**暨南大学本科实验报告专用纸**

课程名称 算法分析与设计实验 成绩评定

实验项目名称 指导教师 李展 实验地点 N116

实验项目编号 6-6 实验项目类型 综合型

学生姓名 张瑞鹏 学号 2020101124

学院 信息科学技术学院 系 计算机 专业 计算机科学与技术

实验时间 2022 年 月 日 午～ 月 日 下 午

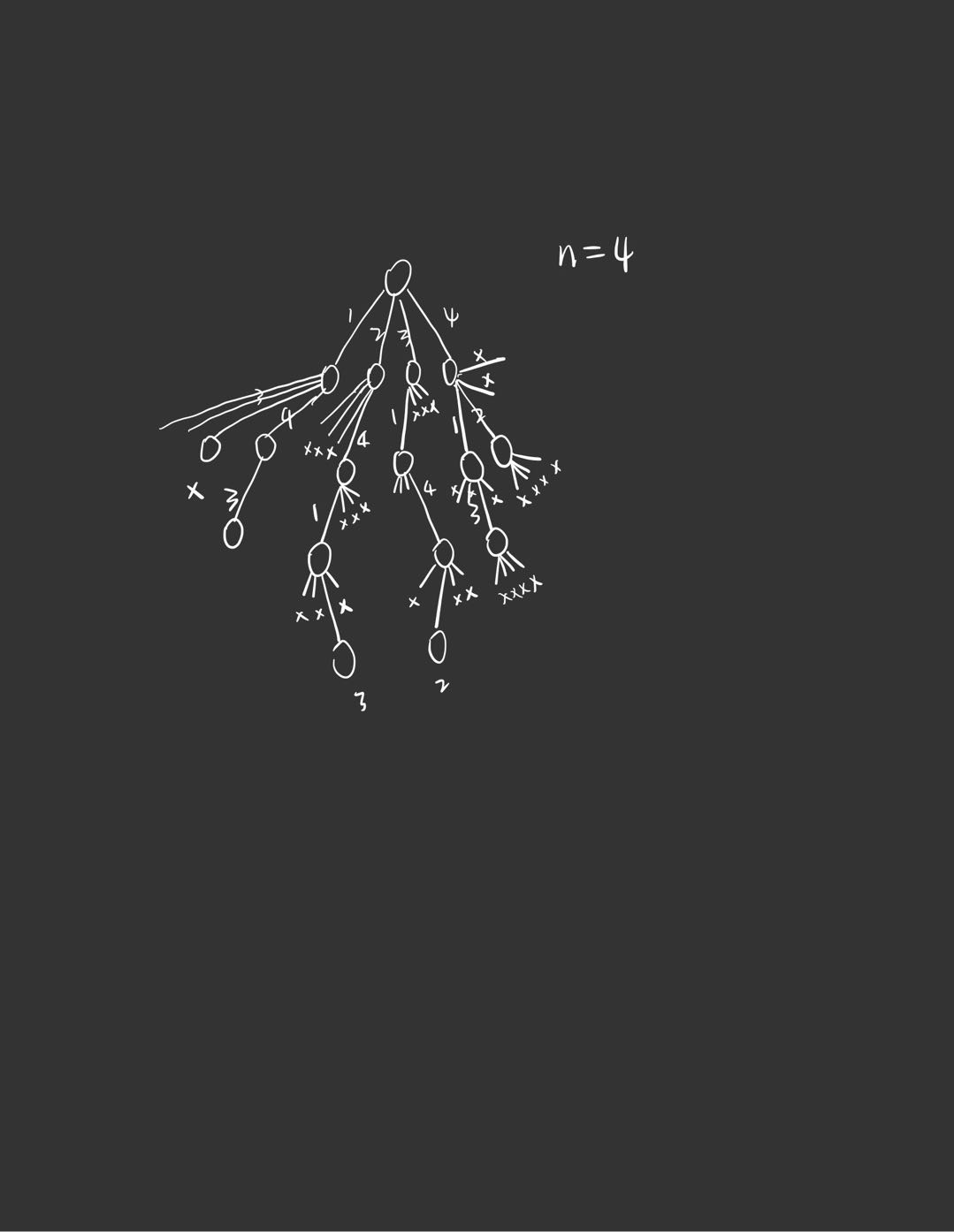
**（一）问题描述**

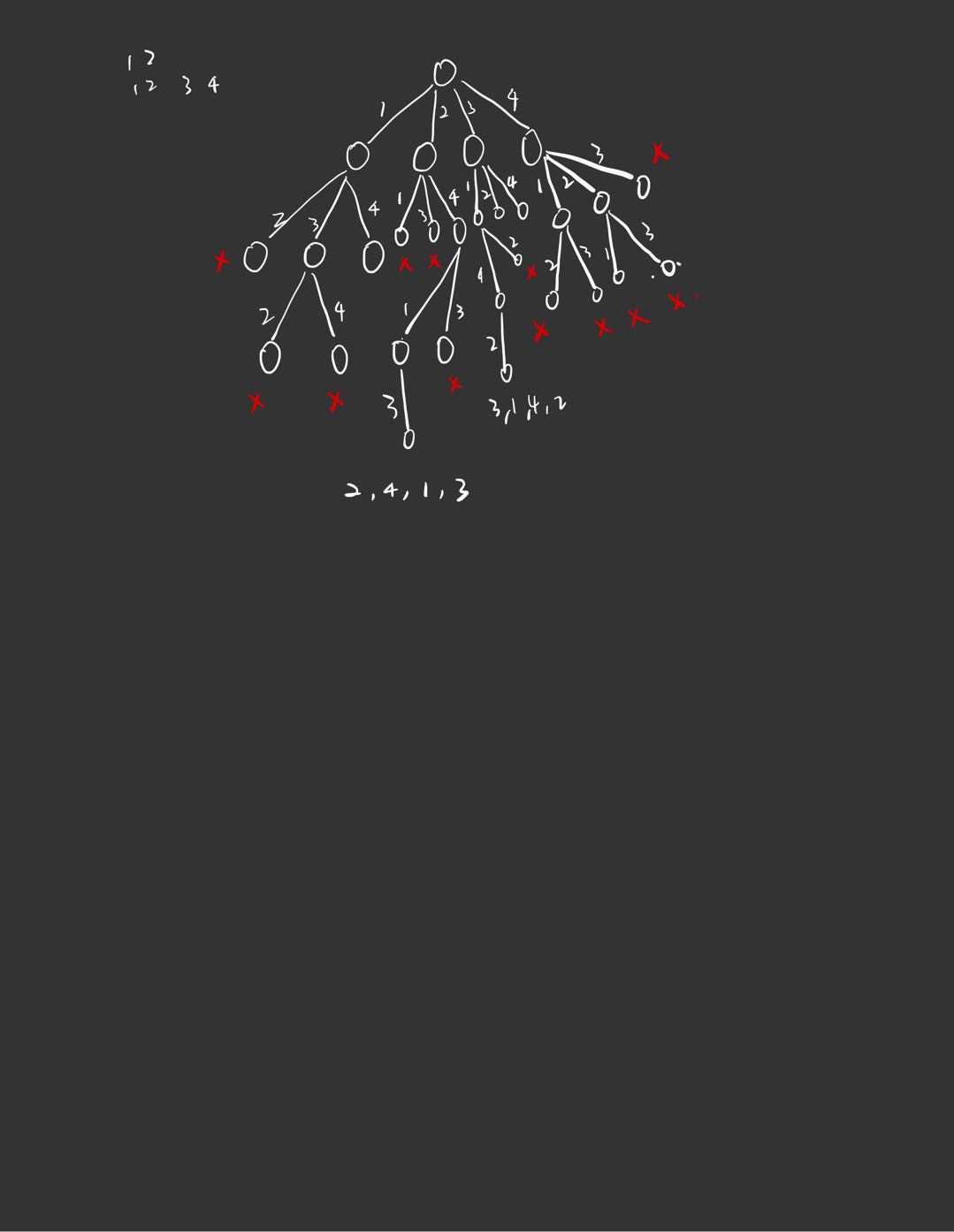
**N皇后问题的分支界限法模式**

**（二）算法思路**（用文字简单说明）

**N皇后问题之前用回溯算法写过，分别用过排列树方式还有子集树方式，回溯算法的话的树是虚拟的，而分支限界方法的树确实实实在在存在的，因此，相当于是牺牲了空间，换取在优先级上的提升，但是对于并没有什么优先级的题目，分支限界显得就不那么乐观。**

**只能用fifo的队列，没有优先级，因此退化成回溯，但是还占用了空间，所以不及回溯。**





**（三）算法实施步骤和流程**（伪代码/流程图等方式描述）

**回溯解法：**

void QueenBack(int t,int n,int \*x,int &result) {

if (t == n) {

if (place(x, n)) {

result++;

for (int i = 1; i <= n; i++) cout << x[i] << " ";

cout << endl;

}

}

for (int i = t; i <= n; i++) {

swap(x, t, i);

if (place(x,t)) QueenBack(t + 1, n, x, result);

swap(x, i, t);//现场还原

}

}

分支限界法：

void pushqueue(int level,treenode \*prents,int chosen) {

treenode\* newnode = new treenode;

newnode->level = level;

newnode->chosen = new int[level + 1];

newnode->chosen[0] = 0;

newnode->chosen[level] = chosen;

for (int i = level - 1; i >= 1; i--) newnode->chosen[i] = prents->chosen[i];

bool flag = place(newnode, level);

if (flag) {//place(newnode, level)

q.push(\*newnode);

treenode\* b = newnode;

