**A\*算法程序**

**public class Eight{**

**int g;**

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

**int zi,zj; //0的位置**

**Eight former;**

**public Eight()**

**{**

**g=0;**

**zi=-1;**

**zj=-1;**

**former=null;**

**}**

**public Eight(Eight other){**

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

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

**e[i][j] = other.e[i][j];**

**}**

**zi=other.zi;**

**zj=other.zj;**

**former=other.former;**

**}**

**public void setFormer(Eight e){**

**this.former=e;**

**}**

**public void listAll( Eight e ){**

**System.out.println("最优路径为：");**

**List l=new List();**

**l.insertAtFront(e);**

**while( e.former != null ){**

**l.insertAtFront(e.former);**

**e = new Eight(e.former);**

**}**

**while(!l.isEmpty()){**

**e=l.getFirstNode();**

**e.print();**

**l.removeFromFront();**

**}**

**return ;**

**}**

**public boolean equals(Eight a)**

**{**

**int i=0;**

**int j=0;**

**if(a==null)**

**return false;**

**else {**

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

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

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

**return false;**

**}**

**return true;**

**}**

**}**

**public void Swap(int i,int j,int m,int n){**

**int temp;**

**temp = this.e[i][j];**

**this.e[i][j] = this.e[m][n];**

**this.e[m][n] = temp;**

**}**

**public int h(){**

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

**int h =0,i,j;**

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

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

**if(this.e[i][j]!=dest[i][j] && e[i][j]!=0)**

**h++;**

**}**

**return h;**

**}**

**public int f(){**

**return g+h();**

**}**

**public Eight[] ex(){**

**List e =new List();**

**int i=0,j=0,k=0;**

**int m,n;**

**boolean flag = true;**

**for(i=0;i<3&&flag;i++)**

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

**if(this.e[i][j]==0){**

**flag=false;**

**break;**

**}**

**}**

**i=i-1;**

**if(i-1>=0){**

**Eight a=new Eight(this);**

**m=i-1;**

**a.Swap(m,j,i,j);**

**e.insertAtBack(a);**

**++k;**

**}**

**if(i+1<3){**

**Eight a=new Eight(this);**

**m=i+1;**

**a.Swap(m,j,i,j);**

**e.insertAtBack(a);**

**++k;**

**}**

**if(j-1>=0){**

**Eight a=new Eight(this);**

**n=j-1;**

**a.Swap(i,n,i,j);**

**e.insertAtBack(a);**

**++k;**

**}**

**if(j+1<3){**

**Eight a=new Eight(this);**

**n=j+1;**

**a.Swap(i,n,i,j);**

**e.insertAtBack(a);**

**++k;**

**}**

**Eight b[]=new Eight[k];**

**for(int x=0;x<b.length;x++){**

**b[x]=e.getFirstNode();**

**b[x].setFormer(this);**

**b[x].g=this.g+1;**

**e.removeFromFront();**

**}**

**return b;**

**}**

**static void sort(Eight a[]){**

**Eight temp;**

**for(int i=0;i<a.length;i++){**

**for(int j=a.length-1;j>i;j--){**

**if(a[j].f()<a[j-1].f()){**

**temp=a[j-1];**

**a[j-1]=a[j];**

**a[j]=temp;**

**}**

**}**

**}**

**}**

**static void listSort(List l){**

**Eight a[]=new Eight[l.length];**

**for(int i=0;i<a.length;i++){**

**a[i]=l.getFirstNode();**

**l.removeFromFront();**

**}**

**sort(a);**

**for (int i=0;i<a.length;i++)**

**{**

**l.insertAtBack(a[i]);**

**}**

**}**

**public boolean hasIt(List l){**

**ListNode s=l.firstNode;**

**boolean b=false;**

**while(s!=null)**

**{**

**if(this.equals(s.data)){**

**b=true;**

**break;**

**}**

**s=s.next;**

**}**

**return b;**

**}**

**public void print()**

**{**

**if(this!=null){**

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

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

**System.out.print(e[i][j]);**

**}**

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

**}**

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

**}**

**else**

**return;**

**}**

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

