**电子科技大学**

**计算机科学与工程学院**

**标 准 实 验 报 告**

**（实验）课程名称 人工智能**

**电子科技大学教务处制表**

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：刘文晨 学 号：2018080901006 指导教师：钟秀琴**

**实验地点：主楼A2-412 实验时间：2020-10-18 周日 5-8节 9-节**

**一、实验室名称： 计算机学院实验中心**

**二、实验项目名称：知识表示方法—渡河问题**

**三、实验学时：4学时**

**四、实验原理：**

本次试验选择传教士过河问题，以状态空间法实现。解答步骤如下：

1. 设置状态变量并确定值域

M为传教士人数，C 为野人人数，B为船数，要求M≥C且M,C≤3，L表示左岸，R表示右岸。

初始状态 目标状态

L R L R

M 3 0 M 0 3

C 3 0 C 0 3

B 1 0 B 0 1

1. 确定状态组，分别列出初始状态集和目标状态集

用三元组来表示：（ML , CL , BL）（均为左岸状态）

其中,BL∈{ 0 , 1}

：(3 , 3 , 1) ： (0 , 0 , 0)

初始状态表示全部成员在河的的左岸；

目标状态表示全部成员从河的左岸全部渡河完毕。

1. 定义并确定规则集合

仍然以河的左岸为基点来考虑，把船从左岸划向右岸定义为Pij操作。其中，第一下标i表示船载的传教士数，第二下标j表示船载的食人者数；同理，从右岸将船划回左岸称之为Qij操作，下标的定义同前。共有10种操作，操作集为

F={P01，P10，P11，P02，P20，Q01，Q10，Q11，Q02，Q20}

P10 if ( ML ,CL , BL=1 ) then ( ML–1 , CL , BL –1 )

P01 if ( ML ,CL , BL=1 ) then ( ML , CL–1 , BL –1 )

P11 if ( ML ,CL , BL=1 ) then ( ML–1 , CL–1 , BL –1 )

P20 if ( ML ,CL , BL=1 ) then ( ML–2 , CL , BL –1 )

P02 if ( ML ,CL , BL=1 ) then ( ML , CL–2 , BL –1 )

Q10  if ( ML ,CL , BL=0 ) then ( ML+1 , CL , BL+1 )

Q01 if ( ML ,CL , BL=0 ) then ( ML , CL+1 , BL +1 )

Q11 if ( ML ,CL , BL=0 ) then ( ML+1 , CL +1, BL +1 )

Q20 if ( ML ,CL , BL=0 ) then ( ML+2 , CL +2, BL +1 )

Q02 if ( ML ,CL , BL=0 ) then ( ML , CL +2, BL +1 )

1. 当状态数量不是很大时，画出合理的状态空间图

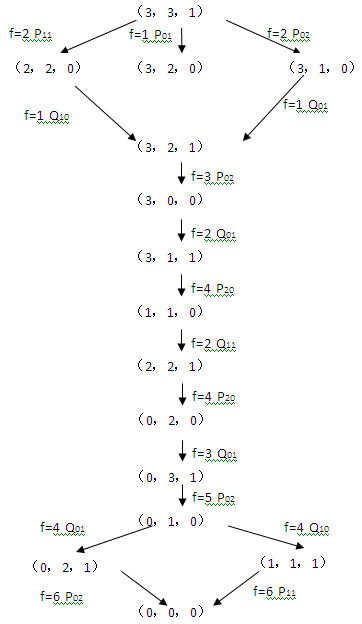


图1 状态空间图

箭头旁边所标的数字表示了Ｐ或Ｑ操作的下标，即分别表示船载的传教士数和食人者数。

接下来进行树的遍历，根据规则由根（初始状态）扩展出整颗树，检测每个结点的“可扩展标记”，为“-1”的即目标结点。由目标结点上溯出路径。

**五、实验目的：**

（1）了解知识表示相关技术；

（2）掌握状态空间法的分析方法。

**六、实验内容：**

状态空间法实验。从前有一条河，河的左岸有m(=3)个传教士、c(=3)个野人和一艘最多可乘n(=2)人的小船。约定左岸，右岸和船上或者没有传教士，或者野人数量不超过传教士，否则野人会把传教士吃掉。搜索一条可使所有的野人和传教士安全渡到右岸的方案。