}

else delete newnode;

}

void nqueenBB(int n) {

int nowlevel = 0;

treenode head = { NULL,0 };

treenode\* nowpointer = &head;

int flag = 0;

while (nowlevel < n) {

for (int i = 1; i <= n; i++) pushqueue(nowlevel + 1, nowpointer, i);

flag = q.empty();

if (flag) return;

nowpointer = &q.front();

q.pop();

nowlevel = nowpointer->level;

}

for (int i = 1; i <= n; i++) {

outcome[i] = nowpointer->chosen[i];

}

return;

}

在这里也可以稍微的比较一下代码量，很明显，分支界限法的代码量还是偏大的。

**（四）源代码**（通过了编译运行的正确程序）

#include<iostream>

#include<queue>

using namespace std;

struct treenode {

int \*chosen;//选择的位置

int level;

};

queue<treenode> q;//这次直接把两种结构体定义成一种

int n = 4;

int outcome[10] = { 0 };//第一个位置空出来

bool place(treenode\* checknode, int nowlevel) {

for (int i = 1; i < nowlevel; i++) {

if (checknode->chosen[i] == checknode->chosen[nowlevel]) return false;

if (abs(checknode->level - i) == abs(checknode->chosen[i] - checknode->chosen[nowlevel])) return false;

}

return true;

}

void pushqueue(int level,treenode \*prents,int chosen) {//现有的位置，父亲节点，选择的位置

treenode\* newnode = new treenode;

newnode->level = level;

newnode->chosen = new int[level + 1];

newnode->chosen[0] = 0;

newnode->chosen[level] = chosen;

for (int i = level - 1; i >= 1; i--) {

newnode->chosen[i] = prents->chosen[i];

}

bool flag = place(newnode, level);

if (flag) {//place(newnode, level)

q.push(\*newnode);

treenode\* b = newnode;

}

else delete newnode;

}

void nqueenBB(int n) {

int nowlevel = 0;

treenode head = { NULL,0 };

treenode\* nowpointer = &head;

int flag = 0;

while (nowlevel < n) {

for (int i = 1; i <= n; i++) pushqueue(nowlevel + 1, nowpointer, i);

flag = q.empty();

if (flag) return;

nowpointer = &q.front();

q.pop();

nowlevel = nowpointer->level;

}

for (int i = 1; i <= n; i++) outcome[i] = nowpointer->chosen[i];

return;

}

int main(void) {

nqueenBB(n);

cout << "test example:" << n << endl;

if (outcome[1] == 0) cout << "no result" << endl;

else {

for (int i = 1; i <= n; i++) cout << outcome[i] << " ";

