重 庆 大 学

学 生 实 验 报 告

实验课程名称 算法设计与分析

开课实验室 DS1501

学 院 大数据与软件学院 年级 22 专业班 软件工程 2 班

学 生 姓 名 潘锄葳 学 号 20221982

开 课 时 间 2023 至 2024 学年第 2 学期

总 成 绩	
教师签名	付春雷

大数据与软件学院制

《算法设计与分析》实验报告

开课实验室：DS1501　　2024 年 6 月 4 日

学院	大数据与软件学院	年级、专业、班	22 软工 2 班	姓名	潘铷葳	成绩	
课程名称	算法设计与分析	实验项目名　称	回溯法实验和分枝限界法实验	指导教师	付春雷		
教师评语	教师签名： 年　月　日						

一、实验目的

- 掌握回溯法算法框架，掌握分枝限界法的特点和算法框架。
- 熟练掌握“满足方程解问题求解算法”的实现，熟练掌握“4 皇后问题求解算法”的实现。
- 主要任务：实现教材配套实验指导书“第 2 章 2.5.4 小节 求解满足方程解问题”和“第 2 章 2.6.1 小节 求解 4 皇后问题”。

二、使用仪器、材料

PC 微机；
Windows 操作系统，VS2015 编译环境（不限）；

三、实验步骤

问题一：
1.定义输出函数：定义一个函数 print，用来输出满足条件的解。
2.递归求解函数：定义一个递归函数 solve 来生成所有变量的排列，并检查每个排列是否满足方程条件。
3.主函数：在主函数中初始化变量数组并调用递归求解函数，最终输出所有满足条件的解。

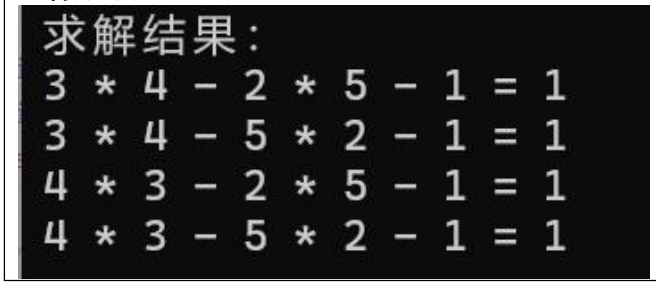
问题二：
（1）队列式分支限界法
1.定义节点结构：定义一个结构体 node 来表示搜索树中的节点，包括节点编号、行号和存储当前解的列号数组。
2.判断函数：定义一个函数 judge 用于判断当前节点是否符合皇后问题的要求，即是否存在冲突。
3.求解函数：定义一个函数 queen 用于执行队列式分枝限界法，利用队列进行广度优先搜索。
4.主函数：调用求解函数并输出结果。
（2）优先队列式分支限界法
1.定义节点结构：定义一个结构体 node 来表示搜索树中的节点，包括节点编号、行号和存储当前解的列号数组，并重载小于运算符用于优先队列的比较。
2.判断函数：定义一个函数 judge 用于判断当前节点是否符合皇后问题的要求，即是否存在冲突。
3.求解函数：定义一个函数 queen 分别用于执行队列式和优先队列式分枝限界法，利用队列和优先队列进行广度优先搜索。
4.主函数：调用求解函数并输出结果。

四、实验过程原始记录(数据、图表、计算等)

问题一：

```cpp
//实验4. 求解满足方程解问题
//编写一个实验程序,求出a,b、c ,d ,e,满足ab-cd-e=1方程,其中所有变量的取值为1～5并且均不相同。

#include <iostream>
#include <algorithm>
using namespace std;

// 输出满足条件的排列
void print(int a[]) {
    cout << a[0] << " * " << a[1] << " - " << a[2] << " * " << a[3] << " - " << a[4] << " = 1" << endl;
}

// 递归求解函数，生成所有排列并检查
void solve(int a[], int i) {
    if (i == 5) {
        if (a[0] * a[1] - a[2] * a[3] - a[4] == 1) {
            print(a);
        }
        return;
    } else {
        for (int j = i; j < 5; ++j) {
            swap(a[i], a[j]);
            solve(a, i + 1);
            swap(a[i], a[j]);
        }
    }
}

int main() {
    int a[] = {1, 2, 3, 4, 5};
    cout << "求解结果: " << endl;
    solve(a, 0);
    return 0;
}
```

