

به نام نور



تمرین شماره 6 رباتیک

دانشجو: ریحانه نیکوبیان

شماره دانشجویی: 99106747

سال تحصیلی: 1402

Road Mapping طراحی مسیر به روش

میخواهیم مسئله تمرین قبل را به روش Road mapping و به کمک الگوریتم Dijkstra حل کنیم. برای این منظور برنامه ای بنویسید که:

الف) تعداد موانع N ، مختصات دو سر خطوط تشکیل دهنده موانع (هر مانع یک پاره خط است)، مختصات نقطه شروع حرکت و مختصات نقطه هدف از یک فایل متن به نام `input.txt` در مسیر جاری بخواند. فرمت فایل متن به صورت زیر است:

```
N
Sx1, Sy1, Ex1, Ey1
.
.
SxN, SyN, ExN, EyN
X_start, Y_start
X_end, Y_end
```

ب) کوتاه ترین مسیر را محاسبه کند و به همراه کلیه موانع روی یک شکل رسم کند.

کد نوشته شده این سوال به شرح زیر است:

```
clc;
clear;
close all;
data = importdata('input.txt', ' ');
N= data(1);

for i=1:N
    A(i,1)=data(4*i-2);
    A(i,2)=data(4*i-1);    % save point of obstacle
    A(i,3)=data(4*i);
    A(i,4)=data(4*i+1);
end

S=[data(N*4+2),data(N*4+3)];
F=[data(N*4+4),data(N*4+5)];    % save start and end point
start=S;
final=F;

plot([S(1),F(1)],[S(2),F(2)],'ro','Linewidth',2);
hold on

k=0; % polynomyial
B=A;
L=1;
while size(B,1)~=0    % organize obstacle and formation ross
```

```

value=true;
k=k+1;
obs{k}(:,1)=[B(1,1);B(1,2)];
obs{k}(:,2)=[B(1,3);B(1,4)];
R(L,:)= [B(1,1);B(1,2)];
L=L+1;
R(L,:)= [B(1,3);B(1,4)];
L=L+1;
c1=B(1,1);
c2=B(1,2);
c3=B(1,3);
c4=B(1,4);
r=3;
B(1,:)=[];

while value==true && size(B,1)~=0
for j=1:size(B,1)
    if (c3==B(j,1)) && (c4==B(j,2))

        obs{k}(:,r)=[B(j,3);B(j,4)];
        c1=B(j,1);
        c2=B(j,2);
        c3=B(j,3);
        c4=B(j,4);

        R(L,:)= [B(j,3);B(j,4)];
        L=L+1;
        B(j,:)=[];
        r=r+1;
        break

    elseif c3==B(j,3) && c4==B(j,4)
        obs{k}(:,r)=[B(j,1);B(j,2)];
        c1=B(j,3);
        c2=B(j,4);
        c3=B(j,1);
        c4=B(j,2);
        R(L,:)= [B(j,1);B(j,2)];
        L=L+1;

        B(j,:)=[];
        r=r+1;
        break
    elseif j==size(B,1)
        value=false ;

    end

end

end

s=size(R,1);
R(s,:)=[];

```

```

L=L-1;

end

R=Cat(1,R,S) ;
R=Cat(1,R,F) ;

n_obs=size(obs,2);
K=1;
for i=1:n_obs      % formation all side obs
    edj=size(obs{i},2)-1;
    for j=1:edj
        for q=1:edj
            if j==q
                continue
            end

            yall_obs(k,:)=[obs{i}(1,j),obs{i}(2,j),obs{i}(1,q),obs{i}(2,q)];
            k=k+1;
        end
    end
end
yall_obs(1,:)=[];
yall_obs(1,:)=[];
Yall=zeros(size(R,1),size(R,1));
for i=1:(size(R,1))      %recognize possible yall
    for j=1:(size(R,1))
        value=fun(R(i,:),R(j,:),yall_obs);
        Yall(i,j)=value;
    end
end
number_p=size(R,1);
value=ones(number_p,1)*inf;
value(number_p)=0;

path_1=[number_p] ;
j=1 ;

Yall2=Yall;
while j~=number_p      %dikestra
    min=1000;
    for i=1:number_p
        if Yall2(path_1(j),i)~=0
            t=value(path_1(j))+ Yall2(path_1(j),i);
            if t<value(i)
                value(i)=t;
            end

            if value(i)<min
                min=value(i);
                next=i;
            end
        end
    end
end

