**算法思想：迭代加深**

迭代加深基本思想：

深搜广搜都有自己的缺点。迭代加深(以下简称ID)则结合两者的优点，用深搜模拟广搜。

ID不过是由以下两个思想启发产生。为了避免过大的空间需求，必须使用深搜。为了避免搜索无用且过深的搜索树，必须按广搜的模式控制搜索深度。控制深度的深搜，就是ID。

深搜之前，我们设定一个限定的深度，要求深搜不得超过这个深度。

每一次深搜，我们检查当前所处的深度。如果达到限定的深度，立刻返回。

一次完整的深搜完成，如果没有找到答案，增加限定的深度，继续ID。

伪代码：

IterativeDeepening()

{

/\*max depth limited in DFS\*/

global maxdepth=1;

while (answer hadn't been found)

{

dfs(0);

maxdepth++;

}

return answer;

}

dfs(int depth)

{

/\*reach depth limit\*/

if (depth==maxdepth) return;

check whether answer is found;

/\*do dfs\*/

do dfs(depth+1);

}

实验代码：C语言

#include <stdio.h>

int maxdepth = 1;//最大寻找层

int tips = 0;//标记位

int answer;//位置

int i;

int depth=0 ;

int DFS(int maxdepth,int wonder,int a[10])

{

for(int c=0;c<=(depth+1)\*4;c++)

{

if(a[c]==wonder) answer=c;

}

if(depth = maxdepth)

if(tips = 1)

printf("找到答案") ;return answer;

}

int id(int limit,int wonder,int a[10])

{

DFS(maxdepth,wonder,a) ;

depth++;

while(maxdepth <= limit)//等答案没有找到的时候

{

maxdepth++;

}

return answer;

}

int main()

{int d;

int line[10]={

1,2,3,4,5,6,7,8,9,10

};//利用数组方式来存储将要比较的数据

printf("请输入你想要查找的数\n");

scanf("%d",&d);

id(5,d,line);

printf("这个数的位置在%d\n",answer);

return 0;

}

实验结果