人工智能基础实验报告

刘砺志

（2014 级计算机 1 班 22920142203873）

本文是人工智能基础——用盲目搜索求解八数码问题的实验报告。

1 实验概述

本次实验要求选择一种盲目搜索算法（深度优先搜索或广度优先搜索）编程求解八数码问题（Eight Puzzle Problem）。所谓八数码问题，就是在 3×3 的方格棋盘上，摆放着 1 到 8 这八个数码，有 1 个方格是空的，如图 1所示，要求对空格执行空格左移、空格右移、空格上移和空格下移这四个操作使得棋盘从初始状态到目标状态。

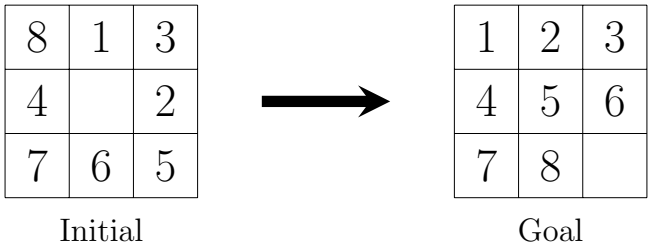

图 1 八数码问题

2 实验原理

2.1 深度优先搜索（Depth-First Search，DFS）

深度优先搜索的算法描述如下：

输入：一个图 G 和图 G 上的出发顶点 v

输出：从顶点 v 出发被访问到的所有顶点

一个递归实现的 DFS 伪代码如下：

DFS(G, v)

```
1  label v as discovered
2  for all edges from v to w in G.adjacentEdges(v) do
3      if vertex w is not labeled as discovered then
4          recursively call DFS(G, w)
```

另一个非递归版本的 DFS 描述如下：

DFS(G, v)

```
1  let S be a stack
2  S.push(v)
3  while S is not empty
4      v = S.pop()
5      if v is not labeled as discovered then
6          label v as discovered
7          for all edges from v to w in G.adjacentEdges(v) do
8              S.push(w)
```

2.2 广度优先搜索（Breadth-First Search，BFS）

广度优先搜索的算法描述如下：

输入：一个*搜索问题*。这个“搜索问题”是对有特别要求的实际搜索问题的抽象。

输出：一个从起始状态到目标状态依次访问所要执行动作的有序列表。

BFS(*problem*)

```
1  // a FIFO open_set
2  open_set = Queue()
3  // an empty set to maintain visited nodes
4  closed_set = set()
5  // a dictionary to maintain meta information (used for path formation)
6  meta = dict()              // key -> (parent state, action to reach child)

7  // initialize
8  start = problem.get_start_state()
9  meta[start] = (None, None)
10 open_set.enqueue(start)
```

```
11  while not open_set.is_empty()
12      parent_state = open_set.dequeue()

13      if problem.is_goal(parent_state)
14          return construct_path(parent_state, meta)

15          for (child_state, action) in problem.get_successors(parent_state)
16              if child_state in closed_set
17                  continue

18              if child_state not in open_set
19                  meta[child_state] = (parent_state, action)
20                  open_set.enqueue(child_state)

21          closed_set.add(parent_state)
```

```
CONSTRUCT-PATH(state, meta)
 1  action_list = list()

 2  while TRUE
 3      row = meta[state]
 4      if len(row) == 2
 5          state = row[0]
 6          action = row[1]
 7          action_list.append(action)
 8      else
 9          break

10      return action_list.reverse()
```

2.3 迭代加深深度优先搜索（Iterative Deepening Depth-First Search, IDDFS）

迭代加深深度优先搜索的算法描述如下，下面的伪代码使用了递归版本的深度限制 DFS（depth-limited DFS，称为 DLS）。

```
IDDFS(root)
1  for depth from 0 to ∞
2      found ← DLS(root, depth)
3      if found ≠ NIL
4          return found

DLS(node, depth)
1  if depth = 0 and node is a goal
2      return node
3  if depth > 0
4      for each child of node
5          found ← DLS(child, depth-1)
6          if found ≠ NIL
7              return
8  return NIL
```

