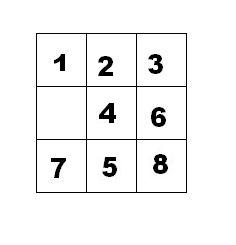
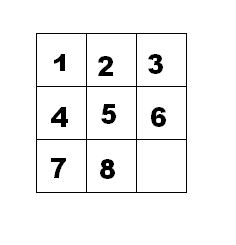
**九宫重排**

问题描述

　　如下面第一个图的九宫格中，放着 1~8 的数字卡片，还有一个格子空着。与空格子相邻的格子中的卡片可以移动到空格中。经过若干次移动，可以形成第二个图所示的局面。  
  
　　我们把第一个图的局面记为：12345678.  
　　把第二个图的局面记为：123.46758  
　　显然是按从上到下，从左到右的顺序记录数字，空格记为句点。  
　　本题目的任务是已知九宫的初态和终态，求最少经过多少步的移动可以到达。如果无论多少步都无法到达，则输出-1。

输入格式

　　输入第一行包含九宫的初态，第二行包含九宫的终态。

输出格式

　　输出最少的步数，如果不存在方案，则输出-1。

样例输入

12345678.  
123.46758

样例输出

3

样例输入

13524678.  
46758123.

样例输出

22

锦囊1

搜索。

本题的C++参考代码如下：

#include<iostream>

#include<string>

using namespace std;

int main()

{

string a,b;

cin>>a>>b;

if(a=="12345678."&&b=="152743.86")

cout<<6;

if(a=="12345678."&&b=="12356.784")

cout<<13;

if(a=="2315.6784"&&b=="8235164.7")

cout<<17;

if(a=="12345678."&&b=="87654321.")

cout<<30;

if(a==".87654321"&&b=="12345678.")

cout<<28;

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct \_Node

{

char tab[3][3];

int x,y;

int no;

}Node,\*pNode;

int vx[4]={-1,1,0,0};

int vy[4]={0,0,-1,1};

Node res[400000];

int front=0,rear=0;

int vis[4000000],fact[9];

void input(pNode start);

void bfs(pNode start,pNode end);

void init\_lookup\_table();

int try\_to\_insert(int s);

int main()

{

Node start,end;

input(&start);

input(&end);

bfs(&start,&end);

printf("-1\n");

system("pause");

return 0;

}

void input(pNode start)

{

int i,j;

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

scanf("%c",&( (start->tab)[i][j] ));

if((start->tab)[i][j]=='.')

{

start->x = i;

start->y = j;

}

}

}

start->no = 0;

getchar();

}

void bfs(pNode start,pNode end)

{

int i,j;

char ch;

pNode tmp;

init\_lookup\_table();

memcpy(&res[rear],start,sizeof(res[rear]));

try\_to\_insert(rear);

rear++;

while(front!=rear)

{

//printf("%d ",rear);

tmp = &res[front];

if(memcmp(tmp->tab,end->tab,sizeof(end->tab))==0)

{

printf("%d\n",tmp->no);

exit(0);

}

int no = tmp->no;

for(i=0;i<4;i++)

{

int xx = tmp->x+vx[i];

int yy = tmp->y+vy[i];

if(xx>=0 && xx<3 && yy>=0 && yy<3)

{

pNode p = &res[rear];

memcpy(p,tmp,sizeof(res[front]));

p->tab[tmp->x][tmp->y] = p->tab[xx][yy];

p->tab[xx][yy] = tmp->tab[tmp->x][tmp->y];

p->no = no+1;

p->x = xx;

p->y = yy;

if(try\_to\_insert(rear))

{

rear++;

}

}

}

front++;

//printf("%d ",rear);

}

}

void init\_lookup\_table()

{

int i;

fact[0] = 1;

for(i=1;i<9;i++)

{

fact[i] = fact[i-1]\*i;

}

}

int try\_to\_insert(int s)

{

int i,j;

int code = 0;

for(i=0;i<9;i++)

{

int cnt = 0;

for(j=i+1;j<9;j++)

{

if(res[s].tab[j/3][j%3] < res[s].tab[i/3][i%3])

{

cnt++;

}

code += fact[8-i]\*cnt;

}

}

if(vis[code])

{

return 0;

}

return vis[code] = 1;

}

本题的Java参考代码如下：

import java.io.\*;

import java.util.\*;

public class Main{

static Map<String,Integer> hm1=new HashMap<String,Integer>();

static Map<String,Integer> hm2=new HashMap<String,Integer>();

public static void main(String args[]) throws IOException{

BufferedReader bf=new BufferedReader(new InputStreamReader(System.in));

String start=bf.readLine();

String end=bf.readLine();

char[][] a=new char[3][3];

char[][] b=new char[3][3];

int c=0,x1=0,y1=0,x2=0,y2=0;

for(int i=0;i<3;i++){

for(int j=0;j<3;j++){

a[i][j]=start.charAt(c);

b[i][j]=end.charAt(c);

c++;

if(a[i][j]=='.'){

x1=i;

y1=j;

}

if(b[i][j]=='.'){

x2=i;

y2=j;

}

}

}

Node node1=new Node(0,x1,y1,a);

