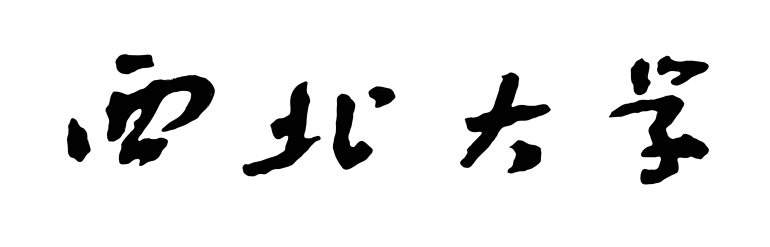
****

****

**人工智能实践课实验报告**

**题 目： 　状态空间搜索算法**

**年 级： 2019**

**学 号： 2019111137**

**专 业： 　软件工程**

**姓 名： 杨润林**

1. **实验内容**

**用深度、宽度、最佳优先搜索算法，三种算法来解八数码问题和一字棋问题。**

1. **解决思路**
   1. **八数码** 
      1. **定义 board类来表示每一个状态**
      2. **只有当输入状态和目标状态的排列逆序数奇偶性相同时，才有解**
      3. **生成子节点的函数move()**
         1. **分为三种情况，0处于四个顶点、四个中点、中心**
         2. **顶点只能向两个方向移动**
         3. **中点只能向三个方向移动**
         4. **中心可以向四个方向移动**
         5. **编写四个函数up,down,left,right,返回向该方向移动后的状态**
   2. **一字棋**
      1. **假设X先手**
      2. **给出的总是X的胜利路径**
      3. **启发函数h(n)=E(X)-E(O),求可能的胜利路线条数差值**
      4. **E(X)是通过检测八条胜利路线上是否有对方棋子来给出胜利路线条数的**
      5. **生成子节点的函数move()**
         1. **在chess类中定义了一个属性turn，保存下一步要填的值**
         2. **循环所有为空的位置，将turn值填入，将该状态加入res集合中**
         3. **return res**
2. **实验结果**

**分别用三总算法，较好地解决了八数码问题和一字棋问题，能给出输入状态到目标状态的路径，速度较快；假如问题无解，能直接给出判断，不会进入死循环。**

1. **总结**
2. **在编码过程中遇到了要对ArrayList排序的问题，调用了sort方法，传入ArrayList对象和一个匿名的重写了compare方法的Comparator对象，实现了按照某个属性排序的功能。**
3. **编码过程中，每编完一个函数或者部分，建议先对其进行测试，不要写完一大段代码再进行测试，此时debug难度和花费的时间比较多。**