```

```

        end

    end
    path_1=cat(2,path_1,next);

    for k=1:number_p
        Yall2(path_1(j),k)=0;
        Yall2(k,path_1(j))=0;
    end
    j=j+1;
end

final_path=[number_p-1];
j=1;

while final_path(j)~=number_p
    for i=1:number_p
        if (value(final_path(j))==(value(i)+Yall(final_path(j),i))) && Yall(final_path(j),i)~=0

            next=i;
            final_path=cat(2,final_path,next);
            plot([R(final_path(j),1),R(next,1)], [R(final_path(j),2),R(next,2)], 'b', 'Linewidth',2);
            j=j+1;
            hold on
            break
        end
    end
end

for i=1:n_obs
    o=polyshape(obs{i}(1,:),obs{i}(2,:));
    plot(o);
    hold on
end

```

ابتدا داده ها خوانده می شود. تعداد یال های موانع در N ذخیره می شود. بعد یال های مانع در A ذخیره می شود. نقطه شروع و پایان مسیر هم به ترتیب در S, F ذخیره می شود. بعد وارد یک حلقه **while** می شویم که این حلقه راس هارا بدون تکرار در ماتریس R ، می ریزد. به علاوه یک سل درست می کند که در هر المان آن راس های یکی از موانع چند ضلعی جای می گیرد. این بخش کد به این صورت کار می کند که از ضلع اول یعنی سطر اول ماتریس B که ماتریس A را در آن کپی کردیم شروع می کند و آن را در سلول اول می ریزد. بعد به دنبال راس مشترکی با یکی از دو سر این ضلع در تمام اضلاع می گردد. هر جا پیدا کرد، آن را در همان سلول ذخیره می کند و حالا دنبال راس مشترکی در اضلاع دیگر با این ضلع جدید می گردد تا آخر. وقتی به ضلع آخر رسیدیم یا اصلا مانع یک خط باشد، الگوریتم وارد سلول بعدی می شود. هر مرحله وقتی ضلع جدیدی پیدا شد، برای بهینه سازی ضلع قبلی از ماتریس B

حذف می شود تا موانع تکراری پیدا نشود. این توضیح کلی کد خط 22 تا حدود 82 است. حال ماتریس R همه رؤس بدون تکرار موانع را در خود ذخیره کرده است. در دو سطر آخر آن نقطه ابتدا و انتها را اضافه می کنیم. حال همه راس های گراف را داریم. در هر یک از سلول های سل obs هم ماتریس راس های یک مانع جا دارد.

برای تشکیل گراف ممکن یعنی اتصال دو راس ممکن علاوه بر اضلاع مانع به قطر ها نیز نیاز داریم. بنابراین یک ماتریس به نام yall_obs می سازیم. این حلقه به این شکل کار می کند که در هر سلول obs تمام جایگشت ترتیبی دو راس را به عنوان یک مانع در نظر می گیرد و به صورت یک سطر به ماتریس گفته شده اضافه می کند. (خط 87 تا 101)

حال ما تمام راس ها و تمام موانع را داریم. برای اینکه یال های ممکن گراف را تشکیل بدهیم باید هر دو راس را انتخاب کنیم، خط بین این دو راس را تشکیل دهیم و بررسی کنیم آیا با هریک از این خطوط مختصات دو سر آن را در yall_obs ریختیم برخورد دارد یا نه. اگر با هیچ کدام برخورد نداشت، یعنی این یال شکل می گیرد. برای این منظور این حلقه هر دو نقطه از R را به همراه ماتریس yall_obs به تابع fun می برد. این تابع دو نقطه را گرفته، یک خط تشکیل می دهد و در حلقه برخورد با هر یالی را چک می کند. هر جا که برخورد کرد، مقدار صفر به معنی عدم تشکیل را برمیگرداند و ادامه نمی دهد. اگر هیچ برخوردی نداشت، فاصله دو نقطه را بر می گرداند. معادله برخورد دو خط AB، CD به شرح زیر است:

$$A + \alpha(B - A) = D + \beta(C - D)$$

که با مرتب کردن و به دست آوردن alpha، beta برخورد کردن یا نکردن مشخص می شود. اگر هم الفاهم بتا هر دو بین صفر و یک باشند، یعنی برخورد داریم. در غیر این صورت اگر عدد منفی مثبت یا بی نهایت شود دو پاره خط متقاطع نیستند و برای ما مشکلی ایجاد نمی کند.

```
function value=fun(A,B,obs)
n=size(obs,1);
A=transpose(A);
B=transpose(B);
value=1000;

for i=1:n
C=transpose(obs(i,1:2));
D=transpose(obs(i,3:4));
R=[B-A,D-C];
ans=inv(R)*[D-A];
alpha=ans(1);
beta=ans(2);
if alpha<0.99 && alpha>0.01 && beta<0.99 && beta>0.01
value=0;
break
end
```

```

end
if value~=0
    value=norm(A-B);
end

