**《人工智能原理与方法》**

**实验报告**

**实验二：八数码问题的求解**

**（局部择优搜索）**

**专业班级：12级计算机二班**

**学号：1240069 姓名：李伟康**

**实验地点： 理工楼913**

**实验时间：2014年6月1日**

实验目的：

1．加深对局部择优搜索算法的理解，并掌握其在具体问题中的应用。

2．复习《数据结构》课程的相关知识，学会根据具体问题设计可解的数据结构。

3. 复习C语言的相关知识，提升自己的编程能力以及解决具体问题的能力。

实验语言环境：

1. VC++6.0（英文版）
2. 通过C语言实现

八数码问题的求解：

1. **局部择优算法说明**

(1) 初始结点S0放入OPEN中，计算f(S0)；

(2) 若OPEN为空，则搜索失败，问题无解；

(3) OPEN表中最顶端结点放到CLOSED表中，并给出顺序编号n；

(4) 若n为目标结点D，则搜索成功，问题有解；

(5) 若n无子结点，转(2)；

(6) 扩展n结点，用估价函数f(x)计算每个子节点的估价值，并按估价值从小到大的顺序依次放到OPEN表的首部，为每个子节点配置指向父节点的指针，转(2) 。

1. **数据结构设计与分析**
2. **算法实现所需数据结构分析**

局部择优搜索算法需要用到OPEN表和CLOSED表，结合《数据结构》的相关知识，此处利用栈来实现OPEN表和CLOSED表的构建。

1. **数据结构具体设计**

#define maxsize 100//栈长度

//数据类型转换及顺序表构建

typedef struct{

int value[3][3];

int index[3][3];

int F;//估价函数f(x)的值

int D;//d(x)为节点的深度

int H;//h(x)为表示节点x的格局与目标节点格局不相同的个数

}datatype;

//顺序栈定义

typedef struct{

datatype data[maxsize+1];//栈数组

int top; //栈顶指针

}Seqstack;

OPEN表和CLOSED表均用此栈进行构建。下面以OPEN表为例说明此顺序栈的具体含义：

顺序栈中的data是一个自定义的结构体类型，每一个值表示OPEN表的一个节点。结构体datatype用于存储每一个结点中的具体状态及估价函数的相关信息。估价函数为f(x)=d(x)+h(x)，其中d(x)为节点的深度，h(x)为表示节点x的格局与目标节点格局不相同的个数。d(x)通过全局变量depth进行计算（每次扩展时事depth自加1），h(x)通过函数caculate\_num\_of\_differences()计算。（利用双层for循环扫描当前状态和终止状态，遇到不同的位置，H自加1）