**附录（代码）：**

1. **puzzle8.java**

**package project\_java1;**

**import java.util.ArrayList;**

**import java.util.Collections;**

**import java.util.Comparator;**

**import java.util.Scanner;**

**import java.util.concurrent.TimeUnit;**

**class board{**

**public int h;**

**public int f;**

**public board pre=null;**

**public int[][] dot=new int[3][3];**

**public int[] pos=new int[2];**

**public static int depth0(board b) {**

**if(b==null)**

**return 0;**

**else**

**return depth0(b.pre)+1;**

**}**

**public int depth() {**

**return depth0(this);**

**}**

**public board(int[][] dot) {**

**this.dot=dot;**

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

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

**{**

**if(dot[i][j]==0){**

**this.pos[0]=i;**

**this.pos[1]=j;**

**}**

**}}**

**}**

**//判断两个节点是否相等**

**public Boolean equal(board a) {**

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

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

**if(dot[i][j]!=a.dot[i][j])**

**return false;**

**}**

**return true;**

**}**

**public static board up(int[][] dot,int[] pos) {**

**int[][] res=new int[3][3];**

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

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

**res[i][j]=dot[i][j];**

**int temp;**

**temp=res[pos[0]][pos[1]];**

**res[pos[0]][pos[1]]=res[pos[0]-1][pos[1]];**

**res[pos[0]-1][pos[1]]=temp;**

**return new board(res);**

**}**

**public static board down(int[][] dot,int[] pos) {**

**int[][] res=new int[3][3];**

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

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

**res[i][j]=dot[i][j];**

**int temp;**

**temp=res[pos[0]][pos[1]];**

**res[pos[0]][pos[1]]=res[pos[0]+1][pos[1]];**

**res[pos[0]+1][pos[1]]=temp;**

**return new board(res);**

**}**

**public static board left(int[][] dot,int[] pos) {**

**int[][] res=new int[3][3];**

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

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

**res[i][j]=dot[i][j];**

**int temp;**

**temp=res[pos[0]][pos[1]];**

**res[pos[0]][pos[1]]=res[pos[0]][pos[1]-1];**

**res[pos[0]][pos[1]-1]=temp;**

**return new board(res);**

**}**

**public static board right(int[][] dot,int[] pos) {**

**int[][] res=new int[3][3];**

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

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

**res[i][j]=dot[i][j];**

**int temp;**

**temp=res[pos[0]][pos[1]];**

**res[pos[0]][pos[1]]=res[pos[0]][pos[1]+1];**

**res[pos[0]][pos[1]+1]=temp;**

**return new board(res);**

**}**

**//返回所有子节点**

**public ArrayList<board> move() {**

**ArrayList<board> res=new ArrayList<>();**

**//四个顶点**

**if(pos[0]==0 && pos[1]==0)**

**{**

**res.add(right(dot,pos));**

**res.add(down(dot,pos));**

**}**

**if(pos[0]==0 && pos[1]==2)**

**{**

**res.add(left(dot,pos));**

**res.add(down(dot,pos));**

**}**

**if(pos[0]==2 && pos[1]==0)**

**{**

**res.add(right(dot,pos));**

**res.add(up(dot,pos));**

**}**

**if(pos[0]==2 && pos[1]==2)**

**{**

**res.add(left(dot,pos));**

**res.add(up(dot,pos));**

**}**

**//四个中点**

**if(pos[0]==0 && pos[1]==1)**

**{**

**res.add(right(dot,pos));**

**res.add(left(dot,pos));**

**res.add(down(dot,pos));**

**}**

**if(pos[0]==1 && pos[1]==0)**

**{**

**res.add(up(dot,pos));**

**res.add(down(dot,pos));**

**res.add(right(dot,pos));**

**}**

**if(pos[0]==2 && pos[1]==1)**

**{**

**res.add(right(dot,pos));**

**res.add(left(dot,pos));**

**res.add(up(dot,pos));**

**}**

**if(pos[0]==1 && pos[1]==2)**

**{**

**res.add(up(dot,pos));**

**res.add(down(dot,pos));**

**res.add(left(dot,pos));**

**}**

**//一个中心**

**if(pos[0]==1 && pos[1]==1)**

**{**

**res.add(right(dot,pos));**

**res.add(down(dot,pos));**

**res.add(left(dot,pos));**

**res.add(up(dot,pos));**

**}**

**return res;**

**}**

**}**

**public class puzzle8 {**

**public static int inversion(board b) {**

**int[] arr=new int[9];**

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

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

**arr[i\*3+j]=b.dot[i][j];**

**int res=0;**

**for(int i=0;i<9;i++) {**

**if(arr[i]==0)**

**continue;**

**for(int j=0;j<i;j++) {**

**if(arr[j]>arr[i])**

**res++;**

**}**

**}**

**return res%2;**

**}**

**public static Boolean contain(ArrayList<board> a,board b) {**

**for (board c : a) {**

**if(b.equal(c))**

**return true;**

**}**

**return false;**

**}**

**public static int[][] b={{1,2,3},{8,0,4},{7,6,5}};**

**public static board target= new board(b);**

**//输出棋盘状态**

**public static void print(board b) {**

**int[][] a=b.dot;**

**System.out.println("--------------");**

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