```

عددی که بر می گردد در یک ماتریس $n \times n$ که n برابر تعداد راس های گراف است ذخیره می کنیم. هر درایه i و j یعنی یال بین دو راس شماره i و j که اگر ممکن نباشد عدد صفر است و اگر یال باشد فاصله دو راس ذخیره شده است. اسم این ماتریس Yall است. (خط 105 تا 110)

حال باید مرحله اول الگوریتم دایکسترا را پیاده کنیم. یعنی فاصله هر راس تا مبدا را مشخص کنیم. یک ماتریس value تشکیل می دهیم که تعداد سطر برابر راس هاست و در هر سطر عدد فاصله راس باید باشد. این مقدار ابتدا برای همه راس ها بی نهایت و برای راس مقصد صفر می گذاریم.

Path_1 مسیر طی شده برای ارزش گذاری هر راس است که از مقصد شروع می شود و تا تمام شدن همه راس ها پیش می رود. از مقصد شروع می شود و راس های مجاور خط نخورده را بررسی می کنیم. عدد هر راس مجاور خط نخورده اگر از حاصل جمع عدد راس فعلی و طول یال بینشون بیشتر باشد اصلاح می شود به حاصل جمع راس فعلی و طول یال و عدد راس مورد نظر در ماتریس value اصلاح می شود. در غیر این صورت ثابت می ماند. بعد از این اصلاح راس مجاور خط نخورده ای که کمترین عدد را دارد می شود راس بعدی ما. دوباره این الگوریتم تکرار می شود.

برای اینکه به راس خط خورده نرویم بعد از گذشتن از یک راس تمام یال هایی که از اون راس عبور می کنند طول صفر می گیرند تا در تکرار کد بررسی نشوند. حال در ماتریس value کمترین فاصله هر راس تا راس مقصد را داریم. (توضیحات کد خط 111 تا 144)

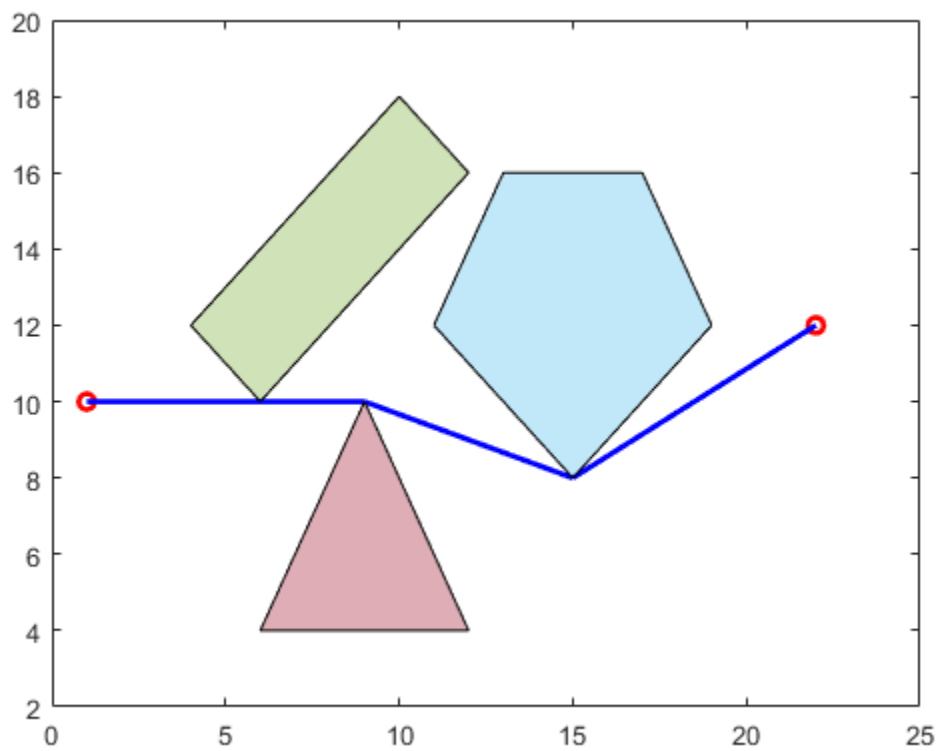
در مرحله آخر، مسیر مورد نظر برای رسیدن از مبدا به مقصد را با final_path نشان دادیم. از مبدا شروع می کنیم و عدد راس را بر میداریم. با استفاده از ماتریس value، Yall بررسی می کنیم هر یال مشترک با این راس که عدد فاصله به علاوه عدد یال وزن مربوطه برابر با عدد راس می شود راس بعدی مسیر ماست. بعد دوباره برای این نقطه جدید، همین الگوریتم تکرار می شود تا به راس مقصد برسیم. در انتها کوتاه ترین مسیر از مبدا تا مقصد را داریم. (خط 145 تا 162)