要求：

（1）提交源代码及可执行文件。

（2）提交实验报告，内容包括：对代码的简单说明、运行结果截图及说明等。

（3）加分点：可手动输入m,c,n，然后输出安全渡河的过程；如无安全渡河的方案，也给出提示

**七、实验器材（设备、元器件）：**

PC微机一台

**八、实验步骤：**

**程序执行流程：**

1. 首先，包含状态（首次为初始状态）的结构体结点（已存入结构体数组）传入处理函数，然后判断该传入结点状态是否为目标状态。

是则遍历打印结构体数组，打印完成之后，返回递归调用处，顺序执行之后代码（此步骤关系到是否能找到所有过河路径）；

否则继续判断是否该传入结点已存在于结构体数组当中，如存在，不再往下执行，返回递归调用处，顺序执行之后代码；

若不存在，则继续判断该传入状态的人数是否合理（是否出现人物数量小于0的情况等），若不合理，返回递归调用处，顺序执行之后代码；

若合理，则继续判断传教士和野人人数限制条件，即在传教士人数不为0的情况下，野人人数是否大于传教士人数，若大于则出现吃人的情况，也就是说该传入状态也不合理，则返回递归调用处，顺序执行之后代码；

若不满足大于条件，则说明该状态是路径转态，也就是合理的，那么进行五种渡河方案的依次变换，首先为第一种渡河方案，两个传教士过河（注意：此处的5中渡河方案没有固定顺序，也可以是其他渡河方案），那么对该传入状态的左岸和右岸的传教士人数和野人人数进行增减。

1. 增减完成并改变船的状态（使用正负一表示，正一为左岸，负一为右岸）以后就产生了一个新的状态，将该状态存入结构体数组，之后此处又递归调用处理函数，将新产生的转态结点传入，再次进行上述条件限制判断。

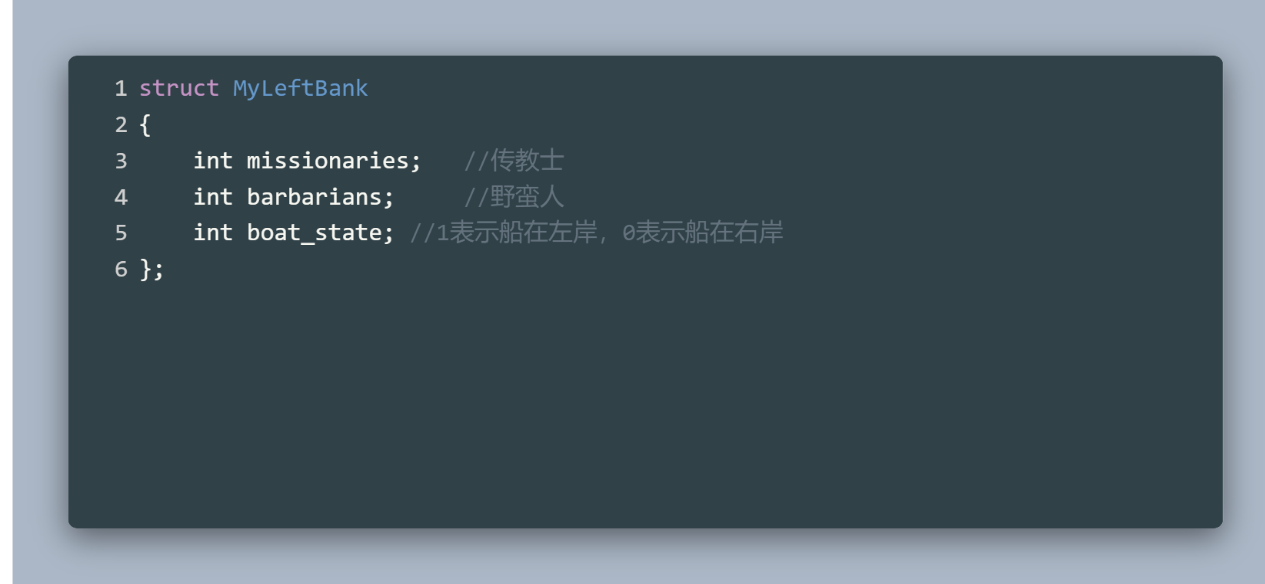
若在该判断途中被返回至递归调用处，说明该状态不合理，则此时将已经存入结构体数组的状态结点移出结构体数组，然后程序顺序执行，进行下一个渡河方案的处理，也就是说，此时的处理是对上一个传入结点的操作（因为刚传入的已经移出了）；