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

**{**

**System.out.print(a[i][j]+" ");**

**}**

**System.out.println();**

**}**

**System.out.println("--------------");**

**}**

**//启发函数h(n)，求错牌个数**

**public static int h(board dot) {**

**int ans=0;**

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

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

**{**

**if(dot.dot[i][j]!=target.dot[i][j])**

**ans+=1;**

**}**

**return ans;**

**}**

**public static ArrayList<board> bfs(board now) throws InterruptedException {**

**ArrayList<board> open=new ArrayList<board>();**

**ArrayList<board> closed=new ArrayList<board>();**

**int times=0;**

**open.add(now);**

**while(open.size()!=0) {**

**times++;**

**if(times%10000==0)**

**{System.out.println("已经尝试了"+(times)+"次");**

**print(open.get(0));}**

**// TimeUnit.SECONDS.sleep(1);**

**// System.out.println("open.size="+open.size());**

**// System.out.println("closed.size="+closed.size());**

**// System.out.println("!!!!!!!open");**

**board X=open.get(0);**

**open.remove(0);**

**if(X.equal(target)) {**

**ArrayList<board> res=new ArrayList<board>();**

**res.add(X);**

**while(X.pre!=null)**

**{res.add(0,X.pre);**

**X=X.pre;**

**}**

**return res;**

**}**

**else {**

**ArrayList<board> children=X.move();**

**closed.add(X);**

**ArrayList<board> remove\_boards=new ArrayList<>();**

**for (board a : children) {**

**if(contain(open,a)||contain(closed,a)) {**

**remove\_boards.add(a);**

**}**

**else**

**a.pre=X;**

**}**

**children.removeAll(remove\_boards);**

**open.addAll(children);**

**}**

**}**

**return (new ArrayList<board>());**

**}**

**public static ArrayList<board> dfs(board now) throws InterruptedException {**

**ArrayList<board> sl=new ArrayList<board>();**

**ArrayList<board> nsl=new ArrayList<board>();**

**ArrayList<board> de=new ArrayList<board>();**

**board cs=now;**

**sl.add(now);**

**nsl.add(now);**

**int times=0;**

**while(nsl.size()!=0) {**

**times++;**

**System.out.println("已经尝试了"+(times)+"次");**

**print(cs);**

**if(cs.equal(target))**

**return sl;**

**ArrayList<board> children=cs.move();**

**ArrayList<board> remove\_boards=new ArrayList<board>();**

**for (board b : children) {**

**if(contain(de,b)||contain(sl,b)||contain(nsl,b)) {**

**remove\_boards.add(b);**

**}**

**}**

**children.removeAll(remove\_boards);**

**if(children.size()==0) {**

**while(sl.size()!=0 && cs.equal(sl.get(0))) {**

**for (board t : de) {**

**print(t);**

**}**

**de.add(cs);**

**sl.remove(0);**

**nsl.remove(0);**

**cs=nsl.get(nsl.size()-1);**

**}**

**sl.add(cs);**

**}**

**else {**

**nsl.addAll(children);**

**cs=nsl.get(nsl.size()-1);**

**sl.add(cs);**

**}**

**}**

**return new ArrayList<board>();**

**}**

**@SuppressWarnings("unchecked")**

**public static ArrayList<board> best\_first\_search(board now) throws InterruptedException {**

**ArrayList<board> open=new ArrayList<board>();**

**ArrayList<board> closed=new ArrayList<board>();**

**open.add(now);**

**int times=0;**

**while(open.size()!=0) {**

**times++;**

**if(times%10000==0)**

**{System.out.println("已经尝试了"+(times)+"次");**

**print(open.get(0));}**

**board X=open.get(0);**

**open.remove(0);**

**if(X.equal(target)) {**

**ArrayList<board> res=new ArrayList<board>();**

**res.add(X);**

**while(X.pre!=null)**

**{res.add(0,X.pre);**

**X=X.pre;**

**}**

**return res;**

**}**

**else {**

**ArrayList<board> children=X.move();**

**closed.add(X);**

**ArrayList<board> remove\_boards=new ArrayList<>();**

**for (board a : children) {**

**if(contain(open,a)||contain(closed,a)) {**

**remove\_boards.add(a);**

**}**

**else**

**{a.pre=X;**

**a.h=h(a);**

**a.f=a.depth();**

**}**

**}**

**children.removeAll(remove\_boards);**

**open.addAll(children);**

**//对children列表，按启发值排序,启发值越小越好**

**Collections.sort(open, new Comparator() {**

**@Override**

**public int compare(Object o1, Object o2) {**

**board b1=(board)o1;**

**board b2=(board)o2;**

**return (b1.f+b1.h)-(b2.f+b2.h);**

**}**

**});**

**// System.out.println("open排序结果");**

**// for(board a:open) {**

**// System.out.println("启发函数:"+(a.f+a.h));**

**// }**

**// System.out.println("排序结束");**

**}**

**}**

**return (new ArrayList<board>());**

**}**

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