در نهایت نقاط مبدا ، مقصد، موانع و مسیر را برای یک نمونه دیتا زیر رسم کرده و نشان دادم.

input.txt - Notepad

File Edit Format View Help

```
12
10 18 12 16
12 16 6 10
6 10 4 12
15 8 11 12
4 12 10 18
13 16 17 16
17 16 19 12
19 12 15 8
11 12 13 16
9 10 12 4
12 4 6 4
6 4 9 10
1 10
22 12
```



سوال امتیازی (۰.۲۵)

فرض کنید هندسه ربات به صورت یک m ضلعی محدب مدل شده است که دوران نمیکند. مختصات m راس ربات در نقطه اولیه در انتهای فایل input.txt داده میشود:

m
X1,Y1
.
.
Xm,Ym

مجددا کوتاه ترین مسیر از نقطه شروع به پایان را با کمک اصلاح موانع و بکارگیری الگوریتم دایکسترا بدست آورده و رسم کنید.

برای حل این مسئله می دانیم باید روش موانع گسترش یافته را استفاده کنیم. بعد مسئله دوباره به مسئله نقطه و موانع چند ضلعی محدب تبدیل می شود و با کد قبلی حل می شود. برای این منظور من کد زیر را نوشتم:

```
clc;
clear;
close all;
data = importdata('input2.txt', ' ');
N= data(1);

for i=1:N
    A(i,1)=data(4*i-2);
    A(i,2)=data(4*i-1); % obstacle
    A(i,3)=data(4*i);
    A(i,4)=data(4*i+1);
end

S=[data(N*4+2),data(N*4+3)];
F=[data(N*4+4),data(N*4+5)];
start=S;
final=F;

plot([S(1),F(1)],[S(2),F(2)], 'ro', 'Linewidth', 2);
hold on

M=data((N+1)*4+2);

robot=zeros(M,2);
for i=1:M
    robot(i,1)=data((N+1)*4+2*i+1);
    robot(i,2)=data((N+1)*4+2*i+2); % yall Robot
end
V=zeros(2,M);
```

```

modified_point=transpose(robat(1,:));
for p=1:M

    t=transpose(robat(p,:)-robat(1,:));    % vector v
    V(:,p)=transpose(t);
end

k=0; % polynomyial
B=A;
L=1;
while size(B,1)~=0    % recognize obs
    value=true;
    k=k+1;
    obs{k}{1}=[B(1,1);B(1,2)];
    obs{k}{2}=[B(1,3);B(1,4)];
    R(L,:)=[B(1,1);B(1,2)];
    L=L+1;
    R(L,:)=[B(1,3);B(1,4)];
    L=L+1;
    c1=B(1,1);
    c2=B(1,2);
    c3=B(1,3);
    c4=B(1,4);
    r=3;
    B(1,:)=[];

while value==true && size(B,1)~=0
    for j=1:size(B,1)
        if (c3==B(j,1)) && (c4==B(j,2))

            obs{k}{r}=[B(j,3);B(j,4)];
            c1=B(j,1);
            c2=B(j,2);
            c3=B(j,3);
            c4=B(j,4);

            R(L,:)=[B(j,3);B(j,4)];
            L=L+1;
            B(j,:)=[];
            r=r+1;
            break

        elseif c3==B(j,3) && c4==B(j,4)
            obs{k}{r}=[B(j,1);B(j,2)];
            c1=B(j,3);
            c2=B(j,4);
            c3=B(j,1);
            c4=B(j,2);
            R(L,:)=[B(j,1);B(j,2)];
            L=L+1;

            B(j,:)=[];

