实验一：知识表示、推理与搜索

实验目标

1. 掌握知识表示的基本方法（状态空间法、产生式系统）。
2. 实现经典搜索算法（广度优先搜索、A*算法）。
3. 分析不同搜索策略的效率差异。

实验环境

- **硬件**：Intel Core i5以上处理器，8GB内存，256GB SSD
- **软件**：Python 3.8+、PyCharm或Jupyter Notebook
- **依赖库**：`collections.deque`（广度优先搜索）、`heapq`（A*算法）、`time`（计时）

实验题目

1. **八数码难题求解**
   - 初始状态：`1 2 3 | 4 0 6 | 7 5 8`（0表示空格）
   - 目标状态：`1 2 3 | 4 5 6 | 7 8 0`
2. **传教士与野人问题**
   - 初始状态：`(3,3,1)`（3传教士、3野人、1船）
   - 目标状态：`(0,0,0)`（全部过河）

实验内容与步骤

1. 知识表示

- **状态空间法**：
  - 将问题抽象为状态节点（如八数码的棋盘布局）和状态转移（空格移动）。
  - 使用元组或列表表示状态（如`[(1,2,3),(4,0,6),(7,5,8)]`）。
- **产生式系统**：
  - 定义规则：
    - 八数码：`若空格在(i,j)，可移动到(i±1,j)或(i,j±1)`
    - 传教士与野人：`若船在左岸，可载1或2人到右岸`
  - 使用条件-动作对表示规则（如`if 空格在(1,1) then 可移动到(2,1)`）。

2. 搜索算法实现

- **广度优先搜索（BFS）**：
  - 使用队列存储待扩展节点，逐层扩展节点，记录访问路径。
  - 伪代码：

```
from collections import deque
def bfs(start, end):
    queue = deque([(start, 0)])  # (状态, 步数)
    visited = set()
    while queue:
        state, steps = queue.popleft()
        if state == end:
            return steps
        if state in visited:
            continue
        visited.add(state)
        # 生成新状态（示例代码略）
```

- **A*算法**：
  - 结合启发式函数（如曼哈顿距离）和代价函数（如移动步数）进行最优路径搜索。
  - 伪代码：

```
import heapq
def a_star(start, end):
    open_set = []
    heapq.heappush(open_set, (0 + heuristic(start, end), 0, start))  # (f=g+h, g, 状态)
    while open_set:
        _, g, state = heapq.heappop(open_set)
        if state == end:
            return g
        # 生成新状态（示例代码略）
```

3. 启发式函数设计

- **八数码**：
  - 曼哈顿距离：每个数字到目标位置的行/列距离之和。
  - 示例：`heuristic(state, end) = sum(abs(x1-x2) + abs(y1-y2) for all tiles)`
- **传教士与野人**：
  - 自定义规则：如 `heuristic = 左岸传教士数 + 左岸野人数`（惩罚非法状态）。

实验要求

1. **代码实现**：
   - 提交BFS和A*算法的完整代码，包含注释。
   - 使用Python类封装搜索算法，支持不同问题实例化。
2. **实验报告**：
   - **内容**：
     - 问题描述与知识表示方法。
     - 算法实现细节与关键代码。
     - 搜索结果（路径长度、扩展节点数、运行时间）。
     - BFS与A*的效率对比（如节点数、时间差异）。
   - **格式**：

- 使用Markdown或Word文档，包含代码截图与结果图表。

3. **扩展任务**：
   - 修改启发式函数，观察对A*性能的影响。
   - 实现深度优先搜索（DFS）或迭代加深搜索（IDS），对比结果。

实验评分标准

项目	分值	评分要点
知识表示设计	20	状态空间与产生式系统定义是否合理
算法实现	30	BFS与A*代码正确性、可读性
实验结果分析	30	搜索效率对比、启发式函数设计合理性
报告撰写	20	结构清晰、图表规范、结论明确

实验示例输出

1. **八数码结果**：

```
初始状态: [(1,2,3),(4,0,6),(7,5,8)]
目标状态: [(1,2,3),(4,5,6),(7,8,0)]
BFS路径长度: 20, 扩展节点数: 1000
A*路径长度: 20, 扩展节点数: 50
```

2. **传教士与野人结果**：

```
初始状态: (3,3,1)
目标状态: (0,0,0)
BFS非法状态数: 15
A*非法状态数: 5
```

实验时间安排

阶段	任务
知识表示设计	状态空间与产生式系统建模
算法实现	BFS与A*代码编写与调试
实验结果分析	对比不同算法性能，优化启发式函数
报告撰写	整理代码、图表与结论，提交实验报告

注意事项

1. **代码规范**：使用函数封装，避免全局变量。
2. **启发式函数**：需保证不低估代价（A*正确性前提）。
3. **非法状态处理**：在传教士与野人问题中，需实时检查状态合法性。

通过本实验，学生将掌握知识表示与搜索算法的核心原理，为后续复杂问题求解奠定基础。