若在判断途中未被返回至递归调用处，也就是说，传入的结点合理了，那么又开始从第一种渡河方案开始对该传入状态进行操作。

1. 按照上述过程循环执行，直到出现目标状态，回到本段开头，遍历结构体数组，打印渡河路径结点信息。
2. 完成以后，返回递归调用处，顺序执行之后代码，此后的操作是在寻找其他渡河路径。

**程序说明：**

1. 结构体MyLeftBank定义传教士、野蛮人和船的状态

****

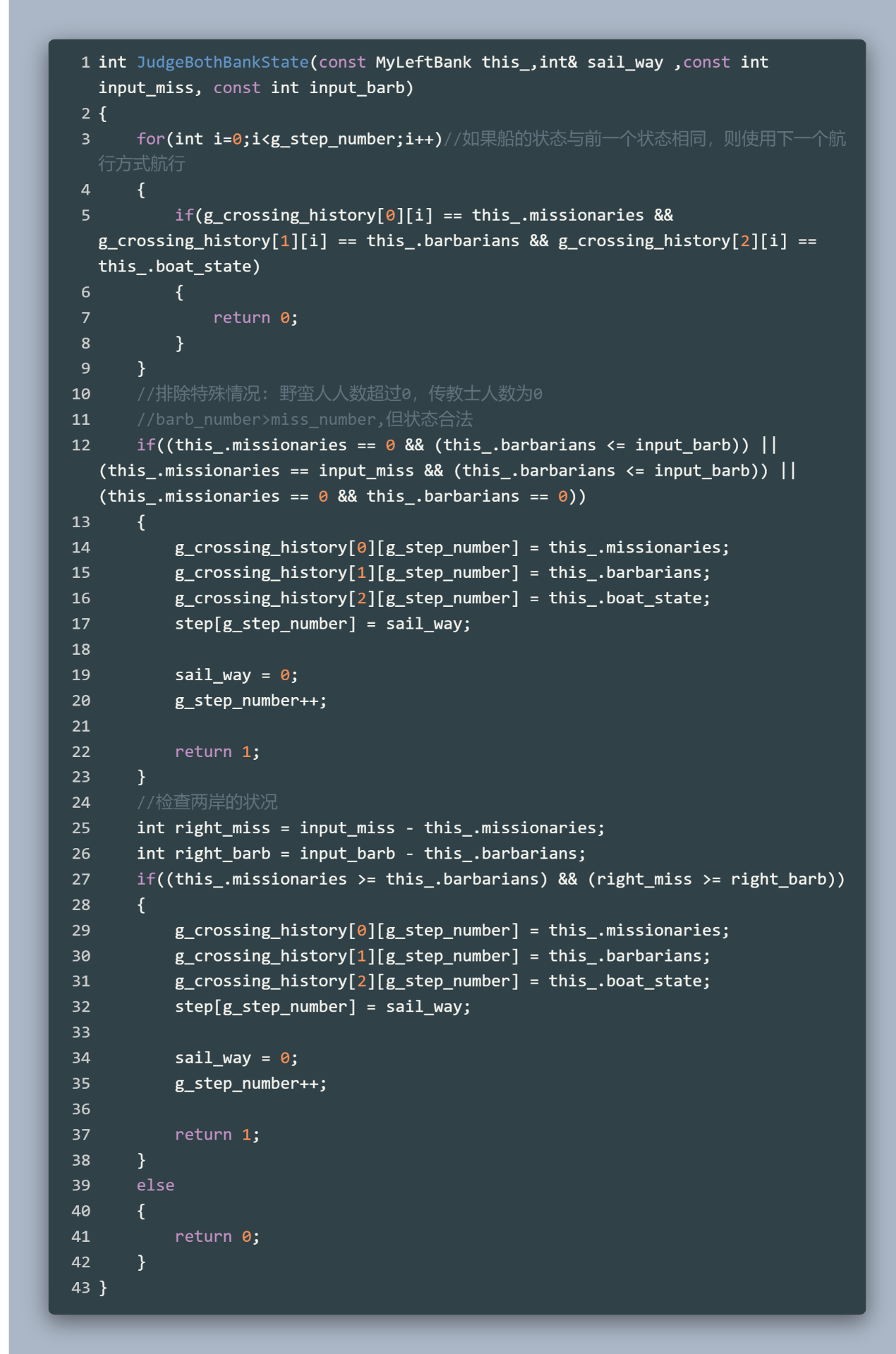
1. InitSailWay函数负责初始化传教士、野蛮人的数量和路线编号



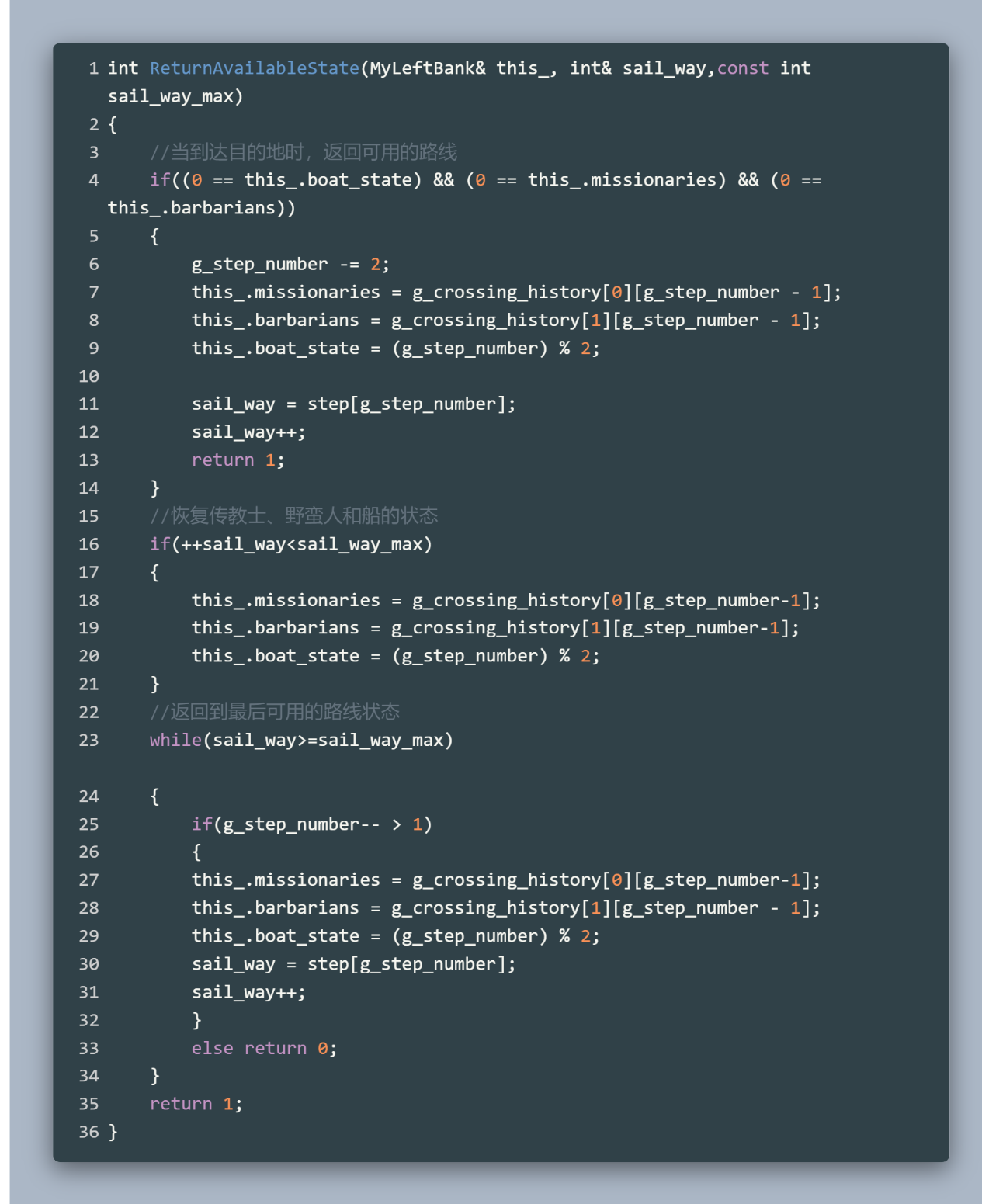
1. BoatGoOppositeBank函数负责船完成一次航行后，传教士、野蛮人和船的状态转换，出错返回-1



