華中科技大學

课程报告

课程名称： 人工智能导论

专业班级：CS1802

学　　号：U201814531

姓　　名：李响

指导教师：冯琪

报告日期：2019. 6

计算机科学与技术学院

目 录

1.1 摘要

迷宫寻路问题是人工智能中的有趣问题，如何表示状态空间和搜索路径是寻路问题的重点，本次研究，我所采用的是 A*搜索算法，在此迷宫问题中，使用 “曼哈顿距离”作为搜索的启发信息，搜索过程为广度优先搜索+贪心策略，并最后打出解路径。

关键词：A*算法，曼哈顿距离

1.2 问题描述

在一个 m 行 n 列的迷宫中求从入口到出口的一条最短路径。数字 0 和 1 分别表示迷宫中的通道和障碍物，数字 5 和 6 分别表示迷宫的出口和入口（出口与入口唯一）。请设计一个程序，对任意设定的迷宫，求出从入口到出口的通路，若没有通路，则直接结束程序，并且提示寻路失败。

1.3 算法实现与原理及其知识表示

1.3.1 算法原理、知识表示及代码实现

由于迷宫已经被简化为由“0”“1”“5”“6”四个数字表示而成的搜索区域，这种方法将我们的搜索区域简化成了一个普通的二维数组。数组中的每一个元素表示对应的一个点。通过找出从起始点到终点所经过的点，就能得到路径。

我设计的算法包括三个部分，分别是读入迷宫数据部分，A*算法搜索路径和打印解路径部分以及帮助 open 表进行排序的部分。其中读入迷宫部分采用二维整型数组进行储存，排序算法采用的是关于指针交换的链表冒泡排序算法，原理和实现不在此赘述，代码详见图 1-2。

```
45  /*读入数据函数，用以读入迷宫信息和确定初始结束位置*/
46  int scanf_map(int m,int n,int *px,int *py,int *pex,int *pey)
47  {
48      int i,j;
49      for(i=0;i<m;i++)
50        for(j=0;j<n;j++)
51        {
52          scanf("%d",&Pmaze[i][j]);
53          if(Pmaze[i][j]==5) *px=i,*py=j;    /*确定起始和终点位置*/
54          else if(Pmaze[i][j]==6) *pex=i,*pey=j;
55        }
56      return 0;
57  }
```

图 1 读入迷宫节点数据函数代码图

```
/*链表排序算法*/
void fsort(struct open **phead,int n,struct open **pend)
{
    int i,j;
    struct open *pNew=*phead,*pFront=*phead;
    for(i=0;i<n-1;i++)
    for(pNew=pFront=*phead,j=0;j<n-i-1;j++,pFront=pNew,pNew=pNew->pnext)
        if(pNew->price > pNew->pnext->price)
        {
            if(pNew==*phead)
            {
                *phead=pNew->pnext;
                pNew->pnext=pNew->pnext->pnext;
                (*phead)->pnext=pNew;
                pNew=*phead;
            }
            else
            {
                pFront->pnext=pNew->pnext;
                pNew->pnext=pFront->pnext->pnext;
                pFront->pnext->pnext=pNew;
                pNew=pFront->pnext;
            }
        }
        pNew=*phead;
        while(pNew)
        {
            *pend=pNew;
            pNew=pNew->pnext;
        }
}
```

图 2 链表结点排序（交换指针）算法代码图

A*算法是迷宫问题解决算法的核心算法，A*算法有两个重要的数据列表：其中一个被用来寻找最短路径点的 open 表，另一个是记录下不会再被考虑点的 closed 表。由于本次的算法实现是采用比较原始的 C 语言实现，所以我并没有完全设计出 open 表和 closed 表，而是采用一个链表来同时模拟两个数据列表的方式完成算法，具体过程如下。