**System.out.println("请以排列的形式，输入初始状态:");**

**Scanner input=new Scanner(System.in);**

**int[][] a=new int[3][3];**

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

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

**a[i][j]=input.nextInt();**

**input.close();**

**board in=new board(a);**

**if(inversion(in)!=inversion(target)) {**

**System.out.println("input逆序数="+inversion(in));**

**System.out.println("target逆序数="+inversion(target));**

**System.out.println("当前状态无法到达目标状态！");**

**return ;**

**}**

**ArrayList<board> res=dfs(in);**

**//ArrayList<board> res=bfs(in);**

**//ArrayList<board> res=best\_first\_search(in);**

**System.out.println("一共需要"+res.size()+"步移动");**

**if(res.size()==0)**

**System.out.println("当前状态无解");**

**else**

**{**

**for (board b : res)**

**{**

**print(b);**

**}**

**}**

**}**

**}**

1. **wordChess.java**

**package project\_java1;**

**import java.util.ArrayList;**

**import java.util.Collections;**

**import java.util.Comparator;**

**import java.util.Scanner;**

**import java.util.concurrent.TimeUnit;**

**class chess{**

**public int h;**

**public int f;**

**public chess pre=null;**

**public int[][] dot=new int[3][3];**

**public int turn;**

**public static int depth0(chess b) {**

**if(b==null)**

**return 0;**

**else**

**return depth0(b.pre)+1;**

**}**

**public int depth() {**

**return depth0(this);**

**}**

**public chess(int[][] dot) {**

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

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

**this.dot[i][j]=dot[i][j];**

**int n0=0,n1=1;**

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

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

**{**

**if(dot[i][j]==0)**

**n0++;**

**else if(dot[i][j]==1)**

**n1++;**

**}**

**}**

**if(n0==n1 || n0>n1)**

**turn=1;**

**else**

**turn=0;**

**}**

**//判断两个节点是否相等**

**public Boolean equal(chess a) {**

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

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

**if(dot[i][j]!=a.dot[i][j])**

**return false;**

**}**

**return true;**

**}**

**//返回所有子节点**

**public ArrayList<chess> move(){**

**ArrayList<chess> res=new ArrayList<chess>();**

**int[][] temp=new int[3][3];**

**//temp=dot**

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

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

**temp[i][j]=dot[i][j];**

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

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

**if(dot[i][j]==-1){**

**temp[i][j]=turn;**

**res.add(new chess(temp));**

**temp[i][j]=dot[i][j];**

**}**

**}**

**return res;**

**}**

**}**

**public class wordChess {**

**public static int row(chess c) {**

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

**if(c.dot[i][0]==c.dot[i][1]&&c.dot[i][0]==c.dot[i][2])**

**return c.dot[i][0];**

**}**

**return -1;**

**}**

**public static int col(chess c) {**

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

**if(c.dot[0][i]==c.dot[1][i]&&c.dot[0][i]==c.dot[2][i])**

**return c.dot[0][i];**

**}**

**return -1;**

**}**

**public static int diag(chess c) {**

**if(c.dot[0][0]==c.dot[1][1]&&c.dot[1][1]==c.dot[2][2])**

**return c.dot[0][0];**

**else if(c.dot[0][2]==c.dot[1][1]&&c.dot[1][1]==c.dot[2][0])**

**return c.dot[0][2];**

**return -1;**

**}**

**//判断游戏结果**

**public static int judge(chess c) {**

**int[] res= {row(c),col(c),diag(c)};**

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

**if(res[i]!=-1)**

**return res[i];**

**return -1;**

**}**

**public static Boolean contain(ArrayList<chess> a,chess b) {**

**for (chess c : a) {**

**if(b.equal(c))**

**return true;**

**}**

**return false;**

**}**

**//输出棋盘状态**

**public static void print(chess b) {**

**int[][] a=b.dot;**

**System.out.println("--------------");**

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

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

**{**

**System.out.printf("%2d ",a[i][j]);**

**}**

**System.out.println();**

**}**

**System.out.println("--------------");**

**}**

**public static int E(chess c,int k) {**

**int res=8;**

**int temp;**

**if(k==0)**

**temp=1;**

**else**

**temp=0;**

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

