人工智能实验报告

实验一　知识表示、推理与搜索

学院：计算机与通信工程　　专业：计算机科学与技术　　班级：计 221

姓名：乔彦博　　学号：U202242223　　日期：2025.4.27

实验目标

1. 熟练掌握知识表示的多种基本方法，包括状态空间法、产生式系统等，并能根据不同问题灵活选择和运用。
2. 准确实现经典搜索算法，如广度优先搜索（**BFS**）、A* 算法（启发式函数）等，理解其算法原理和执行过程。
3. **深入分析不同搜索策略在效率方面的差异，通过实验结果进行可视化、对比和评估。**

实验内容（根据实验要求文档）

1. 问题描述：
   - **八数码问题**：将 3×3 棋盘视为 9 个位置的状态空间，每个状态用矩阵形式 $((1,2,3),(4,0,6),(7,5,8))$ 表示，其中 0 代表空格。空格可与上下左右的数字交换位置，从而产生新状态。
   - **传教士与野人问题**：用三元组 $(m_{\text{left}}, w_{\text{left}}, \text{boat\_side})$ 表示左岸传教士人数、野人人数及船的位置（0 表示在右岸，1 表示在左岸）。遵循以下约束：
     - 船一次可载 1–2 人；
     - 任一岸上若有传教士，其数量必须不少于野人数量。
   - **搜索算法实现要求**：
     - **广度优先搜索（BFS）**：保证找到最短路径解，需维护已访问状态集合，记录扩展节点数与解路径长度；
     - **A* 算法**：在 BFS 基础上集成启发式函数，八数码问题使用曼哈顿距离，传教士与野人问题使用左岸总人数作为估计值；
     - 对比分析两种算法在节点扩展量、运行时间上的差异，评估启发式函数对搜索效率的影响。
2. **实现工具**：
   - **编程语言**　Python 3.11.12（macOS Apple Silicon 原生支持，单线程即可满足实验规模）；
   - **主要库依赖**
     - `collections.deque`：O(1) 双端队列，用于 BFS；
     - `heapq`：二叉最小堆，实现 A* 优先队列；
     - `time`：高精度计时（`perf_counter()`）；

  - `numpy`：矩阵操作（八数码可选）；
  - `matplotlib`：性能结果可视化。
- **开发环境** VS Code + Jupyter Notebook，启用 `-Xfrozen_modules` 以缩短启动时间。

3. **实现方案**：

(a) **状态设计**

- 八数码：定长元组 `Board = Tuple[Tuple[int,...],...]`，并预生成 `GOAL_POS` 以 O(1) 计算曼哈顿距离；
- 传教士与野人：三元组 `State = (ml, cl, boat)`，总合法状态 ≤ 32，便于调试。

(b) **搜索骨架**

- 公共父类 `Node` 保存 `state, g, parent`，使用 `dataclass(slots=True)`；
- BFS 用 `deque`，层次展开；A* 用 `heapq`，按 $f = g + h$ 取最小。

(c) **启发式函数 h**

- 传教士与野人：$h = ml + cl + \mathbf{1}_{\text{boat 在右}}$；
- 传教士与野人：$h = ml + cl + \mathbf{1}_{\text{boat 在右}}$。

(d) **复杂度与边界条件**

- 八数码判重基于棋盘哈希；传教士问题每次生成最多 5 个后继并即时过滤非法状态；
- 若输入即为目标，算法 O(1) 返回；若无解（如八数码奇偶错位）抛出 `RuntimeError`。

4. **实现内容与实验结果**：

(a) **核心代码结构**

- `eight_puzzle_search.py`：实现 BFS 与 A*，附 `demo()`；
- `missionaries_cannibals_search.py`：同上；
- 统一接口：`bfs(start)` / `a_star(start)` → `(path, expanded, elapsed)`。

(b) **实验流程**

i. 设定初始状态；
ii. 分别调用 BFS 与 A*；
iii. 记录路径长度、扩展节点数、耗时；
iv. 可多次测量取均值；
v. 用 `matplotlib` 绘制条形图比较。

(c) **实验结果摘要**

问题	算法	路径长度	扩展节点	运行时间/s
八数码	BFS	2	2	0.0000
八数码	A*	2	2	0.0000
传教士与野人	BFS	11	13	5.1×10^{-5}
传教士与野人	A*	11	14	4.4×10^{-5}

可视化示例代码节选：

```
import matplotlib.pyplot as plt
labels = ['8-Puzzle', 'Missionaries']
bfs_nodes = [2, 13]
astar_nodes = [2, 14]
x = range(len(labels))
plt.bar(x, bfs_nodes, label='BFS')
plt.bar(x, astar_nodes, bottom=bfs_nodes, label='A*')
plt.ylabel('Expanded Nodes')
plt.xticks(x, labels)
plt.legend(); plt.show()
```

实验总结

1．实验结论

BFS 作为无启发盲目搜索，在浅层或状态空间极小问题上已能快速找到最优解；但随着深度增长，其节点爆炸现象明显。A* 依赖启发式函数质量：

- 对八数码，简单曼哈顿距即可显著剪枝（在更深乱序样例中扩展节点可减少数十倍）；
- 对传教士与野人，由于状态空间仅 32 个结点，弱启发式并未体现优势，甚至出现 1 个结点的轻微反超。

2．启发式函数的作用曼哈顿距满足可采纳（不高估）与一致性（$h(n) \le c(n,n') + h(n')$），保证 A* 找到最优解且不重复扩展。若以 Pattern Database 等更强启发式替换，可继续降低时间／空间复杂度，体现“领域知识 $\Rightarrow$ 搜索效率”这一核心思想。

3．存在问题与改进方向

(a) **随机化难例**：八数码随机打乱 20 步，能更直观展示 A* 优势；

(b) **启发式优化**：传教士问题可改用 $\lceil (ml + cl)/2 \rceil$ 作为最少渡河轮次下界；

(c) **算法拓展**：实现 IDS（深度迭代加深）或双向 A*，在有向无障碍图中可进一步加速；

(d) **性能评估**：加入峰值队列长度及内存 Profiling，获得更全面的资源消耗曲线。

4．收获与体会

通过本实验深刻体会到：“搜索策略选择 + 启发式设计”才是 AI 问题求解效率的关键。掌握统一的搜索框架后，投入时间打磨启发式往往收益最高。此外，严谨的实验统计与可视化可帮助我们快速定位瓶颈、迭代算法。