次加上一个 ”<>” 就可以了。每遇到左括号就进栈，每遇到右括号就和栈里的左括号匹配，匹配则左括号出栈，不匹配

则跳出循环并宣告失败。最后遍历完成后栈为空则匹配成功，不为空则说明有多余的左括号在栈中。

五、E 题

1. 题目代号与测评记录序号：{E270592}
2. 解题思路：

   遍历输入的字符串。

   首先，如果是左括号，就压入栈中，并且判断其是否符合“符号的嵌套规则”，具体规则转化成算法可以用如下的方式来判断：如果栈顶的括号是{，那么从栈顶数第二个一定要是{，如果是别的就跳出并宣告失败，因为只要大括号能括在大括号外面；如果栈顶是[，那么从栈顶数第二个一定要是{，因为小括号和中括号都不能在中括号外面；如果栈顶是(，这种情况要特殊讨论，按照题意：小括号外面不能有小括号，也不能有大括号，那么就从栈顶的这个小括号开始遍历，只要遇到的括号不是中括号，那么就出错了，因为不存在{[()]}和{()}的情况。

   其次，如果遍历到的括号是右括号，那么就要和栈中的左括号进行匹配了，匹配成功就把左括号从栈中弹出，不匹配的话就直接跳出并宣告失败即可。

   最后都遍历完成之后，如果栈为空，说明匹配成功，输出 Yes，如果不为空，说明有左括号在栈中没有右括号来和它匹配，说明匹配失败，输出 No。
3. 易错点：

   这道题没 a 是因为没看公告，把{[()]}当做了正确，但{}里面只能放[]不能出现()。还是自己太大意，做题毛手毛脚不仔细。

六、F 题

1. 题目代号与测评记录序号：{F271561}
2. 解题思路：

   这道题的难点在于建模和优化。

   首先要提取一些需要的变量，并且通过恰当的方式组建模型。题目中涉及到的变量有应用程序编号、每一条消息的状态（未读还是已读）、以及 t（最开始的 t 个消息）。把这些变量用一个结构体来表示，所以我建立了 notice 结构体数组，结构体 notice[i]代表第 i 条出现的消息所拥有的的一些指标，notice[i]结构体中包括两个变量，app（int 类型）代表当前消息（notice[i]）所属于的应用程序编号，status（int 类型）代表当前消息的读取状态，0 代表该消息还没产生，1 代表未读，2 代表已读。

   之后是算法。设立三个标记值：cntTotal 指当前未读消息的数量，也就是要输出的值。cntTime 指当前已生成（出现）的消息数量，指向当前最新生成的未读消息。tMax 后面会解释是什么。然后读取两个值（type 和 op），操作 1 代表出现了新的未读消息，那么此刻（cntTime）的 app 值置为 op，status 置为未读（也就是 1），之后 cntTime 自增以指向下一个还未产生的未读消息，总未读消息数（cntTotal）也自增。先讲操作 3，因为操作 2 会用到操作 3 产生的值。操作 3 就是阅读最开始 t 个消息，把它们的 status 置为已读（也就是 2）。那么从 0 开始遍历到 t，只要遇到了 status 为未读（也就是 1）的 notice，

就把它置为 2（已读），并且总未读消息数自减 1 即可，这种算法很好想出来，但是会超时。那么考虑一种情况，比如：某次 t 为 100，代表前 100 个消息置为已读，那么从 0 遍历到 100，依次改变 status 值；之后又有一次，t 为 150，代表前 150 个消息置为已读，那么再从 0 遍历到 150，依次改变 status 值。但是重要的是，之前 t 为 100，改变了最开始 100 个消息的 status，之后 t 为 150，就不需要从 0 开始了，只需要从 100 开始遍历，因为前 100 个的状态必定是已读，不需要改变状态，也就没有遍历的意义。因此设立了前文提到的 tMax，顾名思义，当前 t 的历史最大值。有一个重要的结论：tMax 之前的消息都是已读的!! 因此每次都从 tMax 开始遍历，改变需要标为已读的消息的状态。如果 t 小于等于 tMax，就不要进行操作，因为 tMax 之前的消息都是已读的，更不必说 t 之前的了；如果 t 大于 tMax，那么从 tMax 开始到 t 的所有消息（notice）的 status 置为已读（也就是 2），并且总未读消息数自减相应的数量。之后，把 tMax 置为 t，作为新的 tMax。而对于操作二（把应用程序编号为 op 的所有消息置为已读），其实就类似了，也是由于 tMax 之前的消息必定都是已读的，所以从 tMax 遍历到 cntTime（也就是当前已经生成的总消息数量），只要 app（代表该消息来自于哪个应用程序）等于操作数 op（在操作 2 中 op 代表应用程序的编号）且它的状态是未读，就置为已读，并且总未读消息数自减。最后每次操作之后输出当前的 cntTotal（总未读消息数）即可。