2.4 双端搜索（Bidirectional Search）

双端搜索就是同时从初始状态和目标状态出发，寻找汇合点，连接两条子路径，构成完整的搜索路径。其算法描述如下：

```
BIDIRECTIONAL-SEARCH
 1  Q_I.insert(x_I) and mark x_I as visited
 2  Q_G.insert(x_G) and mark x_G as visited
 3  while Q_I not empty and Q_G not empty do
 4      if Q_I not empty
 5          x ← Q_I.get_first()
 6          if x = x_G or x ∈ Q_G
 7              return SUCCESS
 8          for all u ∈ U(x)
 9              x' ← f(x, u)
10              if x' not visited
11                  Mark x' as visited
12                  Q_I.insert(x')
13              else
14                  Resolve duplicate x'
```

```
    if Q_G not empty
        x' ← Q_G.get_first()
        if x' = x_I or x' ∈ Q_I
            return SUCCESS
        for all u^{-1} ∈ U^{-1}(x')
            x ← f^{-1}(x', u^{-1})
            if x not visited
                Mark x as visited
                Q_G.insert(x)
            else
                Resolve duplicate x
return FAILURE
```

3 实验结果

我采用 Python 3.6.0 编写了所有程序，并采用 Tkinter 模块制造了可视化交互界面，如图 2所示。

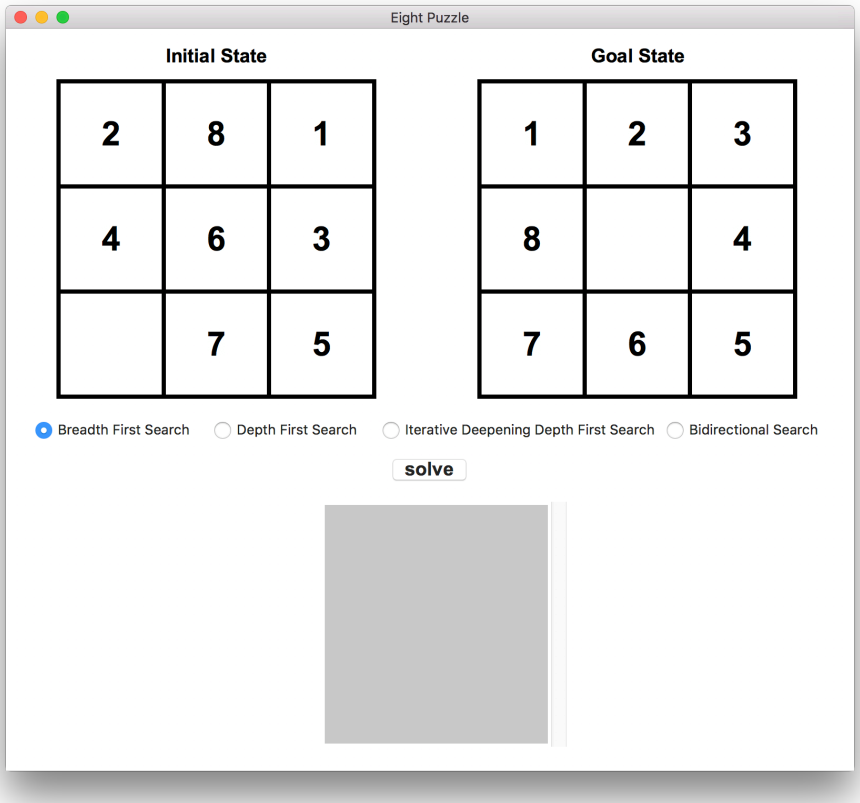

图 2　用户界面展示

首先测试 BFS，测试结果如图 3所示。

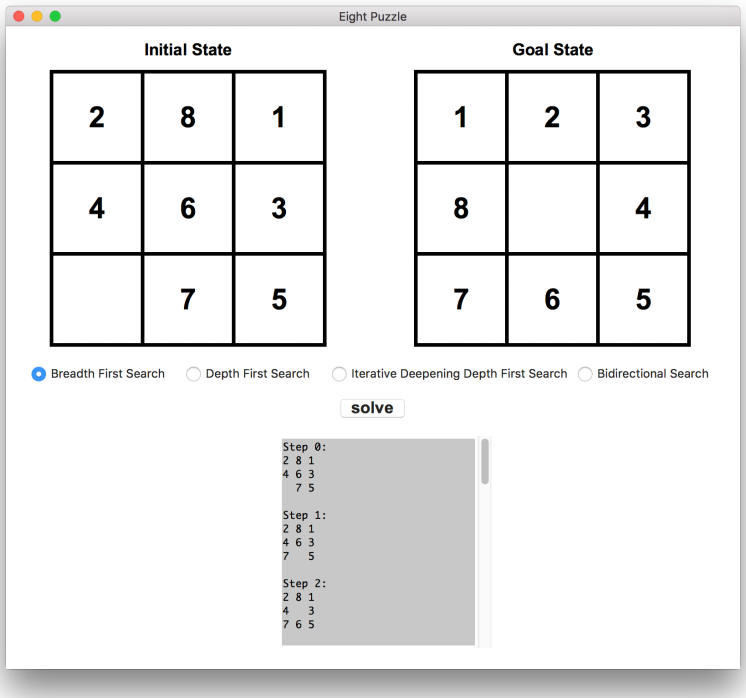

图 3　使用 BFS 解决八数码问题

接着测试 DFS，测试结果如图 4所示。

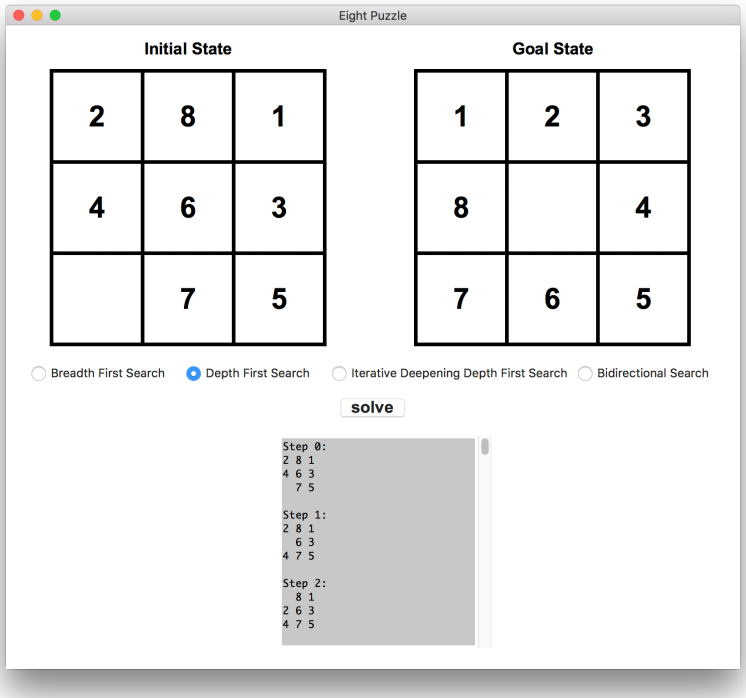

图 4　使用 DFS 解决八数码问题

接着测试 IDDFS，测试结果如图 5所示。

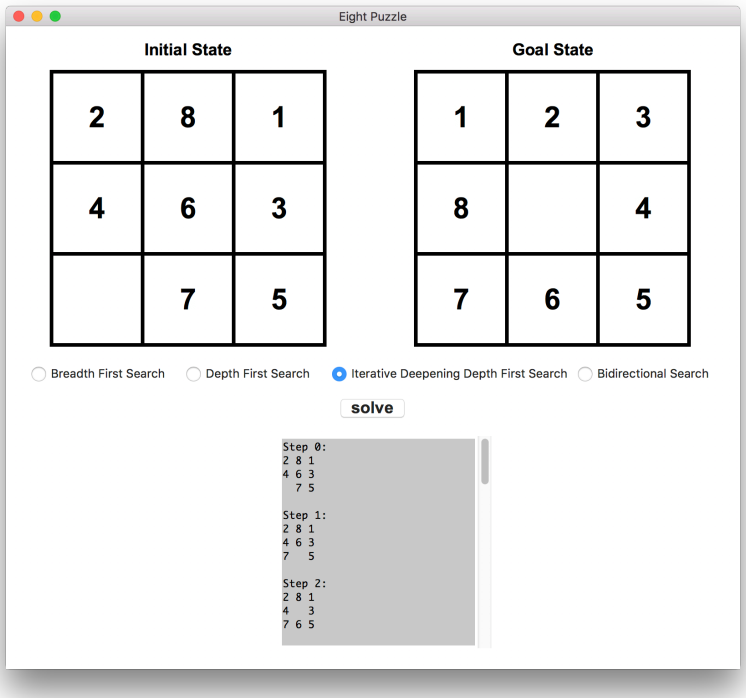

图 5　使用 IDDFS 解决八数码问题

最后测试双端搜索，测试结果如图 6所示。

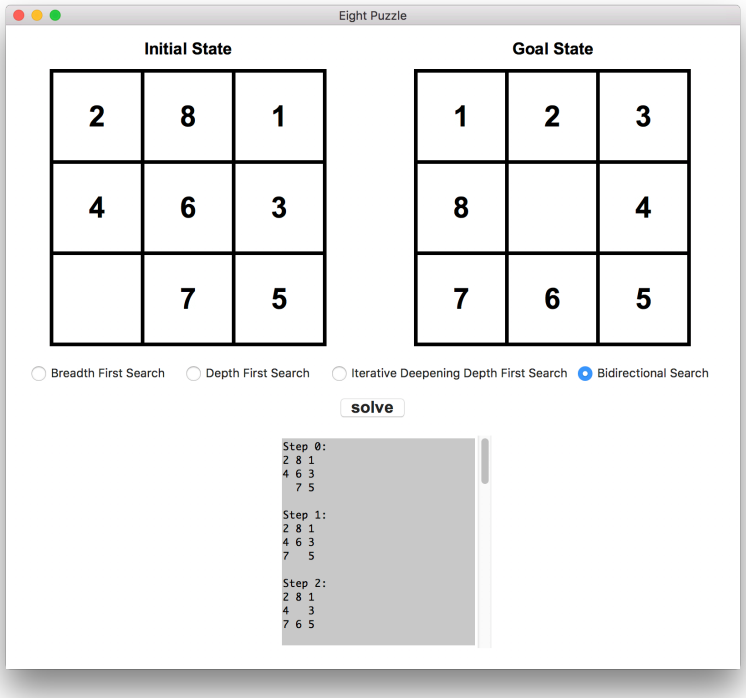

图 6　使用双端搜索解决八数码问题

另外，再展示一下非法的初始状态和目标状态下，程序给出的错误警报，如图 7和图 8所示。

