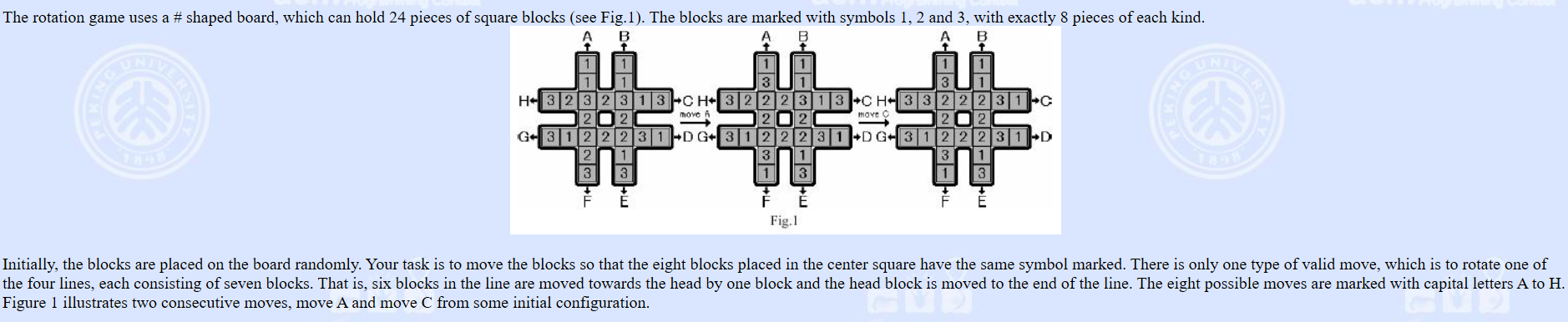
**AI第二次实验**

上次实验我写了一个dfs+剪枝，dfs还是比较简单的。想着将搜索进行到底，老师在课堂上讲到了**迭代加深搜索**(觉得这个好神奇)，这个算法既有dfs内存消耗较低的优点，又有bfs不会像dfs那样无限递归下去的风险，所以这节实验内容我选择了写一个迭代加深搜索。

我选择了poj上的2286题。

题目：



限制：



**题目大意**：

将1，2，3各自8个填入上面图中的24个格子里，然后你可以任意数量任意方向的拉动这8个子条，目标是将中间的8个格子的数字变成一样。让我们找出最少的操作次数。

**思路分析**：

这个时间限制和内存限制就很有意思了，**它给了15秒，但只给了20MB**。这个题目给人的感觉是可以直接dfs的，但是这个存在无限递归下去的风险，然后就想到了bfs，但是状态数实在太多了，bfs用到的栈直接内存爆炸了。最优解：dfs+bfs=IDA。由于是第一次写IDA这个思路是跟着别人的博客学的，还是很好理解的。

加上剪枝条件，这个题就ac了。思路很轻松就能理解，但是这道题的细节太多了（不是迭代加深时的细节），我debug了一节实验课才调试出来。

**ac代码：**

#include<iostream>

#include<algorithm>

#include<cstdio>

#include<algorithm>

#include<vector>

using namespace std;

int a[30];

vector<int> pp;//

//中间的位置是7，8，9，12，13，16，17，18

void print()

{

for(int i=1;i<=24;i++){

if(i==1||i==3||i==12||i==21||i==23)

cout<<" ";

if(i==2||i==4||i==13||i==22||i==24)

cout<<" ";

cout<<a[i];

if(i==2||i==4||i==11||i==13||i==20||i==22||i==24)

cout<<endl;

}

}

int po[8]={7,8,9,12,13,16,17,18};

int rev[8]={5,4,7,6,1,0,3,2};//不满足条件归位的时候需要的操作

int mov[8][8]={

{1,3,7,12,16,21,23},

{2,4,9,13,18,22,24},

{11,10,9,8,7,6,5},

{20,19,18,17,16,15,14},

{24,22,18,13,9,4,2},

{23,21,16,12,7,3,1},

{14,15,16,17,18,19,20},

{5,6,7,8,9,10,11}

};//ABCDEFGH

int h;

int num[4];

bool Judge()

{

for(int i=0;i<7;i++)

if(a[po[i]]!=a[po[i+1]])

return 0;

return 1;

}

int esdep()

{

num[1]=0;

num[2]=0;

num[3]=0;

for(int i=0;i<8;i++)

num[a[po[i]]]++;

int mmax=0;

for(int i=1;i<=3;i++)

mmax=max(mmax,num[i]);

return 8-mmax;

}

void move(int p)

{

int temp=a[mov[p][0]];

for(int j=0;j<6;j++)

a[mov[p][j]]=a[mov[p][j+1]];

a[mov[p][6]]=temp;

}

bool dfs(int b)

{

if(b>h)

return 0;

int temp;

for(int i=0;i<8;i++){

pp.push\_back(i);

move(i);//按照规则移动

int need=esdep();

if(need==0)

return 1;

if(need+b<=h){

if(dfs(b+1)==1)

return 1;

}

move(rev[i]);

pp.pop\_back();

}

return 0;

}

int main()

{

while(scanf("%d",&a[1])&&a[1]!=0){

pp.clear();

h=1;

for(int i=2;i<=24;i++)

scanf("%d",&a[i]);

h=esdep();//至少需要移动这么多次

if(Judge()){

printf("No moves needed\n");

printf("%d\n",a[7]);

continue;

}

while(dfs(1)==0){

h++;

}

for(int i=0;i<pp.size();i++)

printf("%c",char('A'+pp[i]));

printf("\n");

printf("%d\n",a[7]);

}

return 0;

}