1. **算法实现设计与分析**
2. **局部择优算法实现分析**

解决方案的设计目的是模拟人脑使用算法的具体过程。因此，除了定义“静态”数据结构外，还需要定义恰当的“动态”函数。

1. **解决方案所用函数一览**

根据函数在解决方案中的重要性，将其划分为附属函数、基本函数、主要函数，每种函数种类及详细说明如下：

**A.附属函数构建**

a.void Show\_table(datatype value)//输出栈内元素

将当前扩展节点以适当格式输出。

b.bool select()//小菜单函数构建

程序执行结束后给用户的选择功能。用户输入Y，程序继续执行；用户输入N，程序终止执行。

**B.基本函数构建**

1. Seqstack \*InitStack()//初始化栈

据此函数构建OPEN表和CLOSED表，构建后的OPEN表和CLOSED表为空。

b.void Stackempty(Seqstack \*S)//判断栈是否为空

此函数主要用来检测OPEN表为空，即搜索失败的停止。若为空（不含初始化），则表示搜索失败，退出执行程序。

c.void Push(Seqstack \*S,datatype x)//入栈

此函数除了具有入栈（《数据结构》基础知识，此处不再赘述）的基本功能之外，还在此基础上增加了入栈前的判断功能（不是栈满的判断）。

**设计目的：**根据局部择优搜索算法中向OPEN表添加元素的规则知，进入OEPN表的元素为OPEN表之前和当前均不存在的，并且每次扩展的子节点应按估价函数值从大到小依次进栈，使得每次出栈时均得到估价函数值最小的节点。

**实现策略：**由于之前OEPN表中的元素已出栈（被当前元素覆盖），此方案中借助CLOSED表实现对之前OPEN表中的元素的遍历。此方案中的遍历采用双层for循环实现。利用冒泡法对新扩展的节点进行排序，然后依次压入OPEN表中。

d.void Pop(Seqstack \*S,datatype &x)//出栈

此函数主要用来弹出当前检测节点，其实现和基本出栈操作类似。

e.void Check\_input(int value)//检查输入是否正确

根据八数码问题的实际情况，节点状态的输入值只能从0-8中进行选择。若用户输入错误，程序终止执行。

f.void MpSort(datatype array[],int N)//冒泡排序

根据估价函数的值对当前节点扩展的多个子节点进行排序，以满足局部择优算法的要求。

g.void Input\_Eightcodes()//八数码问题的初值输入

此函数是为了让用户输入八数码问题求解前的基本要求。

**C.主要函数构建**

1. void Get\_way(datatype data)//找到结果后向上回溯至根节点

**设计目的：**根据找到的最终节点（搜索成功）“向上规约”找到该解的具体路径，并将其输出。

**实现策略：**从终止节点向上回滚（通过比较当前节点的父节点是否和上一节点的子节点相等实现），直至找到问题的初始节点，并将此路径以一种恰当的方式展示给用户。

b.void Caculate\_num\_of\_differences(datatype &data)//计算节点x的格 局与目标节点格局不相同的个数

函数内部通过双层for循环比较当前状态和终止状态中各个位置的值是否相等，如果不等，则使H自加1。

c.void extend(int temp)//节点的扩展

**设计目的：**根据当前状态进行扩展，得到符合条件的新的状态。

**实现策略：**利用试探法进行求解。每次扩展时使得行号或列号加1得到新的行号和列号（h\_n/l\_n），然后进行以下三个条件的判断：

h\_n/l\_n的值应在0-2之间，避免试探出界。

试探的位置的值应为0（即为空格），保证可移动。

试探的位置的值不应是该状态的前一个状态，避免回退。（为此，特意给每个状态增加了父节点状态，即index）

对满足以上条件的试探位置进行移值操作，将新扩展的节点存入data\_temp数组中，然后经过Mpsort()(冒泡排序)函数后将其压入OPEN表中。

d.bool Test()//检测搜索是否成功

**设计目的：**判断当前正在扩展的节点是否为该问题的终止节点。

**实现策略：**每次扩展前需弹出OPEN表的顶端节点(即为估价函数值最小)，将顶端节点与问题的终止节点进行比较。若相等，则搜索成功，输出解的路径，返回值为true;否则，返回值为false。

d.void Caculate\_Eightcodes()//八数码问题的具体求解

此函数主要用来调用以上设计的子函数，使其协调地运作起来，增加了程序的规整性和可读性。

1. **解决方案整体概述**

用户输入->用户数据的初始化处理->八数码问题的初始化->八数码问题的具体求解->最终结果的显示

Input\_Eightcodes()->Caculate\_Eightcodes();

1. **程序运行结果图示说明**
2. **用户输入界面**



1. **最终结果输出**



分析总结：

1. **主要的工作流程**
2. 用户数据的获取与初始化
3. 八数码问题的初始化
4. 八数码问题的具体求解
5. 最终结果的显示
6. **出现的主要问题与解决过程**

由于有了第一次解决汉诺塔问题的经验，此次处理八数码问题时遇到的问题并不是太多，也没有什么大的问题。整个项目所用时间虽然不少，但大都用到了对问题的验证及对根据验证的结果对程序进行优化。

1. **数据结构的设计**

第一次设计的数据结构并没有index(父节点的状态)，后来在进行解的回溯时发现了这一问题并进行了修改。

1. **扩展节点的生成**

**主要问题：**起初运用试探法时遗漏了“0=<h\_n|l\_n<=2”这一条件，通过后期的检验发现了这一问题并及时进行了修改。

1. **基础知识的再理解**

**结构体的理解**

自定义的结构体数据类型可以直接通过“=”进行两个数的赋值操作。（起初以为不能通过“=”进行赋值，仅仅通过循环赋值和函数操作进行，结果增加了程序的冗余度）

**引用类型形参的理解**

引用类型形参的函数在返回值为空的情况下可直接改变传入形参的值。（起初没有意识到这一点，致使最后的结果出现了错误，后来通过单步调试的方法发现了这个错误）

1. **不足**

该项目采用了局部择优搜索，由于算法本身的限制（有时找不到解）及栈的长度有限，对某些复杂的输入（和终止状态相差太多）往往得不到准确的解。

1. **反思**
2. 以后对具体问题进行求解时应该采用从整体到个体、一般到特殊的思想进行分析，避免犯错，少走弯路。(虽然自己通过边做边想的方法完成了此次实验，但花费时间较多，并且犯了许多错误。)
3. 对具体算法描述时一定要紧扣算法本身，从算法出发，切忌主观臆想。
4. 明确算法和程序的关系，编程只是为了描述相关算法。构建合适易行的算法才是解决问题的根本。
5. 加深对基础知识的理解，平时学习时应加强对基础知识的练习。
6. 注意单步调试方法的使用。
7. 应挑选特征值对设计的程序进行验证，检查程序的极限承受能力，提高程序本身的正确性和容错性。