湖南大學

人工智能
实验报告

姓名：杨杰
学号：201908010705
班级：计科 1907

实验二：约束满足问题

一、 实验名称

实验二：约束满足问题

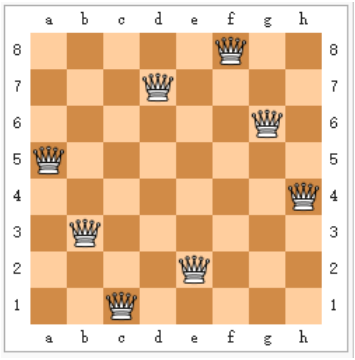
二、 实验目的

1、 求解约束满足问题；

2、 使用回溯搜索算法求解八皇后问题。

三、 实验内容

实训内容：2-4 第六章 约束满足问题



四、 开发环境

OS: Ubuntu20.04

Language: C++

IDE: VSCode

五、 算法原理及步骤

算法原理：

回溯算法实际上是一个类似枚举的搜索尝试过程，主要是在搜索尝试过程中寻找问题的解，当发现已不满足求解条件时，就回溯返回，尝试别的路径。

回溯法是一种选优搜索法，按选优条件向前搜索，以达到目标。但当探索到某一步时，发现原先选择并不优或达不到目标，就退回一步重新选择，这种走不通就退回再走的技术为回溯法，而满足回溯条件的某个状态的点称为“回溯点”。

许多复杂的，规模较大的问题都可以使用回溯法，有“通用解题方法”的美称。

算法步骤：

1. 首先将数组 a、b、c、d 全部初始化为 0，表示行、列、对角线均未摆放皇后；
2. 从第一行开始，把皇后依次放在第 1 列到第 8 列；
3. 每次放置皇后后，在 b、c、d 数组中标记不能放置皇后的位置（不同列，也不在对角线上，因为接着会到下一行，所以不需要标记当前行）；
4. 下一行放置皇后时，根据 b、c、d 数组中的空位置放置皇后，若无空位置，返回至上一行，重新摆放；若有空位置则继续探索下一层；
5. 重复以上步骤 2~4 直到找到最后一行的放置位置；
6. 当一次搜索完成后执行上面所有步骤继续搜索其它放置路线。

六、 实验代码

```
void searchh(int i)
{
    for (int j = 1; j <= 8; j++) //8 个位置遍历一遍
    {
        if ((!b[j]) && (!c[i + j - 1]) && (!d[i - j + 8]))
        {
            /********** Begin **********/
            if (i == 8)
            {
                sum++;
                return;
            }
            //改位置信息
            b[j] = 1;
            c[i + j - 1] = 1;
            d[i - j + 8] = 1;
            searchh(i + 1); //下一步计划
            //回溯，将位置信息改回原来的样子
            b[j] = 0;
            c[i + j - 1] = 0;
            d[i - j + 8] = 0;
            /********** End **********/
        }
    }
}
```

算法复杂度分析：

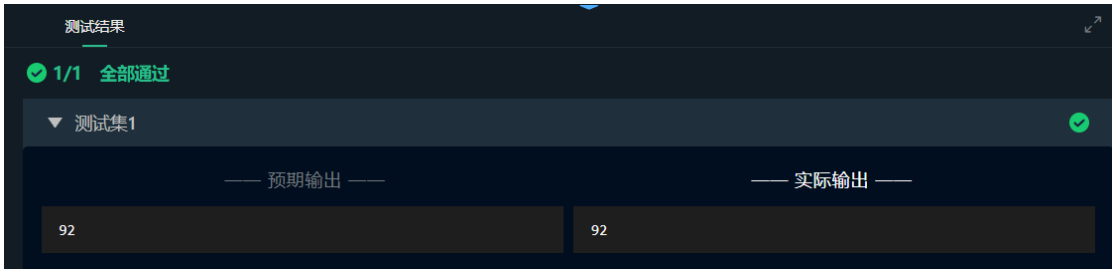
时间复杂度：

最坏的情况：每一行有 n 种情况，有 n 行，所以时间复杂度为 O(n^n)。
但是由于回溯法会提前判断并舍弃一些情况，使得时间复杂度并没有想象中那么高。

空间复杂度：

理论上应该创建一个二维数组来表示棋盘，但实际可以通过算法，用一个一维数组即可解决问题，所以空间复杂度是 O(n).

运行结果：



七、 实验结果与分析

实验结果：

```
1-8
2-4
3-1
4-3
5-6
6-2
7-7
8-5

Number of solutions:92
time: 0.774ms
```

分析：

							Q
			Q				
Q							

		Q					
					Q		
	Q						
						Q	
				Q			

经验证，满足不同行、不同列、不同对角线。

PS:由于空间有限，在此仅给出一种情况并作分析。全部情况及程序代码请查看说明文档。

说明文档：

（请使用 markdown 编辑器打开 lab2.md[在本PDF附件中]，或者使用浏览器打开 lab2.html）


lab2.md


lab2.html

八、 实验总结

实验收获：

通过这个实验，我深入学习了解了回溯法。

回溯法思路的简单描述是：把问题的解空间转化成图或者树的结构表示，然后使用深度优先搜索策略进行遍历，遍历的过程中记录和寻找所有可行解或者最优解。

基本思想类同于：

- 图的深度优先搜索
- 二叉树的后序遍历

经过这次实验，我对于用回溯法求解八皇后问题的相关代码已基本熟悉，算法知识得到了复习与巩固。在写代码与调试的过程中，在解决问题过程中，丰富了个人编程的经历和经验，提高了个人解决问题的能力。

难点重点讨论：

回溯法按深度优先策略搜索问题的解空间树。首先从根节点出发搜索解空间树，当算法搜索至解空间树的某一节点时，先利用剪枝函数判断该节点是否可行（即能得到问题的解）。如果不可行，则跳过对该节点为根的子树的搜索，逐层向其祖先节点回溯；否则，进入该子树，继续按深度优先策略搜索。

回溯法的基本行为是搜索，搜索过程中使用剪枝函数来避免无效的搜索。剪枝函数包括两类：1. 使用约束函数，剪去不满足约束条件的路径；2.使用限界函数，剪去不能得到最优解的路径。

虽然回溯法看起来时间复杂度很高，但是因为有剪枝函数，所以实际的运行时间并不长，问题的关键在于如何定义问题的解空间，转化成树（即解空间树）。