首先，创建一个头结点用于储存初始点的坐标信息以及估价函数值，并将之作为 closed 表的第一个结点，其前指针指向为 NULL，然后把所有与初始结点位置相邻的可通行点（即数字 0 所表示的可通行点以及数字 6 所表示的终点），以这些结点按照一定顺序依次加入链表，其前指针指向起始结点相连，这些结点之间则以尾指针来连接（这样便于排序），通过以上步骤就是将结点添加到 open 表中的全过程，然后通过排序选出一个估值函数值最小的点，将起始结点的尾指针指向该点（本步骤相当于是将节点加入 closed 表中），然后以新加入 closed 表中的结点作为初始结点重复进行这个过程，直到终点结点加入 open 表中，或者 open 表全部清空仍未找到终点结点，则寻路失败，具体代码实现过程见图 3-4，下文将简要描述估值函数的概念。

```
/*核心A*算法*/
int find_path(int startx,int starty,int endx,int endy,int p,int q)
{
    int m,n=0;
    struct open *phead,*pheadopen,*pEnd,*pNew;
    int i=startx,j=starty;
    phead=(struct open *)malloc(sizeof(struct open));
    phead->x=i;
    phead->y=j;
    phead->depth=0;
    phead->pfront=NULL;
    phead->pnext=NULL;
    pheadopen=phead;
    pEnd=phead;

    while(i!=endx||j!=endy)
    {
        for(m=0;m<4;m++)
        if((Pmaze[i+dx[m]][j+dy[m]]==0||Pmaze[i+dx[m]][j+dy[m]]==6)
           &&i+dx[m]>=0&&i+dx[m]<p&&j+dy[m]>=0&&j+dy[m]<q)
        {
            if(Pmaze[i+dx[m]][j+dy[m]]==0) Pmaze[i+dx[m]][j+dy[m]]=2;
            pNew=(struct open *)malloc(sizeof(struct open));
            pNew->x=i+dx[m];
            pNew->y=j+dy[m];
            pNew->pfront=pheadopen;
            pNew->price=abs(endx-pNew->x)+abs(endy-pNew->y);
            pNew->price=pNew->pfront->depth+pNew->price;
            pNew->depth=pNew->pfront->depth+1;
            pEnd->pnext=pNew;
            pNew->pnext=NULL;
            pNew->pfront=pheadopen;
            pEnd=pEnd->pnext;
            n++;
        }
        if(pheadopen->pnext==NULL)
        {
            printf("抱歉，无法到达出口!\n");
            return 0;
        }
        fsort(&pheadopen->pnext,n,&pEnd);
        i=pheadopen->pnext->x;
        j=pheadopen->pnext->y;
        n--;
        pheadopen=pheadopen->pnext;
    }
    get_dist(&pheadopen,p,q);
}
```

图 3-4 核心 A*算法代码图

估价函数的一般形式为 f(n)= g(n)+h(n)，其中 g(n)为从初始节点 S0 到某个节点 n 的代价，故又称为代价函数，h(n)为搜索过程中产生的某种启发信息，故其又称作启发函数。在迷宫问题中，g(n)为搜索的深度，也即是行走的步数，而启发函数 h(n)则采用用“曼哈顿距离”，计算出离终点水平和垂直方向上的长度和。

搜索过程为广度优先搜索（拓展所有可行的子节点加入 open 表）+贪心策略（选取估价函数值最小的点加入 closed 表中）。当终点结点被加入到 open 表作为待检验节点时，表示路径被找到，此时应终止循环；或者当 open 表为空，表明已无可添加的新节

点，则表明着无路径可达终点，此时也终止循环。从终点开始沿前指针向父节点回溯到起点，记录整个遍历中得到的节点坐标，便得到了最优路径，打印路径的代码如下图 5。

```
/*路径打印算法*/
int get_dist(struct open **phead,int m,int n)
{
    int i=0,j;
    struct open *pNew=*phead;
    while(pNew->pfront)
    {
        i++;
        if(Pmaze[pNew->x][pNew->y]==2)Pmaze[pNew->x][pNew->y]=-6;
        printf("(%d,%d)<-",pNew->x+1,pNew->y+1);
        if(i%8==0) printf("\n");
        pNew=pNew->pfront;
    }
    printf("(%d,%d)\n\n",pNew->x+1,pNew->y+1);
    printf("本次寻路成功，所走步数为：%d \n\n",i);
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
            printf("%c ",Pmaze[i][j]+48);
        printf("\n");
    }
    return 0;
}
```