3. 易错点：

   这道题容易超时，所以优化很重要。

   ① 每次未读的状态改变为已读，都要去改写总未读消息数的值（cntTotal），不能到最后遍历所有消息看哪些是已读哪些是未读的从而算出 cntTotal。

   ② tMax 的设立是最重要的，每次操作 3 之后，t 之前的所有消息都应该变为已读。而且 tMax 之前的消息永远都不用再去管了，因为他们永远是已读了。

   ③ tMax 不仅仅在操作 3 中使用，在操作 2 中也要使用，原因同上。

七、G 题

1. 题目代号与测评记录序号：{G272250}
2. 解题思路：

   这道题对我来说是个挑战，很开心能做出来。

   这是一道搜索题，学名应该是广度优先搜索。这道题我是用队列来实现。基本思想就是在某种棋局之下，按照上左右下的顺序以及游戏规则交换空位置和空位置边上的棋子以生成新的棋局，把这种棋局之下的所有可能出现的新棋局的情况遍历出来，放在队列中，然后检查出现的新棋局能否符合游戏的结束规则（任意横竖斜可以四子连成直线），如果队列中的所有棋局都不能，就让他们出队，并且依次再让他们生成新棋局放在队列中，然后记录步数的计数器自增。以此类推，直到生成满足题意的棋局，输出计数器加一的值（因为计数器自增在循环的末尾，最后一轮并没有自增）。

   之后说一下这个思路的一些细节：

   首先，为什么是广度优先搜索。

   这道题要求最小的步数，而不像迷宫要到最后的终点。所以从某一步棋局生

成的下一步所有可能情况中，只要有符合的游戏结束的规则的棋局，即可输出步数。而深度优先搜索，得出的步数并不一定是最少步数，还要把所有能够达到游戏结束规则的步数都计算出来，再取最小值。我认为不如广度优先搜索方便，所以选择了广度优先搜索。

其次，我选择了队列作为存放棋局的容器。

不符合游戏结束规则的棋局扔掉，生成的新棋局放到队列中，符合队列的先进先出的特性。操作方式是：先从初始棋局生成所有可能的新棋局放进队列中，然后依次检查这些棋局是否符合游戏结束的规则，如果符合，输出步数加一，否则，就把这些棋局生成的他们对应的新棋局放到队列中。因此，同时在队列中的所有棋局，它们的步数是相等的。

然后，再谈一谈函数。

1. 要有一个函数来生成新的棋局，叫做 nextChess 函数，遍历棋盘的所有棋子，是‘O’并且移动到目标位置的操作合法，就交换‘O’和目标位置，并把这个移动之后的新棋局放到队列中，移动包括上下左右，但是前提是移动合法，这很重要。
2. 讲一讲移动是否合法的判断函数（moveIsValid）。首先，移动不能越出数组的界限，在边界的‘O’不能和边界以外的东西交换；其次，下棋是轮流的，因此上一手移动的棋子不能和这一手移动的棋子相同，相同就是非法移动，不能和目标位置的棋子交换。这个用结构体数组实现即可，用一个 char 记录每一手移动的棋子颜色，再用下标来记录下棋的步数，就可以定位到前一手移动的棋子的颜色。
3. 要有一个 check 函数，这个比较容易，只要两个斜或者某一竖行某一横行的棋子颜色相同，即可返回 1，代表匹配成功，否则返回 0，代表失败。

基本上这道题的思路就是这样了。

3. 易错点：

① 判断移动是否合法要注意轮流下棋这个规定。

② 队列要有三个指针比较好一些，head（因为 front 高亮了看着不舒服），rear，还要有 current 来指向某一棋局。由于出队的时候 head 指针也向后移动了，但是还要有指针来指向这些已经出队的棋局，因为还要让他们依次生成新的棋局，所以要这样使用。

③ 由于步数多了的话，生成的棋局也会变得非常多（理论上每一个棋局可最多生成 8 个新棋局），所以队列的数组也要开的大一些比较保险。

④ 如果输入的数据本身就是符合游戏结束规则的棋局，就直接输出 0 就行了，省事又省心。