```

```

        r=r+1;
        break
    elseif j==size(B,1)
        value=false ;

    end

end

end

s=size(R,1);
R(s,:)=[];
L=L-1;

end

n_obs=size(obs,2);

min_point=zeros(1,n_obs);
for i=1:n_obs % formation all new point for each obstacle and seperate x_min
    min_x=1000;
    edj=size(obs{1,i},2)-1;
    for j=1:edj
        for k=1:M

            new_point{i}(:,(j-1)*4+k)=obs{1,i}{1,j}+v(:,k);
            if min_x>new_point{i}(1,(j-1)*4+k)
                min_x=new_point{i}(1,(j-1)*4+k);
                min_point(i)=(j-1)*4+k;
            end
        end
    end
end

end

for i=1:n_obs
    n_con=size(new_point{i},2) ;
    q1=new_point{i}(:,min_point(i));
    qL=q1-[1;0];
    qc=q1;
    j=1;
    con{i}(1)=min_point(i);
    Poly{i}(:,1)=q1;
    while con{i}(j)~=min_point(i) || j==1

        alpha_min=360;
        for k=1:n_con
            qi=new_point{i}(:,k);
            v=qi-qc;
            v=cat(1,v,0);
            u=qL-qc;
            u=cat(1,u,0);
            cos=(dot(u,v))/(norm(u)*norm(v));

```

```

    K=[0;0;-1];
    sin=(dot(cross(u,v),K))/(norm(u)*norm(v));
    alpha=atan2(sin,cos);
    if alpha<0
        alpha=2*pi+alpha;
    end
    if alpha==0
        alpha=2*pi;
    end
    if alpha<alpha_min
        alpha_min=alpha;
        next=k;
    end
end
j=j+1;
qL=qC;

qC=new_point{i}(:,next);
con{i}(j)=next;
Poly{i}(:,j)=new_point{i}(:,next);

end
obs{i}=polyshape(Poly{i}(1,:),Poly{i}(2,:));
plot(obs{i});
hold on

end
k=1;

for i=1:size(con,2)
    for j=1:size(con{i},2)-1
        Y(k,:)=new_point{i}(1,con{i}(j)),new_point{i}(2,con{i}(j)),
        new_point{i}(1,con{i}(j+1)),new_point{i}(2,con{i}(j+1))];
        k=k+1;
    end
end

end
A=Y;
N=k-1;

A=int64(A);

```

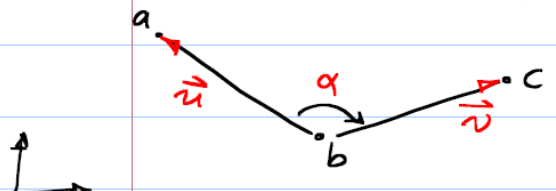
اولین راس ربات را روی همان نقطه شروع باید تعریف شود. همانند قبل اطلاعات را ذخیره می کنیم و اطلاعات راس های ربات را در ماتریس **robot** می ریزیم. یک نقطه که همان راس اول است را به عنوان نقطه نماینده انتخاب می کنیم. بعد از هر راس تا نقطه نماینده یک بردار تشکیل می دهیم. این بردار ها را در ماتریس **V** ذخیره می کنیم. (خط 31 تا 36 کد)

همانند کد قبل موانع را دسته بندی می کنیم و سل obs که رئوس موانع مرتب جای دارد و ماتریس R که راس های این موانع است تشکیل می دهیم. (خط 42 تا 99 کد)

در مرحله بعد باید هر بردار را به راس های مانع اضافه کنیم و نقاط جدید به دست بیاوریم. این نقاط را برای هر مانع در یک ماتریس ذخیره و در سل new_point می ریزیم. همچنین برای هر مانع نقطه ای که کمترین ایکس را دارد مشخص می کنیم و در min_point شماره نقطه مورد نظر را ذخیره می کنیم. (کد خط 100 تا 117)

در این مرحله برای هر مانع باید یک پوشش محدب تشکیل دهیم. برای این منظور برای هر مانع در مرحله اول نقطه ای کمترین ایکس را داشت نقطه qc ، نقطه qc-[1;0] را ql می گیریم. بعد برای تمام نقاط باقی مانده زاویه qLqcqi را حساب می کنیم. نقطه ای که کمترین زاویه در خلاف جهت عقربه های ساعت داشته باشد راس بعدی پوشش محدب ماست. ql مرحله بعدی را برابر qc مرحله قبل قرار می دهیم و راس انتخاب شده qc مرحله جدید است. و دوباره برای همه نقاط دیگر زاویه qLqcqi را حساب می کنیم تا راس بعدی انتخاب شود و همین طور تکرار می کنیم تا نقطه ql اولیه برسیم. زاویه جهت دار نیز به روش زیر حساب می شود. راس های جدید هر مانع در Poly ذخیره می شود. (کد خط 119 تا 165)