图 5 打印路径算法代码图

1.3.2 算法流程图

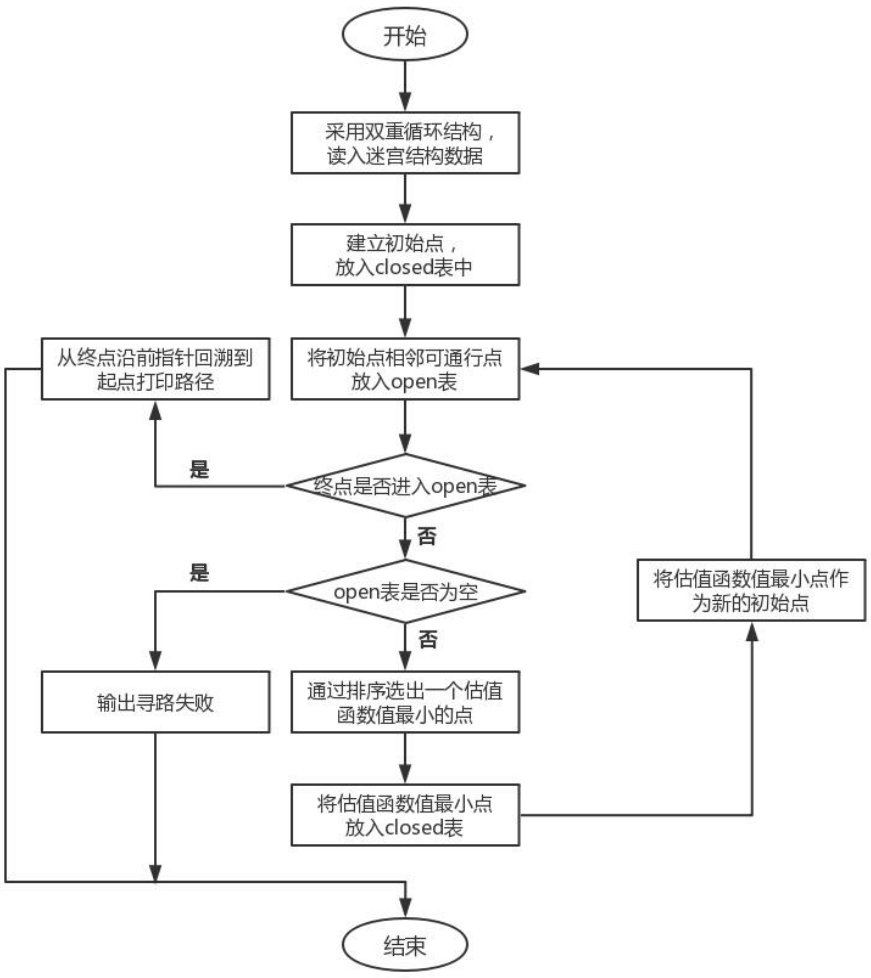

图 6 算法流程图

1.3.3 程序代码及开发工具

开发工具：C 语言，codeblocks

```
#include <stdio.h>
#include<stdlib.h>
#include<string.h>
#define M 2000
int Pmaze[M][M];
/*open 结构链表用以存放开放结点以及关闭结点，xy 表示坐标位置；
   price 是估值函数的数值，depth 是搜索深度，*pfront 用以回溯结点*/
struct open{
	int x, y;
	int price,depth;
	struct open *pnext;
	struct open *pfront;
};
/*迷宫行走的方向，分别为行方向和列方向*/
const int dx[4] = {1, -1, 0, 0};
const int dy[4] = {0, 0, 1, -1};
int scanf_map(int m,int n,int *px,int *py,int *pex,int *pey);
int find_path(int startx,int starty,int endx,int endy,int p,int q);
void fsort(struct open **phead,int n,struct open **pend);
int get_dist(struct open **phead,int m,int n);
int main()
{
	int m,n;
	struct open **phead;
	int startx,starty,*px=&startx,*py=&starty;
	int endx,endy,*pex=&endx,*pey=&endy;
	/*数据注意事项*/
	printf("本算法采用了 A*算法求解迷宫路径问题;\n");
	printf("算法采用以“曼哈顿距离”作为启发信息进行启发式搜索;\n");
	printf("搜索过程为广度优先搜索（估值函数值相同时采用深度策略）\n");
	printf("+贪心策略选择开销最小的状态节点进行拓展;\n");
	printf("下面,您应该根据提示输入各种迷宫的信息;\n");
	printf("输入: 行数和列数，例如：5 3\n");
	scanf("%d%d",&m,&n);
	printf("从下一行开始输入迷宫信息,注意\"1\"代表墙壁,\"0\"代表通道,\n");
	printf("\"5\"代表起点,\"6\"代表终点数字之间请用空格隔开,范例如下:\n");
	printf("5 0 0 0 0 0 0 0 0 0\n1 1 1 1 1 1 1 1 1 0\n1 0 0 0 0 0 0 0 1 0\n");
	printf("1 0 0 1 1 1 1 0 1 0\n1 1 0 0 0 0 1 0 1 0\n");
	printf("1 1 1 1 1 0 1 0 1 0\n6 0 1 0 1 0 1 0 1 0\n");
	printf("1 0 1 0 1 0 1 0 1 0 \n1 0 1 0 1 0 1 0 1 0\n1 0 0 0 0 0 1 0 0 0\n\n");
	scanf_map(m,n,px,py,pex,pey);
	find_path(startx,starty,endx,endy,m,n);
	return 0;
```

```
}
/*读入数据函数，用以读入迷宫信息和确定初始结束位置*/
int scanf_map(int m,int n,int *px,int *py,int *pex,int *pey)
{
	int i,j;
	for(i=0;i<m;i++)
	   for(j=0;j<n;j++)
	   {
	      scanf("%d",&Pmaze[i][j]);
	      if(Pmaze[i][j]==5) *px=i,*py=j;     /*确定起始和终点位置*/
	      else if(Pmaze[i][j]==6) *pex=i,*pey=j;
	   }
	return 0;
}
/*核心 A*算法*/
int find_path(int startx,int starty,int endx,int endy,int p,int q)
{
	int m,n=0;
	struct open *phead,*pheadopen,*pEnd,*pNew;
	int i=startx,j=starty;
	phead=(struct open *)malloc(sizeof(struct open));
	phead->x=i;
	phead->y=j;
	phead->depth=0;
	phead->pfront=NULL;
	phead->pnext=NULL;
	pheadopen=phead;
	pEnd=phead;
	while(i!=endx||j!=endy)
