人工智能实验三

1. 实验题目

用广度优先遍历求解汉诺塔的问题。

1. 实验分析
2. 汉诺塔问题：

n个盘子和3根柱子：A(源)、B(备用)、C(目的)，盘子的大小不同且中间有一孔，可以将盘子“串”在柱子上，每个盘子只能放在比它大的盘子上面。起初，所有盘子在A柱上，问题是将盘子一个一个地从A柱子移动到C柱子。移动过程中，可以使用B柱，但盘子也只能放在比它大的盘子上面。

1. 限制条件：

1.在小圆盘上不能放大圆盘。

2.在三根柱子之间一回只能移动一个圆盘。

3.只能移动在最顶端的圆盘。

3.广度优先搜索（breadth-first search）是解决自动寻路功能的算法之一。作为一种常见的图形搜索算法，它也被广泛应用于解决其他各类算法问题。一般情况下，对于一个节点，它的邻居节点的集合被称作open list，而在这个节点被遍历之前，其他所有已经遍历过的节点存于close list中。

1. 实验步骤

1、对于每一个状态(pegA, pegB, pegC)，由于要寻找其解，放入当前步骤（搜索层）问题表（队列堆栈皆可）。最开始初始化时表中只有一个元素，即最初状态。

2、维护一个搜索集set。对于状态(pegA, pegB, pegC)，如果已在搜索中，则该分支陷入重复，放弃该分支搜索；否则加入搜索集。

3、遍历当前问题表求解。如果有某一状态满足终止条件，那么当前步骤数即为所求，返回；如果没有，则展开得到更深层次的问题表，步骤数++。返回步骤1

四．实验内容

代码：#include <stdio.h>

#define width (rings+1)

int main()

{

int rings, last, next, x, z[500], s[3];

printf( "how many rings? ");

scanf( "%d ",&rings);

for(x=1; x <=rings; x++) /\* put rings on first peg \*/

z[x]=width-x;

for(x=0; x <=2\*width; x+=width) /\* set base for each peg \*/

z[x]=1000;

/\* if even number of rings, put first ring on second peg; if odd, on third \*/

if(rings%2==0)

{

last=1;

s[2]=0;

s[1]=1;

z[width+1]=z[rings];

}

else

{

last=2;

s[1]=0;

s[2]=1;

z[2\*width+1]=z[rings];

}

printf( "from 1 to %d\n ",last+1);

s[0]=rings-1;

while(s[0]+s[1]) /\* while first and second pegs aren 't empty \*/

{

/\* next ring to move is smaller of rings on the two pegs not moved onto last \*/

if(last==0)

next=z[width+s[1]] <z[2\*width+s[2]]?1:2;

if(last==1)

next=z[s[0]] <z[2\*width+s[2]]?0:2;

if(last==2)

next=z[s[0]] <z[width+s[1]]?0:1;

/\* top ring of 'to ' peg must be larger and an even 'distance ' away \*/

if((z[next\*width+s[next]]>z[last\*width+s[last]])|| ((z[last\*width+s[last]]-z[next\*width+s[next]])%2==0))

last=3-next-last;

printf( "from %d to %d\n ",next+1,last+1);

s[next]=s[next]-1;

s[last]=s[last]+1; /\* move from 'next ' to 'last ' peg \*/

z[last\*width+s[last]]=z[next\*width+s[next]+1];

}

}