زاویه جهت دار



$$\vec{u} \cdot \vec{v} = |\vec{u}| |\vec{v}| \cos \alpha$$

$$\cos \alpha = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}| |\vec{v}|} \quad (1)$$

$$\frac{(\vec{u} \times \vec{v}) \cdot (-\hat{k})}{|\vec{u}| |\vec{v}|} = \sin \alpha \quad (2)$$

$$\Rightarrow \alpha = \text{atan2} \left(\overset{(2)}{\sin \alpha}, \overset{(1)}{\cos \alpha} \right)$$

در این مرحله ضلع های یالهای گسترش یافته به ماتریس تبدیل می شوند تا به عنوان ورودی کد سوال قبل بتوانیم استفاده کنیم. (خط 166 تا 172 خط)

در آخر هم موانع گسترش یافته و نقطه شروع و پایان رسم می شود.

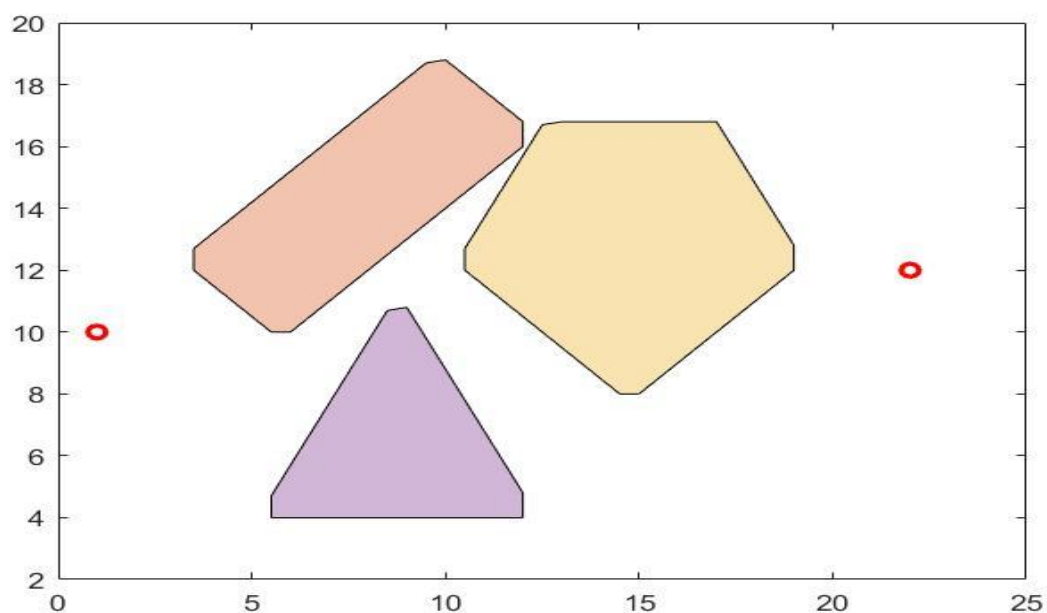
حال این مسئله با حل مسئله ای که نقطه شروع و پایان اولیه را دارد و عملگر یک نقطه است و موانع همین موانع گسترش یافته جدید است تبدیل می شود.

در زیر یک نمونه ورودی برای این کد و موانع گسترش یافته که کد رسم می کند را آوردم.

input2.txt - Notepad

File Edit Format View Help

```
12
10 18 12 16
12 16 6 10
6 10 4 12
15 8 11 12
4 12 10 18
13 16 17 16
17 16 19 12
19 12 15 8
11 12 13 16
9 10 12 4
12 4 6 4
6 4 9 10
1 10
22 12
4
1 10
1 10.8
0.5 10
0.5 10.7
```



