**ROLL NO. 50**

**ASSIGNMENT No. 2 : BEST FIRST SEARCH AND A\* ALGORITHM**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int matrix[6][6];

int rowx[8]={1,2,3,4,4,4,3,2};

int colx[8]={1,1,1,2,3,4,4,4};

int start[1][2]={{4,0}},goal[1][2]={{3,3}};

int N =8;

int R=6;

int C=6;

/\*int matrix[5][7];

int rowx[9]={1,1,1,2,2,2,2,2};

int colx[9]={1,2,5,1,2,3,4,5};

int start[1][2]={{3,0}},goal[1][2]={{1,3}};

int N =8;

\*/

struct path

{

int row;

int col;

int heu;

struct path \* next;

struct path \* parent;

};

struct path \* headc = NULL;

struct path \* heado = NULL;

struct stack

{

int row;

int col;

};

struct stack stk[50];

void print()

{

printf("---------------------------------------------------\n");

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

{

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

{

if(matrix[i][j]==-1)

{

printf("|X|\t");

}

else if(i==start[0][0] && start[0][1]==j)

printf("|S|\t");

else

printf("|%d|\t",matrix[i][j]);

}

printf("\n");

printf("---------------------------------------------------\n");

}

}

void calc\_heuristc()

{

int k,l,count;

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

{

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

{

matrix[i][j]=abs(goal[0][0]-i)+abs(goal[0][1]-j);

}

}

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

matrix[rowx[i]][colx[i]]=-1;

}

struct path \* Astar(struct path \* temp)

{

struct path \* t,\*tempo;

int r,c,count,counth=0;

int sucr[4]={-1,-1,-1,-1};

int succ[4]={-1,-1,-1,-1};

if(temp->row==goal[0][0] && temp->col==goal[0][1])

{

return temp;

}

r=temp->row;

c=temp->col;

if((r-1)>=0 && matrix[r-1][c]!=-1)

{

sucr[0]=r-1;

succ[0]=c;

}

if((c+1)<C && matrix[r][c+1]!=-1)

{

sucr[1]=r;

succ[1]=c+1;

}

if((r+1)<R && matrix[r+1][c]!=-1)

{

sucr[2]=r+1;

succ[2]=c;

}

if((c-1)>=0 && matrix[r][c-1]!=-1)

{

sucr[3]=r;

succ[3]=c-1;

}

printf("\n\n");

/\*for(int i =0; i<4;i++)

{

printf("(%d,%d)",sucr[i],succ[i]);

}\*/

t=headc;

while(t!=NULL)

{

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

{

//printf("\n%d==%d && %d==%d\n",t->row,sucr[i],t->col,succ[i]);

if(t->row==sucr[i] && t->col==succ[i])

{

//printf("\*\*Came\*\*");

sucr[i]=-1;

succ[i]=-1;

}

}

t=t->next;

}

// r=sucr[0];

// c=succ[0];

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

{

if(sucr[i]!=-1)

{

if(heado==NULL)

{

heado=(struct path \*)malloc(sizeof(struct path));

heado->row=sucr[i];

heado->col=succ[i];

t=temp;

counth=0;

while(t!=headc)

{

counth++;

t=t->parent;

}

counth++;

heado->heu=counth+matrix[sucr[i]][succ[i]];

heado->next=NULL;

heado->parent=temp;

}

else

{

tempo = heado;

while(tempo->next!=NULL)

{

tempo=tempo->next;

}

tempo->next=(struct path \*)malloc(sizeof(struct path));

tempo->next->row=sucr[i];

tempo->next->col=succ[i];

t=temp;

counth=0;

while(t!=headc)

{

counth++;

t=t->parent;

}

counth++;

tempo->next->heu=counth+matrix[sucr[i]][succ[i]];

tempo->next->next=NULL;

tempo->next->parent=temp;

}

}

}

printf("\n\n");

t=heado;

while(t!=NULL)

{

printf("(%d,%d)",t->row,t->col);

t=t->next;

}

tempo=heado;

t=heado;

while(tempo!=NULL)

{

if(tempo->heu<t->heu)

{

t=tempo;

}

tempo=tempo->next;

}

temp->next=(struct path \*)malloc(sizeof(struct path));

temp=temp->next;

temp->row=t->row;

temp->col=t->col;

temp->heu=t->heu;

//printf("\tchosen->%d,%d",temp->row,temp->col);

temp->next=NULL;

temp->parent=t->parent;

tempo=heado;

if(t==heado)

{

heado=heado->next;

free(t);

}

else

{

while(tempo->next!=t)

{

tempo=tempo->next;

}

tempo->next=t->next;

free(t);