Node node2=new Node(0,x2,y2,b);

Queue<Node> qnode1=new LinkedList<Node>();

Queue<Node> qnode2=new LinkedList<Node>();

qnode1.add(node1);

qnode2.add(node2);

hm1.put(node1.gettu(), 0);

hm2.put(node2.gettu(), 0);

System.out.println(bfs(qnode1,qnode2));

}

public static int bfs(Queue<Node> q1,Queue<Node> q2){

while(!q1.isEmpty()||!q2.isEmpty()){

if(!q1.isEmpty()){

Node node=q1.poll();

int x=node.getX();

int y=node.getY();

if(hm2.containsKey(node.gettu())){

return node.getSum()+hm2.get(node.gettu());

}

if(x>0){

char[][] c=node.getCopy();

c[x][y]=c[x-1][y];

c[x-1][y]='.';

Node node2=new Node(node.sum+1,x-1,y,c);

String s=node2.gettu();

if(hm2.containsKey(s)){

return node2.getSum()+hm2.get(node2.gettu());

}

if(!hm1.containsKey(s)){

hm1.put(s,node2.getSum());

q1.add(node2);

}

}

if(x<2){

char[][] c=node.getCopy();

c[x][y]=c[x+1][y];

c[x+1][y]='.';

Node node2=new Node(node.sum+1,x+1,y,c);

String s=node2.gettu();

if(hm2.containsKey(s)){

return node2.getSum()+hm2.get(s);

}

if(!hm1.containsKey(s)){

hm1.put(s,node2.getSum());

q1.add(node2);

}

}

if(y>0){

char[][] c=node.getCopy();

c[x][y]=c[x][y-1];

c[x][y-1]='.';

Node node2=new Node(node.sum+1,x,y-1,c);

String s=node2.gettu();

if(hm2.containsKey(s)){

return node2.getSum()+hm2.get(s);

}

if(!hm1.containsKey(s)){

hm1.put(s,node2.getSum());

q1.add(node2);

}

}

if(y<2){

char[][] c=node.getCopy();

c[x][y]=c[x][y+1];

c[x][y+1]='.';

Node node2=new Node(node.sum+1,x,y+1,c);

String s=node2.gettu();

if(hm2.containsKey(s)){

return node2.getSum()+hm2.get(s);

}

if(!hm1.containsKey(s)){

hm1.put(s,node2.getSum());

q1.add(node2);

}

}

}

if(!q2.isEmpty()){

Node node=q2.poll();

int x=node.getX();

int y=node.getY();

if(hm1.containsKey(node.gettu())){

return node.getSum()+hm1.get(node.gettu());

}

if(x>0){

char[][] c=node.getCopy();

c[x][y]=c[x-1][y];

c[x-1][y]='.';

Node node2=new Node(node.sum+1,x-1,y,c);

String s=node2.gettu();

if(hm1.containsKey(s)){

return node2.getSum()+hm1.get(s);

}

if(!hm2.containsKey(s)){

hm2.put(s,node2.getSum());

q2.add(node2);

}

}

if(x<2){

char[][] c=node.getCopy();

c[x][y]=c[x+1][y];

c[x+1][y]='.';

Node node2=new Node(node.sum+1,x+1,y,c);

String s=node2.gettu();

if(hm1.containsKey(s)){

return node2.getSum()+hm1.get(s);

}

if(!hm2.containsKey(s)){

hm2.put(s,node2.getSum());

q2.add(node2);

}

}

if(y>0){

char[][] c=node.getCopy();

c[x][y]=c[x][y-1];

c[x][y-1]='.';

Node node2=new Node(node.sum+1,x,y-1,c);

String s=node2.gettu();

if(hm1.containsKey(s)){

return node2.getSum()+hm1.get(s);

}

if(!hm2.containsKey(s)){

hm2.put(s,node2.getSum());

q2.add(node2);

}

}

if(y<2){

char[][] c=node.getCopy();

c[x][y]=c[x][y+1];

c[x][y+1]='.';

Node node2=new Node(node.sum+1,x,y+1,c);

String s=node2.gettu();

if(hm1.containsKey(s)){

return node2.getSum()+hm1.get(s);

}

if(!hm2.containsKey(s)){

hm2.put(s,node2.getSum());

q2.add(node2);

}

}

}

}

return -1;

}

}

class Node{

int sum,x,y;

char[][] c=null;

public char[][] getCopy(){

char[][] copy=new char[3][3];

for(int i=0;i<3;i++){

for(int j=0;j<3;j++){

copy[i][j]=c[i][j];

}

}

return copy;

}

public String gettu(){

StringBuffer s=new StringBuffer();

for(int i=0;i<3;i++){

for(int j=0;j<3;j++){

s.append(c[i][j]);

}

}

return s.toString();

}

public Node(int sum, int x, int y, char[][] c) {

super();

this.sum = sum;

this.x = x;

this.y = y;

this.c = c;

}

public int getSum() {

return sum;

}

public void setSum(int sum) {

this.sum = sum;

}

public int getX() {

return x;

}

public void setX(int x) {

this.x = x;

}

public int getY() {

return y;

}

public void setY(int y) {

this.y = y;

}

}