ماتریس A که ضلع های موانع گسترش یافته است را در یک note جدید به همراه نقطه شروع و پایان ذخیره می کنیم. برای مثال از کد بالا و مثال مشخص شده A را خروجی می گیریم و note جدید را مانند زیر شکل می دهیم:

```

input3.txt - Notepad
File Edit Format View Help
25
3.5000 12.0000 3.5000 12.7000
3.5000 12.7000 9.5000 18.7000
9.5000 18.7000 10.0000 18.8000
10.0000 18.8000 12.0000 16.8000
12.0000 16.8000 12.0000 16.0000
12.0000 16.0000 6.0000 10.0000
6.0000 10.0000 5.5000 10.0000
5.5000 10.0000 3.5000 12.0000
10.5000 12.0000 10.5000 12.7000
10.5000 12.7000 12.5000 16.7000
12.5000 16.7000 13.0000 16.8000
13.0000 16.8000 17.0000 16.8000
17.0000 16.8000 19.0000 12.8000
19.0000 12.8000 19.0000 12.0000
19.0000 12.0000 15.0000 8.0000
15.0000 8.0000 14.5000 8.0000
14.5000 8.0000 10.5000 12.0000
5.5000 4.0000 5.5000 4.7000
5.5000 4.7000 8.5000 10.7000
8.5000 10.7000 9.0000 10.8000
9.0000 10.8000 12.0000 4.8000
12.0000 4.8000 12.0000 4.0000
12.0000 4.0000 11.5000 4.0000
11.5000 4.0000 6.0000 4.0000
6.0000 4.0000 5.5000 4.0000
1 10
22 12

```

و این text را در کد سوال یک ورودی می دهیم. کوتاه ترین مسیر را برای ما مشخص می کند. دقیقاً همان کد را برای سوال 2 با نام q2_b تکرار کردم فقط آدرس text ورودی متفاوت است.

```

clc;
clear;
close all;
data = importdata('input3.txt', ' ');
N= data(1);

for i=1:N
    A(i,1)=data(4*i-2);
    A(i,2)=data(4*i-1); % save point of obstacle

```

```

    A(i,3)=data(4*i);
    A(i,4)=data(4*i+1);
end

S=[data(N*4+2),data(N*4+3)];
F=[data(N*4+4),data(N*4+5)];    % save start and end point
start=S;
final=F;

plot([S(1),F(1)],[S(2),F(2)], 'ro', 'Linewidth',2);
hold on

k=0; % polynomyial
B=A;
L=1;
while size(B,1)~=0    % organize obstacle and formation ross
    value=true;
    k=k+1;
    obs{k}(:,1)=[B(1,1);B(1,2)];
    obs{k}(:,2)=[B(1,3);B(1,4)];
    R(L,:)=[B(1,1);B(1,2)];
    L=L+1;
    R(L,:)=[B(1,3);B(1,4)];
    L=L+1;
    c1=B(1,1);
    c2=B(1,2);
    c3=B(1,3);
    c4=B(1,4);
    r=3;
    B(1,:)=[];

    while value==true && size(B,1)~=0
        for j=1:size(B,1)
            if (c3==B(j,1)) && (c4==B(j,2))

                obs{k}(:,r)=[B(j,3);B(j,4)];
                c1=B(j,1);
                c2=B(j,2);
                c3=B(j,3);
                c4=B(j,4);

                R(L,:)=[B(j,3);B(j,4)];
                L=L+1;
                B(j,:)=[];
                r=r+1;
                break

            elseif c3==B(j,3) && c4==B(j,4)
                obs{k}(:,r)=[B(j,1);B(j,2)];
                c1=B(j,3);
                c2=B(j,4);
                c3=B(j,1);
                c4=B(j,2);
                R(L,:)=[B(j,1);B(j,2)];

```



```

        L=L+1;

        B(j,:)=[];
        r=r+1;
        break
    elseif j==size(B,1)
        value=false ;

    end

end

end

s=size(R,1);
R(s,:)=[];
L=L-1;

end

R=cat(1,R,S) ;
R=cat(1,R,F) ;

n_obs=size(obs,2);
K=1;
for i=1:n_obs % formation all side obs
    edj=size(obs{i},2)-1;
    for j=1:edj
        for q=1:edj
            if j==q
                continue
            end

            yall_obs(k,:)=[obs{i}(1,j),obs{i}(2,j),obs{i}(1,q),obs{i}(2,q)];
            k=k+1;
        end
    end
end
yall_obs(1,:)=[];
yall_obs(1,:)=[];
Yall=zeros(size(R,1),size(R,1));
for i=1:(size(R,1)) %recognize possible yall
    for j=1:(size(R,1))
        value=fun(R(i,:),R(j,:),yall_obs);
        Yall(i,j)=value;
    end
end
number_p=size(R,1);
value=ones(number_p,1)*inf;
value(number_p)=0;

path_1=[number_p] ;
j=1 ;