**Eight e=new Eight();**

**List open =new List();**

**List closed =new List();**

**open.insertAtBack(e);**

**while(true){**

**if(open.isEmpty())**

**break;**

**Eight a=open.getFirstNode();**

**if (a.h()==0){**

**e=a;**

**break;**

**}**

**open.removeFromFront();**

**a.ex();**

**closed.insertAtBack(a);**

**for(int i=0;i<a.ex().length;i++){**

**if(!a.ex()[i].hasIt(open) && !a.ex()[i].hasIt(closed)){**

**open.insertAtBack(a.ex()[i]);**

**}**

**}**

**listSort(open);**

**}**

**e.listAll(e);**

**System.out.println("open表为：");**

**open.print();**

**System.out.println("closed表为：");**

**closed.print();**

**}**

**}**

**List 代码是：**

**class ListNode{**

**public Eight data;**

**public ListNode next;**

**public ListNode()**

**{**

**next=null;**

**}**

**public ListNode(Eight o)**

**{**

**data=o;**

**next=null;**

**}**

**public ListNode(Eight o,ListNode nextNode)**

**{**

**data=o;**

**next=nextNode;**

**}**

**// Return the Eight in this node**

**public Eight getEight()**

**{**

**return data;**

**}**

**public ListNode getnext()**

**{**

**return next;**

**}**

**}**

**public class List{**

**public ListNode firstNode;**

**public ListNode lastNode;**

**public int length;**

**private String name; // String like "list" used in printing**

**public List(String s)**

**{**

**name=s;**

**firstNode=lastNode=null;**

**length=0;**

**}**

**//Constructor: Constructor an empty List with "List" as the name**

**public List(){**

**this("list");**

**}**

**public Eight getFirstNode(){**

**return firstNode.data;**

**}**

**//Insert an Eight at the front of the List If List is empty, firstNode**

**//and lastNode refer to same Eight. Otherwise,firstNode refers to new node.**

**public void insertAtFront(Eight insertItem)**

**{**

**if(isEmpty())//如果链表为空,返回true,否则返回false.**

**firstNode=lastNode=new ListNode(insertItem);**

**else**

**firstNode=new ListNode(insertItem,firstNode);**

**length++;**

**}**

**//Insert an Eight at the end of the List If List is empty, firstNode and**

**//lastNode refer to same Eight. Otherwise, lastNode's next instance variable refers to new node.**

**public void insertAtBack(Eight insertItem)**

**{**

**if(isEmpty())**

**firstNode=lastNode=new ListNode(insertItem);**

**else**

**lastNode=lastNode.next=new ListNode(insertItem);**

**length++;**

**}**

**// Remove the first node from the List.**

**public Eight removeFromFront() throws EmptyListException**

**{**

**Eight removeItem=null;**

**if(isEmpty())**

**throw new EmptyListException(name);**

**removeItem=firstNode.data; //retrieve the data reset the firstNode and lastNode references**

**if(firstNode.equals(lastNode))**

**firstNode=lastNode=null;**

**else**

**firstNode=firstNode.next;**

**length--;**

**return removeItem;**

**}**

**//Remove the last node from the List**

**public Eight removeFromBack() throws EmptyListException**

**{**

**Eight removeItem=null;**

**if(isEmpty())**

**throw new EmptyListException(name);**

**removeItem=lastNode.data; //retrieve the data reset the firstNode and lastNode references**

**if(firstNode.equals(lastNode))**

**firstNode=lastNode=null;**

**else**

**{**

**ListNode current=firstNode;**

**while(current.next !=lastNode){**

**current=current.next;**

**}**

**lastNode=current;**

**current.next=null;**

**}**

**length--;**

**return removeItem;**

**}**

**//Return true if the List is empty.**

**public boolean isEmpty(){**

**return firstNode==null;**

**}**

**//Output the List contents**

**public void print()**

**{**

**if(isEmpty()){**

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

**return;**

**}**

**ListNode current=firstNode;**

**do**

**{**

**current.data.print();**

**System.out.println("fֵΪ:"+current.data.f());**

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

**current=current.next;**

**}while(current !=null);**

**}**

**}**

**// Class EmptyListException definition**

**class EmptyListException extends RuntimeException{**

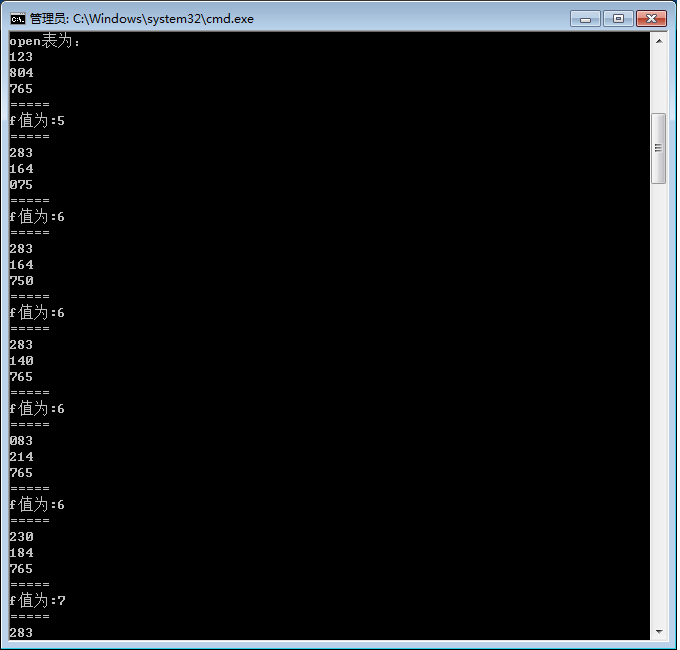
**public EmptyListException(String name)**

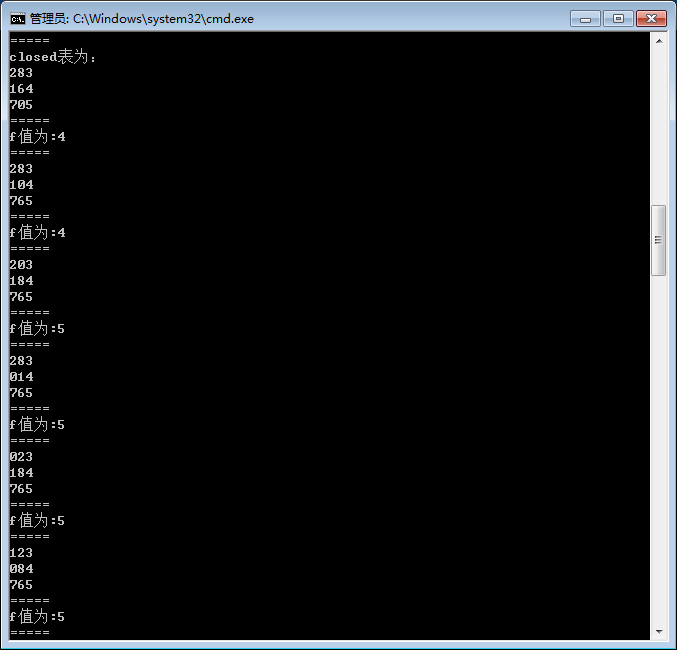
**{**

**super("The " + name + " is empty");**

**}**

**}**

**程序截图如下：**

****