**全局搜索算法的实验分析**

**实验零：补全代码并填写到实验报告中（报告中只需填写C++实现）**

breadth\_first\_search.hpp

// 扩展待访问的结点集合：

【copy code here】

// ……

depth\_first\_search.hpp

// ……

heuristic\_search.hpp

// ……

**实验一：程序语言和I/O对程序运行时间的影响**

实验目的：了解C++和Python语言程序的运行效率及I/O在程序运行时间中的占比

实验步骤：分别运行2次C++和Python版本的宽度优先搜索解11和12皇后问题的样例程序，一次在显示器上输出解，一次注释掉输出解的语句，比较它们找到全部解的运行时间。全部采用树搜索。

实验结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 时间（秒） | C++程序 | | Python程序 | |
| 输出解 | 不输出解 | 输出解 | 不输出解 |
| 11 |  |  |  |  |
| 12 |  |  |  |  |

实验结果分析：

**实验二：深度优先和宽度优先的时间效率比较**

实验目的：比较深度优先和宽度优先算法在求解N皇后问题上的时间效率

实验步骤：分别运行C++和Python版本的宽度优先和深度优先搜索算法求解N皇后问题，注释掉输出解的语句，比较它们找到全部解的运行时间。全部采用树搜索

实验结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 时间（秒） | 宽度优先 | | 深度优先 | |
| C++ | Python | C++ | Python |
| 8皇后 |  |  |  |  |
| 9皇后 |  |  |  |  |
| 10皇后 |  |  |  |  |
| 11皇后 |  |  |  |  |
| 12皇后 |  |  |  |  |
| 13皇后 |  |  |  |  |
| 14皇后 |  |  |  |  |
| 15皇后 |  |  |  |  |
| …… |  |  |  |  |

实验结果分析：

**实验三：深度优先和宽度优先的空间效率比较**

实验目的：比较深度优先和宽度优先算法在求解N皇后问题上的空间效率

实验步骤：分别运行C++和Python版本的宽度优先和深度优先搜索算法求解N皇后问题，注释掉输出解的语句，比较它们找到全部解时开节点集同时存储的最大节点数。全部采用树搜索

实验结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 空间效率  （节点数） | 宽度优先 | | 深度优先 | |
| C++ | Python | C++ | Python |
| 8皇后 |  |  |  |  |
| 9皇后 |  |  |  |  |
| 10皇后 |  |  |  |  |
| 11皇后 |  |  |  |  |
| 12皇后 |  |  |  |  |
| 13皇后 |  |  |  |  |
| 14皇后 |  |  |  |  |
| 15皇后 |  |  |  |  |
| …… |  |  |  |  |

实验结果分析：

**实验四：比较不同算法求解最短路径问题**

实验目的：比较分析一致代价、贪心、A\*算法求解最短路径问题的差异

实验步骤：在根据要求完善代码的基础上，分别运行一致代价、贪心、A\*算法求解最短路径问题的程序（C++或者Python，语言不限），填写并分析实验结果。

实验结果：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 进入优先队列顺序 | 输出的解路径 | 解路径的花费 |
| 一致代价 |  |  |  |
| 贪心搜索 |  |  |  |
| A\* |  |  |  |

实验结果分析：

1. 截图【一致代价搜索：进优先队列顺序、输出的解路径、解路径的花费】
2. 截图【贪心搜索：进优先队列顺序、输出的解路径、解路径的花费】
3. 截图【A\*搜索：进优先队列顺序、输出的解路径、解路径的花费】