```

```

Yall2=Yall;
while j~=number_p    %dijkstra
    min=1000;
    for i=1:number_p
        if Yall2(path_1(j),i)~=0
            t=value(path_1(j))+ Yall2(path_1(j),i);
            if t<value(i)
                value(i)=t;
            end

            if value(i)<min
                min=value(i);
                next=i;
            end

        end

    end

    path_1=cat(2,path_1,next);

    for k=1:number_p
        Yall2(path_1(j),k)=0;
        Yall2(k,path_1(j))=0;
    end
    j=j+1;
end

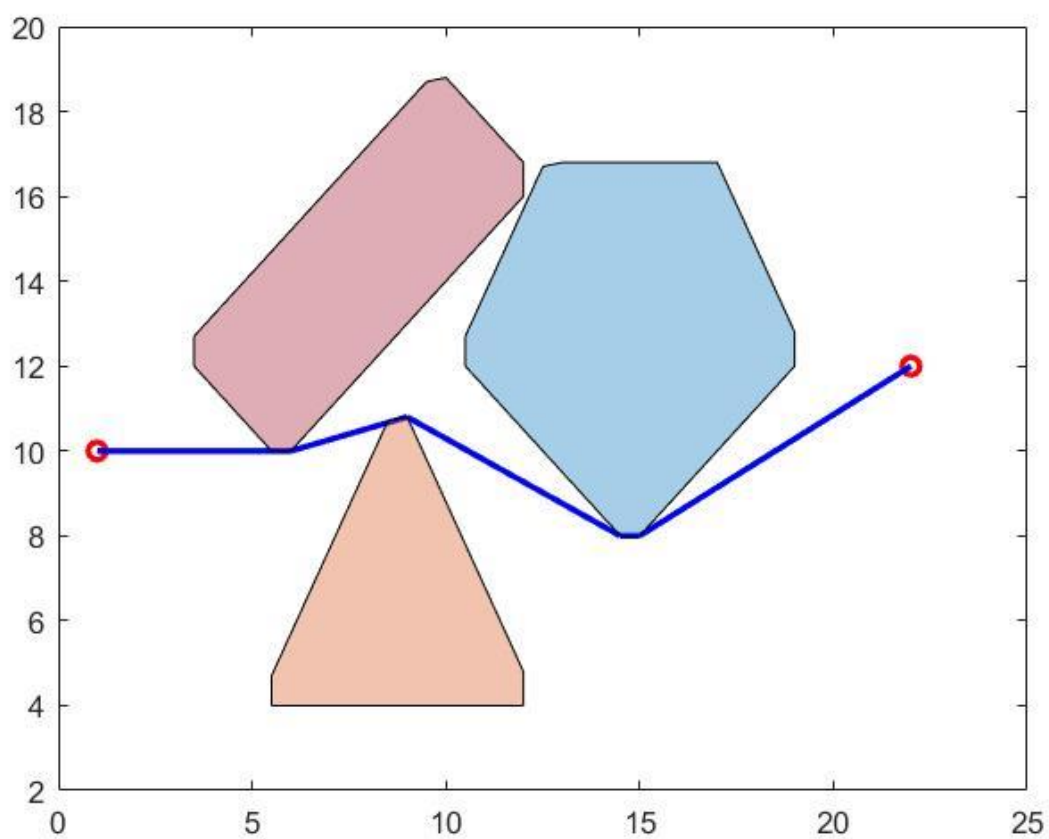
final_path=[number_p-1];
j=1;

while final_path(j)~=number_p
    for i=1:number_p
        if (value(final_path(j))==(value(i)+Yall(final_path(j),i))) && Yall(final_path(j),i)~=0

            next=i;
            final_path=cat(2,final_path,next);
            plot([R(final_path(j),1),R(next,1)], [R(final_path(j),2),R(next,2)], 'b', 'Linewidth',2);
            j=j+1;
            hold on
            break
        end
    end
end

for i=1:n_obs
    o=polyshape(obs{i}(1,:),obs{i}(2,:));
    plot(o);
    hold on
end

```



این مسیر طی شده اولین راس ربات در حضور همان موانع اولیه است . در واقع مسئله را با موانع گسترش یافته و ربات به صورت نقطه معادل سازی کردیم.