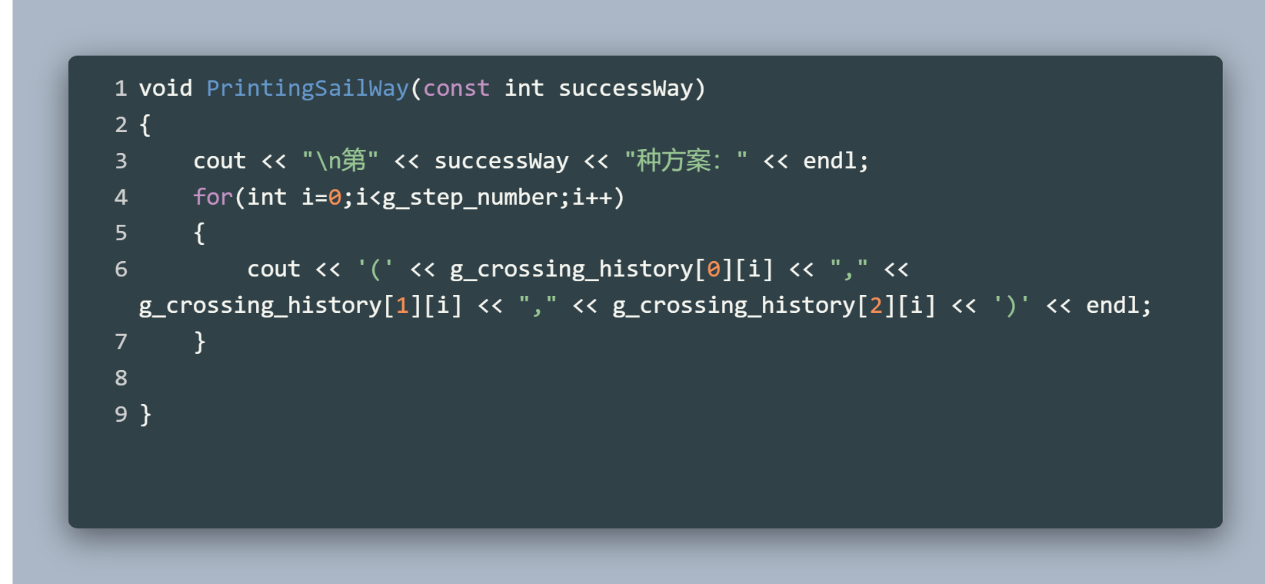
1. JudgeBothBankState函数负责检查两岸的状况是否合法



1. ReturnAvailableState函数负责当到达目标状态时，返回可用的路线



1. PrintingSailWay函数负责输出安全渡河的方案编号和过程

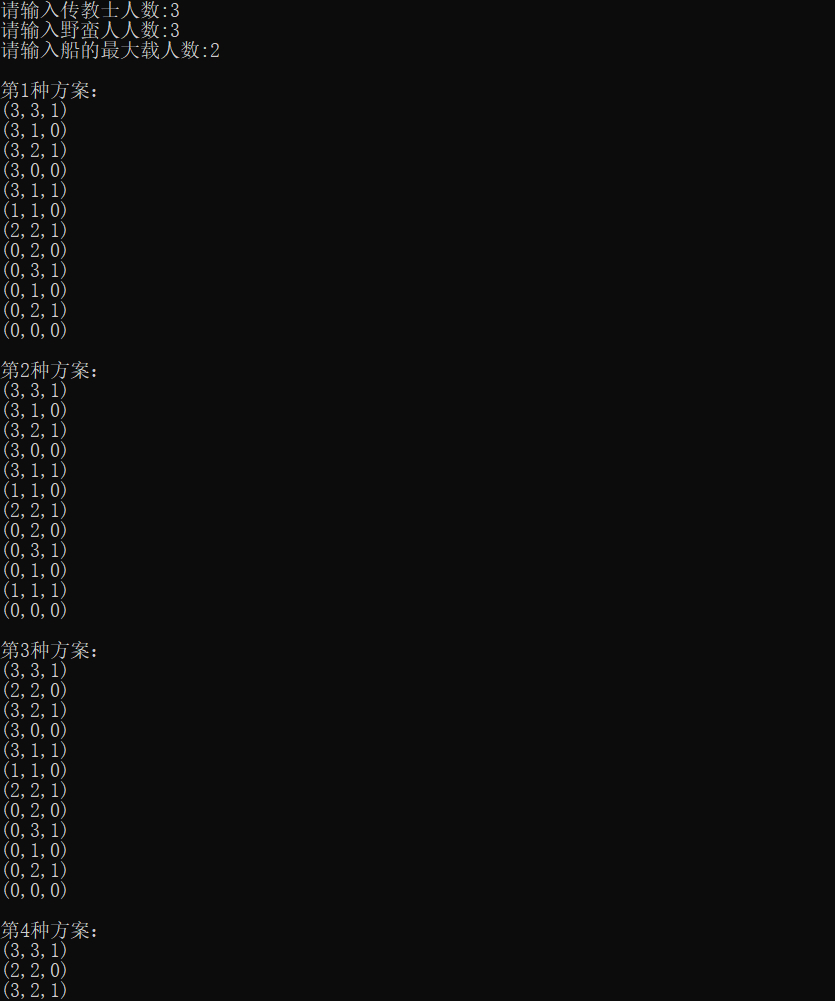


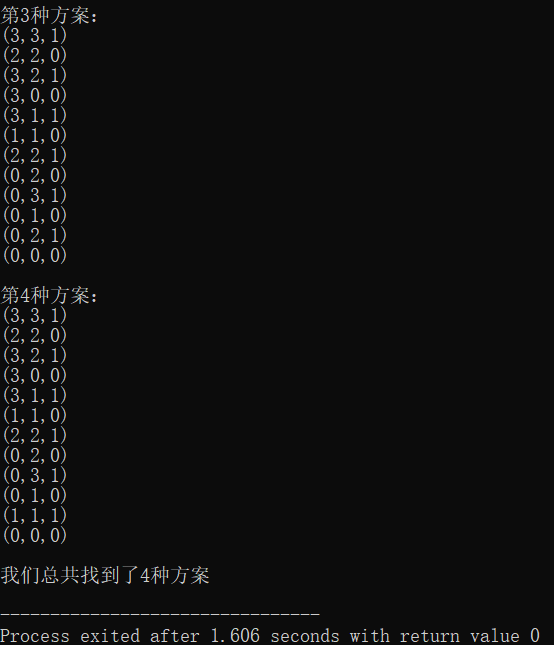
1. CrossRiver函数负责统计安全渡河的方案数，如无安全渡河的方案，也给出“很遗憾，我们找不到方案”的提示



**九、实验数据及结果分析：**

程序截图：





分析可知，程序结果符合前面给出的状态空间图的过程。

当m,c,n增大时，程序花费的时间大幅度增加：





**十、实验结论：**

该程序实现了状态空间法求解传教士过河问题，可手动输入m,c,n，然后输出安全渡河的过程；如无安全渡河的方案，也给出提示。

**十一、总结及心得体会：**

通过本次试验，我掌握了实现了状态空间法求解传教士过河问题和深度优先搜索算法。

**十二、对本实验过程及方法、手段的改进建议：**

此程序仅仅实现计算效果，寻找出所有可执行方案，并未开多线程进行优化。因此当计算数据过大时，计算需要很多时间，可能导致界面进程堵塞，会造成程序停止响应。

**报告评分：**

**指导教师签字：**