**if(temp==c.dot[i][1]||temp==c.dot[i][0]||temp==c.dot[i][2])**

**res--;**

**if(temp==c.dot[1][i]||temp==c.dot[0][i]||temp==c.dot[2][i])**

**res--;**

**}**

**if(temp==c.dot[0][0]||temp==c.dot[2][2])**

**res--;**

**if(temp==c.dot[2][0]||temp==c.dot[0][2])**

**res--;**

**return res;**

**}**

**//启发函数h(n)=E(X)-E(O),求可能的胜利路线条数差值**

**public static int h(chess dot) {**

**return E(dot,1)-E(dot,0);**

**}**

**public static ArrayList<chess> bfs(chess now) throws InterruptedException {**

**ArrayList<chess> open=new ArrayList<chess>();**

**ArrayList<chess> closed=new ArrayList<chess>();**

**int times=0;**

**open.add(now);**

**while(open.size()!=0) {**

**times++;**

**if(times%10==0)**

**{System.out.println("已经尝试了"+(times)+"次");**

**print(open.get(0));}**

**chess X=open.get(0);**

**open.remove(0);**

**int j=judge(X);**

**if(j==1) {**

**ArrayList<chess> res=new ArrayList<chess>();**

**res.add(X);**

**while(X.pre!=null)**

**{res.add(0,X.pre);**

**X=X.pre;**

**}**

**System.out.println("X胜利");**

**return res;**

**}**

**else {**

**ArrayList<chess> children=X.move();**

**closed.add(X);**

**ArrayList<chess> remove\_chesss=new ArrayList<>();**

**for (chess a : children) {**

**if(contain(open,a)||contain(closed,a)) {**

**remove\_chesss.add(a);**

**}**

**else**

**a.pre=X;**

**}**

**children.removeAll(remove\_chesss);**

**open.addAll(children);**

**}**

**}**

**return (new ArrayList<chess>());**

**}**

**public static ArrayList<chess> dfs(chess now) throws InterruptedException {**

**ArrayList<chess> sl=new ArrayList<chess>();**

**ArrayList<chess> nsl=new ArrayList<chess>();**

**ArrayList<chess> de=new ArrayList<chess>();**

**chess cs=now;**

**sl.add(now);**

**nsl.add(now);**

**while(nsl.size()!=0) {**

**if(judge(cs)==1)**

**return sl;**

**ArrayList<chess> children=cs.move();**

**ArrayList<chess> remove\_chesss=new ArrayList<chess>();**

**for (chess b : children) {**

**if(contain(de,b)||contain(sl,b)||contain(nsl,b)) {**

**remove\_chesss.add(b);**

**}**

**}**

**children.removeAll(remove\_chesss);**

**if(children.size()==0) {**

**while(sl.size()!=0 && cs.equal(sl.get(0))) {**

**for (chess t : de) {**

**print(t);**

**}**

**de.add(cs);**

**sl.remove(0);**

**nsl.remove(0);**

**cs=nsl.get(nsl.size()-1);**

**}**

**sl.add(cs);**

**}**

**else {**

**nsl.addAll(children);**

**cs=nsl.get(nsl.size()-1);**

**sl.add(cs);**

**}**

**}**

**return new ArrayList<chess>();**

**}**

**@SuppressWarnings("unchecked")**

**public static ArrayList<chess> best\_first\_search(chess now) throws InterruptedException {**

**ArrayList<chess> open=new ArrayList<chess>();**

**ArrayList<chess> closed=new ArrayList<chess>();**

**open.add(now);**

**int times=0;**

**while(open.size()!=0) {**

**times++;**

**if(times%10000==0)**

**{System.out.println("已经尝试了"+(times)+"次");**

**print(open.get(0));}**

**chess X=open.get(0);**

**open.remove(0);**

**if(judge(X)==1) {**

**ArrayList<chess> res=new ArrayList<chess>();**

**res.add(X);**

**while(X.pre!=null)**

**{res.add(0,X.pre);**

**X=X.pre;**

**}**

**return res;**

**}**

**else {**

**ArrayList<chess> children=X.move();**

**closed.add(X);**

**ArrayList<chess> remove\_chesss=new ArrayList<>();**

**for (chess a : children) {**

**if(contain(open,a)||contain(closed,a)) {**

**remove\_chesss.add(a);**

**}**

**else**

**{a.pre=X;**

**a.h=h(a);**

**a.f=a.depth();**

**}**

**}**

**children.removeAll(remove\_chesss);**

**open.addAll(children);**

**//对children列表，按启发值排序,启发值越小越好**

**Collections.sort(open, new Comparator() {**

**@Override**

**public int compare(Object o1, Object o2) {**

**chess b1=(chess)o1;**

**chess b2=(chess)o2;**

**return (b1.f+b1.h)-(b2.f+b2.h);**

**}**

**});**

**// System.out.println("open排序结果");**

**// for(chess a:open) {**

**// System.out.println("启发函数:"+(a.f+a.h));**

**// }**

**// System.out.println("排序结束");**

**}**

**}**

**return (new ArrayList<chess>());**

**}**

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

**System.out.println("请以排列的形式，输入初始状态:");**

**System.out.println("-1:空 , 0:O , 1:X");**

**Scanner input=new Scanner(System.in);**

**int[][] a=new int[3][3];**

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

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

**a[i][j]=input.nextInt();**

**input.close();**

**chess in=new chess(a);**

**print(in);**

**//ArrayList<chess> res=dfs(in);**

**//ArrayList<chess> res=bfs(in);**

**ArrayList<chess> res=best\_first\_search(in);**

**System.out.println("一共需要"+(res.size()-1)+"步移动");**

**if(res.size()==0)**

**System.out.println("当前状态无解");**

**else**

**{**

**for (chess b : res)**

**{**

**print(b);**

**}**

**}**

**}**

**}**