运行结果如下：



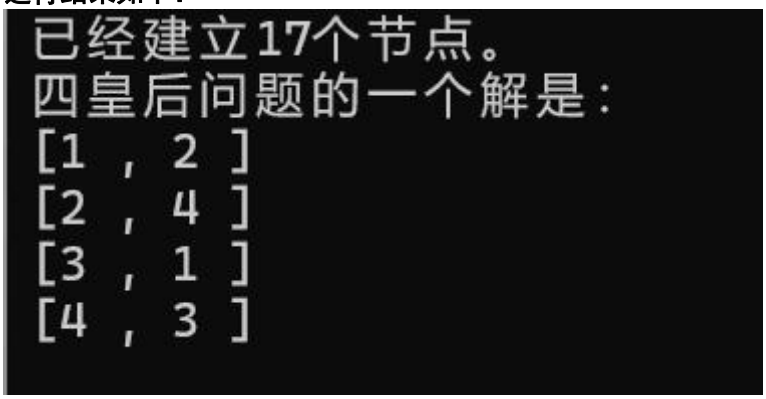
代码解释：
1. print 函数：负责输出满足条件的解。
2. solve 函数：
基于递归的方法，生成所有可能的排列。
在每次排列结束时（即 i == 5），检查是否满足方程条件。
如果满足条件，则调用 print 函数输出结果。
3. main 函数：初始化数组并调用 solve 函数开始递归求解过程。

问题二：
1. 队列式分支限界法代码

```
//实验4. 求解4皇后问题
//编写一个实验程序,采用队列式和优先队列式分枝限界法求解4皇后问题的一个解,
//分析这两种方式的求解过程,比较创建的队列结点个数。

//队列式分支限界法
#include <iostream>
#include <vector>
#include <queue>
#include <cmath>
using namespace std;

struct Node {
    int no; // 编号
    int row; // 行号
    vector<int> cols; // 存储列编号
};

// 判断位置 (i, j) 是否符合要求
bool judge(const vector<int>& cols, int i, int j) {
    for (int k = 0; k < i; k++) {
        if (cols[k] == j || abs(k - i) == abs(cols[k] - j))
            return false;
    }
    return true;
}

// 队列式分枝限界法求解4皇后问题
void queen() {
    int count = 1;
    Node e, e1;
    queue<Node> q;

    e.no = count++; // 建立根节点，编号从1开始，行号是-1
    e.row = -1;
    q.push(e);

    while (!q.empty()) {
```

```
            e = q.front();
            q.pop();

            if (e.row == 3) { // 到达叶子节点，即找到一个解
                cout << "已经建立" << count - 1 << "个节点。" << endl;
                cout << "四皇后问题的一个解是：" << endl;
                for (int i = 0; i < 4; i++)
                    cout << "[" << i + 1 << " , " << e.cols[i] + 1 << " ]" << endl;
                return;
            } else {
                for (int j = 0; j < 4; j++) {
                    int i = e.row + 1;
                    if (judge(e.cols, i, j)) {
                        e1.no = count++;
                        e1.row = i;
                        e1.cols = e.cols;
                        e1.cols.push_back(j);
                        q.push(e1);
                    }
                }
            }
        }
    }

    int main() {
        queen();
        return 0;
    }
```