	{
		for(m=0;m<4;m++)

if((Pmaze[i+dx[m]][j+dy[m]]==0||Pmaze[i+dx[m]][j+dy[m]]==6)&&i+dx[m]>=0&&i+d
x[m]<p&&j+dy[m]>=0&&j+dy[m]<q)
		{
			if(Pmaze[i+dx[m]][j+dy[m]]==0) Pmaze[i+dx[m]][j+dy[m]]=2;
			pNew=(struct open *)malloc(sizeof(struct open));
			pNew->x=i+dx[m];
			pNew->y=j+dy[m];
			pNew->pfront=pheadopen;
			pNew->price=abs(endx-pNew->x)+abs(endy-pNew->y);
			pNew->price=pNew->pfront->depth+pNew->price;
			pNew->depth=pNew->pfront->depth+1;
			pEnd->pnext=pNew;
			pNew->pnext=NULL;
			pNew->pfront=pheadopen;
```

```
            pEnd=pEnd->pnext;
            n++;
        }
        if(pheadopen->pnext==NULL)
        {
            printf("抱歉，无法到达出口!\n");
            return 0;
        }
        fsort(&pheadopen->pnext,n,&pEnd);
        i=pheadopen->pnext->x;
        j=pheadopen->pnext->y;
        n--;
        pheadopen=pheadopen->pnext;
    }
    get_dist(&pheadopen,p,q);
}
/*链表排序算法*/
void fsort(struct open **phead,int n,struct open **pend)
{
    int i,j;
    struct open *pNew=*phead,*pFront=*phead;
    for(i=0;i<n-1;i++)
    for(pNew=pFront=*phead,j=0;j<n-i-1;j++,pFront=pNew,pNew=pNew->pnext)
        if(pNew->price > pNew->pnext->price)
        {
            if(pNew==*phead)
            {
                *phead=pNew->pnext;
                pNew->pnext=pNew->pnext->pnext;
                (*phead)->pnext=pNew;
                pNew=*phead;
            }
            else
            {
                pFront->pnext=pNew->pnext;
                pNew->pnext=pFront->pnext->pnext;
                pFront->pnext->pnext=pNew;
                pNew=pFront->pnext;
            }
        }
        pNew=*phead;
        while(pNew)
        {
            *pend=pNew;
            pNew=pNew->pnext;
        }
```

```
}
/*路径打印算法*/
int get_dist(struct open **phead,int m,int n)
{
	int i=0,j;
	struct open *pNew=*phead;
	while(pNew->pfront)
	{
		i++;
		if(Pmaze[pNew->x][pNew->y]==2)Pmaze[pNew->x][pNew->y]=-6;
		printf("(%d,%d)<-",pNew->x+1,pNew->y+1);
		if(i%8==0) printf("\n");
		pNew=pNew->pfront;
	}
	printf("(%d,%d)\n\n",pNew->x+1,pNew->y+1);
	printf("本次寻路成功，所走步数为：%d \n\n",i);
	for(i=0;i<m;i++)
	{
		for(j=0;j<n;j++)
			printf("%c ",Pmaze[i][j]+48);
		printf("\n");
	}
	return 0;
}
```