}

printf("\t(%d,%d,%d)\t",temp->row,temp->col,temp->heu);

t=heado;

tempo=NULL;

/\* if(tempo->row==temp->row && tempo->row==temp->row )

{

heado=t->next;

free(tempo);

tempo=heado;

}\*/

while(t!=NULL)

{

if(t->row==temp->row && t->row==temp->row )

{

if(t==heado)

{

heado=t->next;

free(t);

t=heado;

}

else

{

tempo->next=t->next;

free(t);

t=tempo->next;

}

}

else

{

tempo=t;

t=t->next;

}

}

Astar(temp);

}

struct path \* bestFS(struct path \* temp)

{

struct path \* t,\*tempo;

int r,c,count;

int sucr[4]={-1,-1,-1,-1};

int succ[4]={-1,-1,-1,-1};

if(temp->row==goal[0][0] && temp->col==goal[0][1])

{

return temp;

}

r=temp->row;

c=temp->col;

if((r-1)>=0 && matrix[r-1][c]!=-1)

{

sucr[0]=r-1;

succ[0]=c;

}

if((c+1)<C && matrix[r][c+1]!=-1)

{

sucr[1]=r;

succ[1]=c+1;

}

if((r+1)<R && matrix[r+1][c]!=-1)

{

sucr[2]=r+1;

succ[2]=c;

}

if((c-1)>=0 && matrix[r][c-1]!=-1)

{

sucr[3]=r;

succ[3]=c-1;

}

printf("\n\n");

t=heado;

while(t!=NULL)

{

printf("(%d,%d) ",t->row,t->col);

t=t->next;

}

t=headc;

while(t!=NULL)

{

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

{

if(t->row==sucr[i] && t->col==succ[i])

{

sucr[i]=-1;

succ[i]=-1;

}

}

t=t->next;

}

// r=sucr[0];

// c=succ[0];

// printf("\n\n");

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

{

if(sucr[i]!=-1)

{

if(heado==NULL)

{

heado=(struct path \*)malloc(sizeof(struct path));

heado->row=sucr[i];

heado->col=succ[i];

heado->next=NULL;

heado->parent=temp;

}

else

{

tempo = heado;

while(tempo->next!=NULL)

{

tempo=tempo->next;

}

tempo->next=(struct path \*)malloc(sizeof(struct path));

tempo->next->row=sucr[i];

tempo->next->col=succ[i];

tempo->next->next=NULL;

tempo->next->parent=temp;

}

}

}

tempo=heado;

t=heado;

while(tempo!=NULL)

{

if(matrix[tempo->row][tempo->col]<matrix[t->row][t->col])

{

t=tempo;

}

tempo=tempo->next;

}

temp->next=(struct path \*)malloc(sizeof(struct path));

temp=temp->next;

temp->row=t->row;

temp->col=t->col;

printf("chosen->%d,%d",temp->row,temp->col);

temp->next=NULL;

temp->parent=t->parent;

tempo=heado;

if(t==heado)

{

heado=heado->next;

free(t);

}

else

{

while(tempo->next!=t)

{

tempo=tempo->next;

}

tempo->next=t->next;

free(t);

}

bestFS(temp);

}

int main()

{

struct path \* temp,\*end;

int select;

int top=0;

calc\_heuristc();

while(1)

{

printf("1.)Best First Search\n2.)A\* Search\n");

scanf("%d",&select);

if(select==1 || select==2)

break;

}

print();

headc=(struct path \*)malloc(sizeof(struct path));

headc->row=start[0][0];

headc->col=start[0][1];

headc->next=NULL;

headc->parent=NULL;

//end=bestFS(headc);

if(select==1)

end=bestFS(headc);

else

end=Astar(headc);

printf("\n\n\n");

temp=headc;

/\*while(temp!=NULL)

{

printf("(%d,%d)\t",temp->row,temp->col);

temp=temp->next;

}\*/

//printf("\nreverse path\n");

while(end!=headc)

{

// printf("(%d,%d)\t",end->row,end->col);

stk[top].row=end->row;

stk[top].col=end->col;

top++;

end=end->parent;

}

stk[top].row=end->row;

stk[top].col=end->col;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PATH\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

for(int i=top; i>-1;i--)

{

printf("(%d,%d)---->",stk[i].row,stk[i].col);

}

printf("REACHED GOAL\n\n");

return 0;

}

/\*

int matrix[5][7]={ {-2,-2,-2,-2,-2,-2,-2},

{-2,-2,-1,100,-2,-1,-2},

{-2,-1,-1,-1,-1,-1,-2},

{-100,-2,-1,-2,-1,-2,-2},

{-2,-2,-2,-2,-2,-2,-2},

};\*/

/\*temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}\*/

**OUTPUT :**