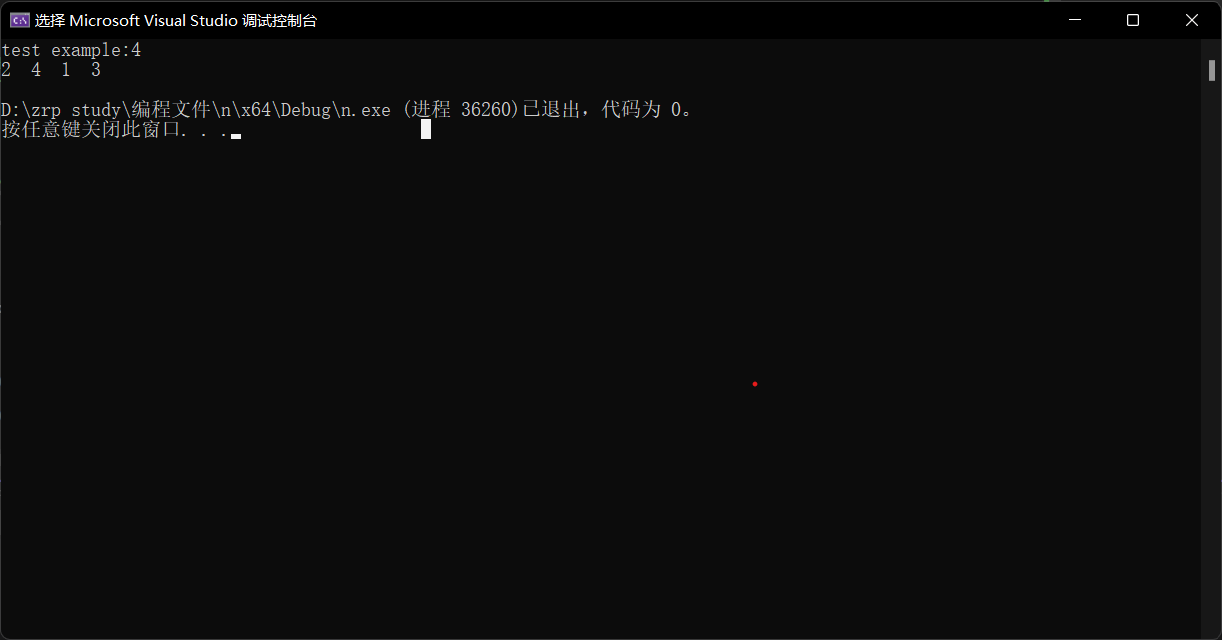
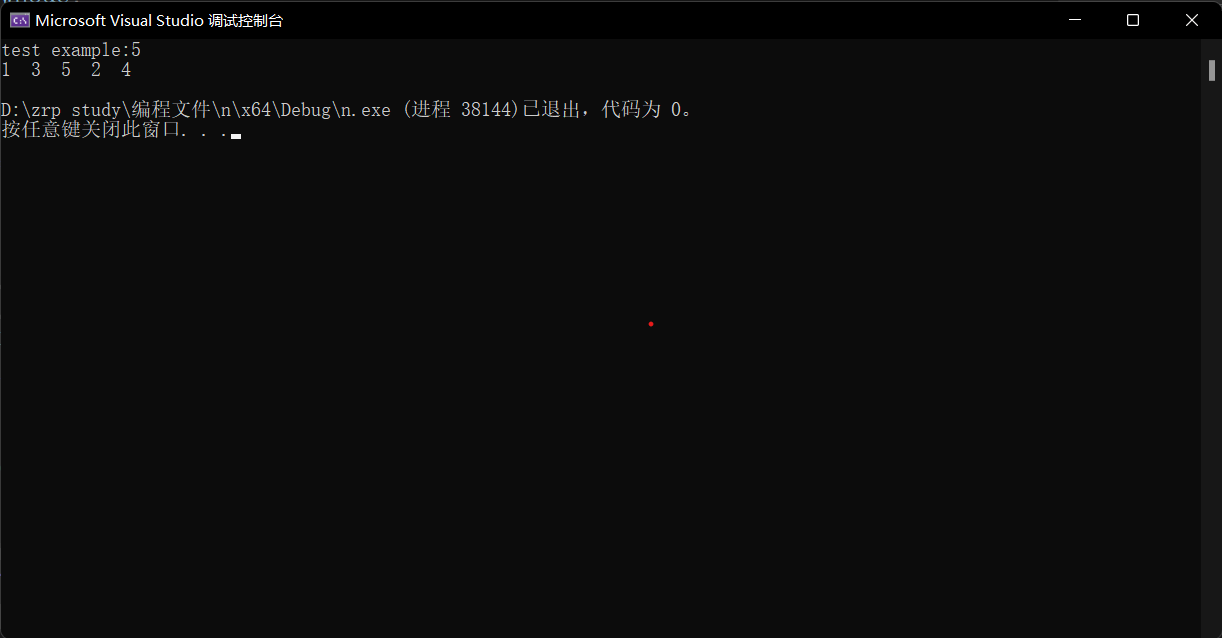
}

cout << endl;

return 0;

}

**（五）测试结果**（至少有两个以上算例及程序运行结果，截图贴进实验报告）



**（六）实验总结**（至少三句话，可以写复杂度分析、遇到问题、可能的改进措施、心得体会等）

1.分支限界法和回溯都是针对解空间树的解题方法，回溯是采用了深度优先的搜索方式，而分支限界法的话采用的是广度优先或者优先队列的方式。假设是要找一个最优解，回溯法是需要把所有叶子节点都遍历到，比较到，才能够得到答案。但是对于这种题目的话，分支限界法可以通过设计一个优先队列，优先让高质量的节点先结束遍历，更早的得到最优解，更早的结束算法。对于谁才是高质量的解，只能是一种预测，不能说确实这个最优的节点遍历下去一定能得到答案，所以设计优先队列的时候也没有必要设计过于复杂，做合理的限制，这样就已经比回溯算法时间上更优了。

2.分支限界法有一个问题，它的所有节点都是实实在在存在的，它需要真实的空间，原因是它广度优先遍历的话节点会非常的多，必须通过树的方式实实在在的连到一起，这样一来，没有什么信心能够比回溯算法效率高很多的话就不需要采取分支限界法了，因为它耗费的空间比回溯大得多。

3.n皇后问题根本体现不出来优先级，而且没到最后一步，也无法得到最终结果，反而，采取分支限界法，无端端增加了很多的内存，所以说事实上这种算法是十分低效的。

**暨南大学本科实验报告专用纸(附页)**