运行结果如下：



代码解释：

1. Node 结构体：包含节点的编号、行号和列号数组，用于表示当前节点的状态。
2. judge 函数：检查当前节点位置是否符合皇后问题的要求，防止皇后互相攻击。
3. queen 函数：

使用广度优先搜索，通过队列逐层扩展节点。

每生成一个新的节点时，检查其是否符合条件，如果符合则继续扩展。

找到解时输出结果并终止搜索。

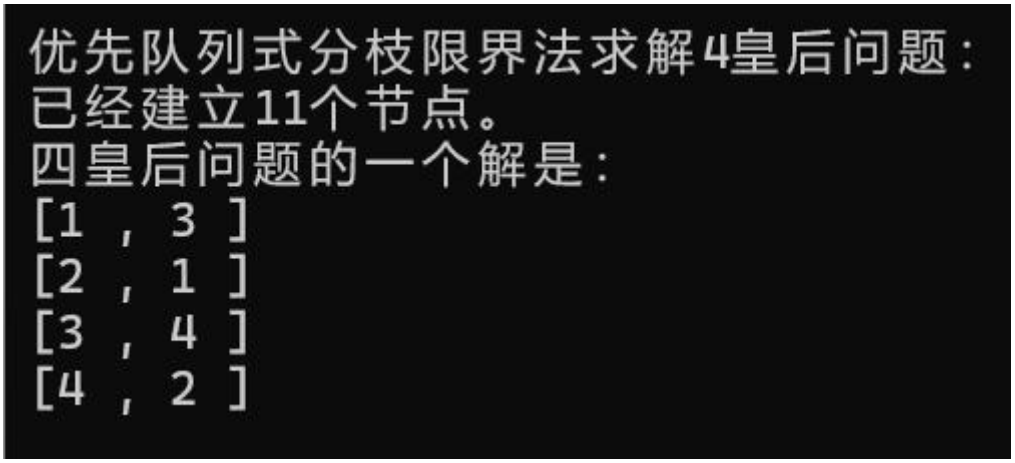
4. main 函数：调用 queen 函数，开始求解过程并输出结果。

2. 优先队列式分支限界法代码

```cpp
#include <iostream>
#include <vector>
#include <queue>
#include <cmath>
using namespace std;

struct Node {
    int no; // 节点编号
    int row; // 当前行号
    vector<int> cols; // 存储列编号

    // 重载小于运算符用于优先队列的比较
    bool operator<(const Node& s) const {
        return row < s.row;
    }
};

// 判断位置 (i, j) 是否符合要求
bool judge(const vector<int>& cols, int i, int j) {
    for (int k = 0; k < i; k++) {
        if (cols[k] == j || abs(k - i) == abs(cols[k] - j))
            return false;
    }
    return true;
}

// 优先队列式分枝限界法求解 4 皇后问题
void queen_priority_queue() {
    int i, j, count = 1;
    Node e, e1;
    priority_queue<Node> q;

    e.no = count++;
    e.row = -1;
    q.push(e);

    while (!q.empty()) {
        e = q.top();
        q.pop();

        if (e.row == 3) { // 到达叶子节点，即找到一个解
            cout << "已经建立" << count - 1 << "个节点。" << endl;
            cout << "四皇后问题的一个解是: " << endl;
            for (i = 0; i < 4; i++)
                cout << "[" << i + 1 << " , " << e.cols[i] + 1 << " ]"
```

```
    << endl;
46.                 return;
47.             } else {
48.                 for (j = 0; j < 4; j++) {
49.                     i = e.row + 1;
50.                     if (judge(e.cols, i, j)) {
51.                         e1.no = count++;
52.                         e1.row = i;
53.                         e1.cols = e.cols;
54.                         e1.cols.push_back(j);
55.                         q.push(e1);
56.                     }
57.                 }
58.             }
59.         }
60.     }
61.
62.     int main() {
63.         cout << "优先队列式分枝限界法求解 4 皇后问题: " << endl;
64.         queen_priority_queue();
65.         return 0;
66.     }
```

运行结果如下：



代码解释：

1.Node 结构体：包含节点的编号、行号和列号数组，用于表示当前节点的状态，并重载 operator< 运算符用于优先队列的比较。

2.Judge 函数：检查当前节点位置是否符合皇后问题的要求，防止皇后互相攻击。

3.queen_priority_queue 函数：

使用优先队列，通过优先级队列方式扩展节点。

每生成一个新的节点时，检查其是否符合条件，如果符合则继续扩展。

找到解时输出结果并终止搜索。

4.main 函数：调用 queen_priority_queue 函数，开始求解过程并输出结果。

五、实验结果及分析

结果都已对应显示在原始数据记录中，结果都与预期的分析符合。