1.4 运算测试结果

【测试结果说明】 测试结果分为三部分，第一部分是寻路所得的最短路径，第二部分是最短路径的步数，第三部分是路径示意图，其中数字 0 代表的是未加入 open 表的结点，数字 1 表示障碍物，数字 5 和 6 分别表示起点和终点，数字 2 表示加入 open 表但未加入 closed 表的结点，字符*表示路径位置，详见图 7-9。

```
本算法采用了A*算法求解迷宫路径问题;
算法采用以“曼哈顿距离”作为启发信息进行启发式搜索;
搜索过程为广度优先搜索（估值函数值相同时采用深度策略）
+贪心策略选择开销最小的状态节点进行拓展;
下面,您应该根据提示输入各种迷宫的信息;
输入: 行数和列数，例如：5 3
10 10
从下一行开始输入迷宫信息，注意"1"代表墙壁，"0"代表通道，
"5"代表起点，"6"代表终点，数字之间请用空格隔开，范例如下：
5 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 0
1 0 0 0 0 0 0 0 1 0
1 0 0 1 1 1 1 0 1 0
1 1 0 0 0 0 0 1 0 1 0
1 1 1 1 1 1 0 1 0 1 0
6 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0 1 0
1 0 0 0 0 0 1 0 0 0
```