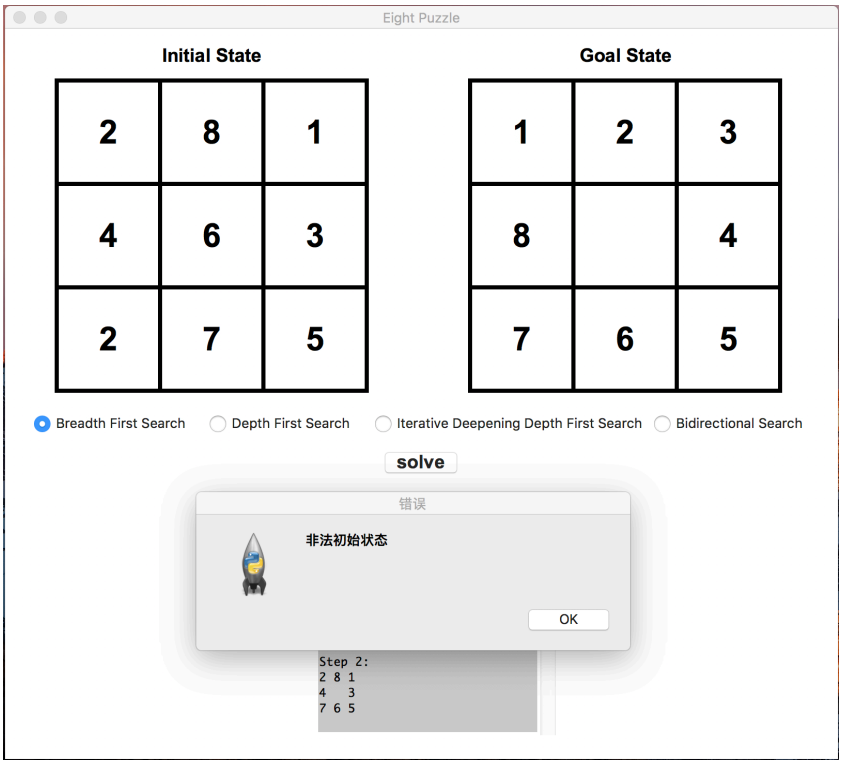

图 7　非法的初始状态

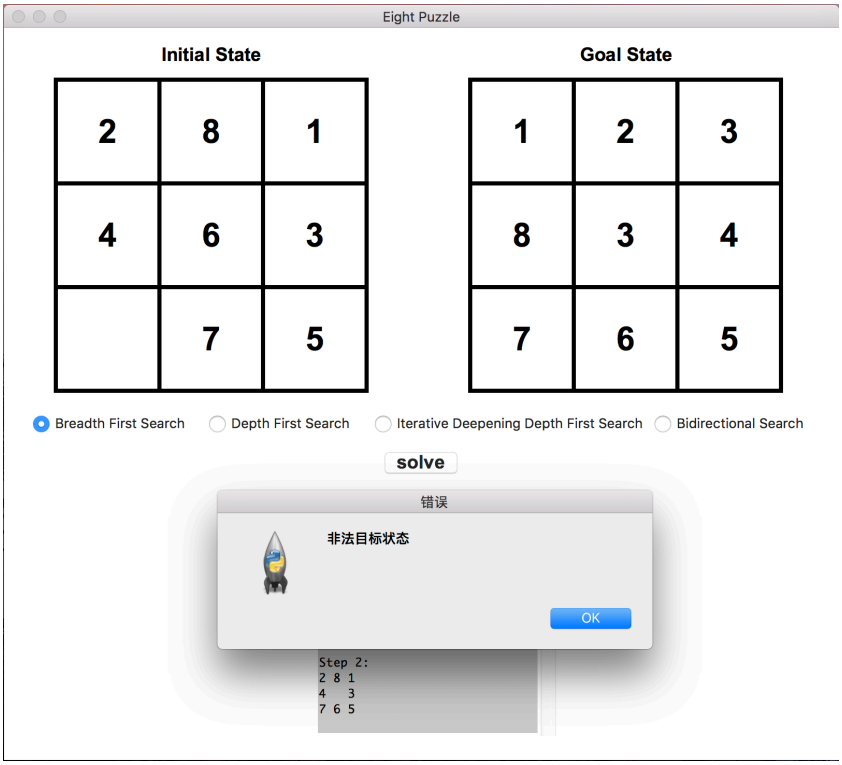

图 8　非法的目标状态

参考文献

[1] 人工智能, Nils J. Nilsson 著, 郑扣根, 庄越挺译, 潘云鹤校, 北京: 机械工业出版社, 2000 年 9 月

[2] https://en.wikipedia.org/wiki/Depth-first_search

[3] https://en.wikipedia.org/wiki/Breadth-first_search

[4] https://en.wikipedia.org/wiki/Iterative_deepening_depth-first_search

[5] http://www.geeksforgeeks.org/bidirectional-search/

[6] http://planning.cs.uiuc.edu/node50.html