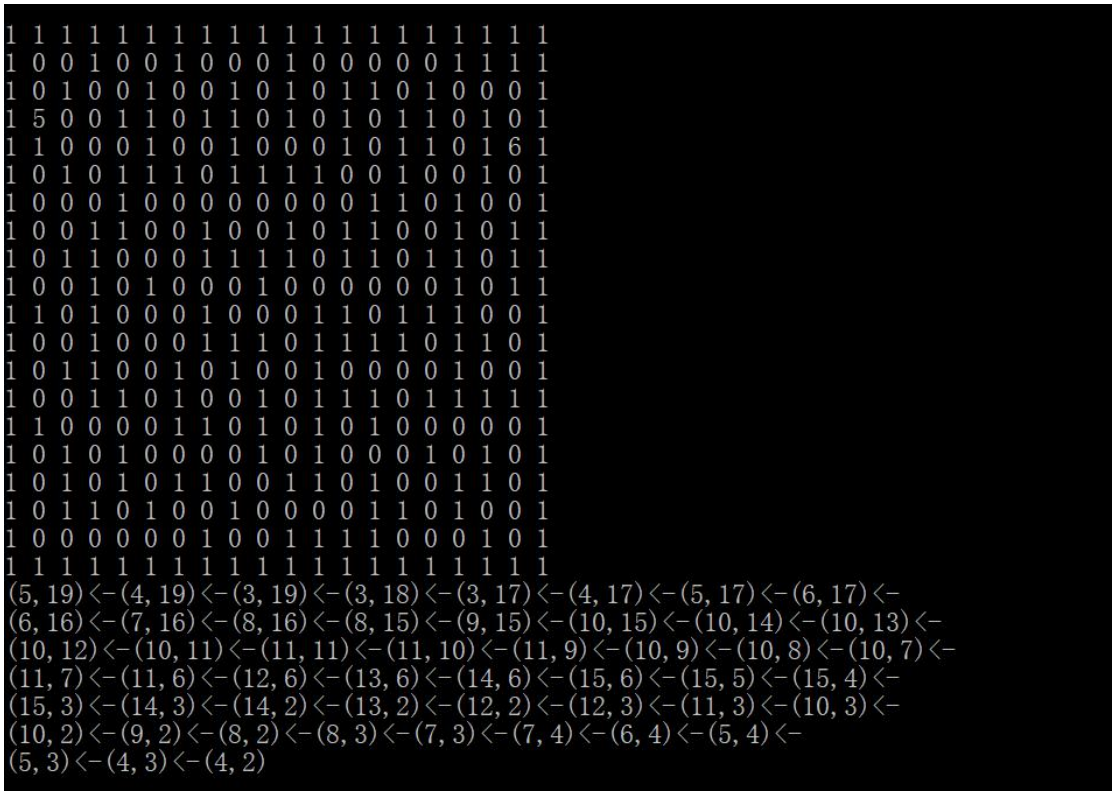

图 7-9 运行测试结果示意图

1.5 小结与展望

本次解决迷宫问题的算法是基础的 A*算法，由上图可以看出，在处理迷宫问题上，A*算法的优化程度并不高，其运行效率在许多情况下近似于宽度优先搜索的算法，尤其是在起点终点分别是在对角上时，其浪费大量时间扩展对称的结点。对我来说，在 A*

算法的原理比较容易理解，但实际动手操作时才发现问题重重，由于本学期有些疏于编程的原因，导致我出现了一些比较低级的错误，所幸基础仍在，低级的错误很快就被解决了。困扰我最多的问题是关于 open 表和 closed 表的实现问题，纸面上将节点加入列表或将节点从列表中删除都是显而易见且轻而易举的，但是实际操作就显得比较繁琐。特别的是，有我并不会使用 C++编程，导致我无法使用 C++中快捷的队列及其排序算法来完成效果，所以我就只能采用链表的方式模拟 open 表和 closed 表，使用指针来明确他们之间的关系，具体的实现方式已经在算法实现中叙述完全，此处就不再赘述。除了这个最大的难题，还有许多的细节问题，比如迷宫如何抽象化，数据的读入和输出，结点的标记等等，这些问题都只有亲身经历才能有所体会。

总而言之，这次的研究给我带来了巨大的收获，也让我体会到了人工智能这门课的巨大魅力，人工智能导论课是我上大学以来第一次接触人工智能这这一项技术，在课程中，通过老师的讲解，我对人工智能有了一些简单的认识，我了解到这门课真的是一门富有挑战性的学科，我希望我在之后的学习中继续学习相关方面的知识，持续丰富自己的知识库。

致谢

在本次研究报告中，我要感谢冯琪老师精彩的课程讲解，是他精彩而详尽的讲解使得我可以快速完成这一次的研究报告。同时，感谢在网上的 A*算法的分析分享博主。再次谢谢你们！