Weekly Homework 1

Math Class: Optimization Methods
Zhou FENG
U201510104

September 16, 2018

1 Exercise 1

Use the Dijkstra algorithm and the successive approximation algorithm taught in class to find the minimal distances in a graph.

1.1 Dijkstra Algorithm

1.1.1 Discription of problem

The target undirected graph $G_1 = (V, E, W)$ is showed below as an image plotted by matlab and an adjacent matrix.

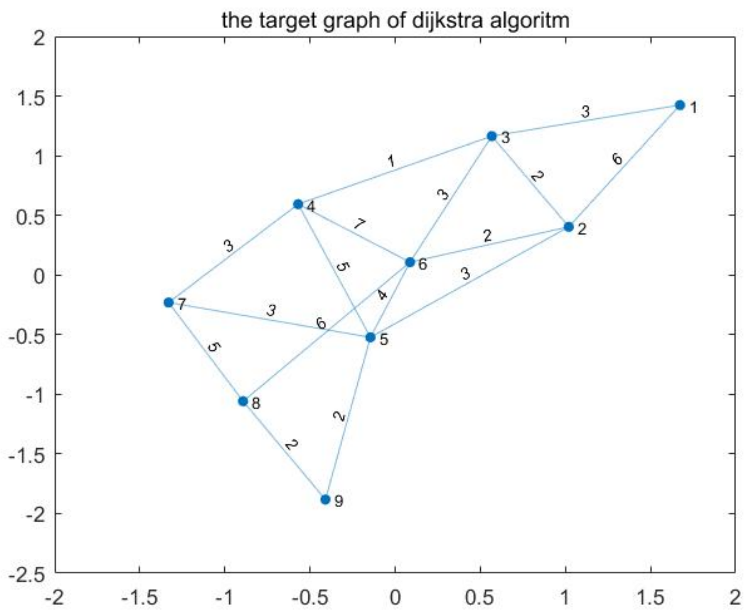

Figure 1: The plotted G_1

	v1	v2	v3	v4	v5	v6	v7	v8	v9
v1	0	6	3	Inf	Inf	Inf	Inf	Inf	Inf
v2	6	0	2	Inf	3	2	Inf	Inf	Inf
v3	3	2	0	1	Inf	3	Inf	Inf	Inf
v4	Inf	Inf	1	0	5	7	3	Inf	Inf
v5	Inf	3	Inf	5	0	4	3	Inf	2
v6	Inf	2	3	7	4	0	Inf	6	Inf
v7	Inf	Inf	Inf	3	3	Inf	0	5	Inf
v8	Inf	Inf	Inf	Inf	Inf	6	5	0	2
v9	Inf	Inf	Inf	Inf	2	Inf	Inf	2	0

Table 1: Adjacent Matrix of G_1

Then apply the Dijksra algorithm to find the minimal distance between V_1 and other vertices.

1.1.2 Solution

Since the weight attached to each edge of G_1 is nonnegative, the principle that the Dijkstra algorithm is based on is assued. Hence the Dijkstra algorithm can be applied properly. The final result is listed below.

	Minimal Distance	Path				
v2	5	1	3	2		
v3	3	1	3			
v4	4	1	3	4		
v5	8	1	3	2	5	
v6	6	1	3	6		
v7	7	1	3	4	7	
v8	12	1	3	6	8	
v9	10	1	3	2	5	9

Table 2: The result of Dijkstra algorithm

1.2 Successive Approximation Algorithm

1.2.1 Discription of problem

The target directed graph $G_2 = (V, E, W)$ is showed below as an image plotted by matlab and an adjacent matrix.

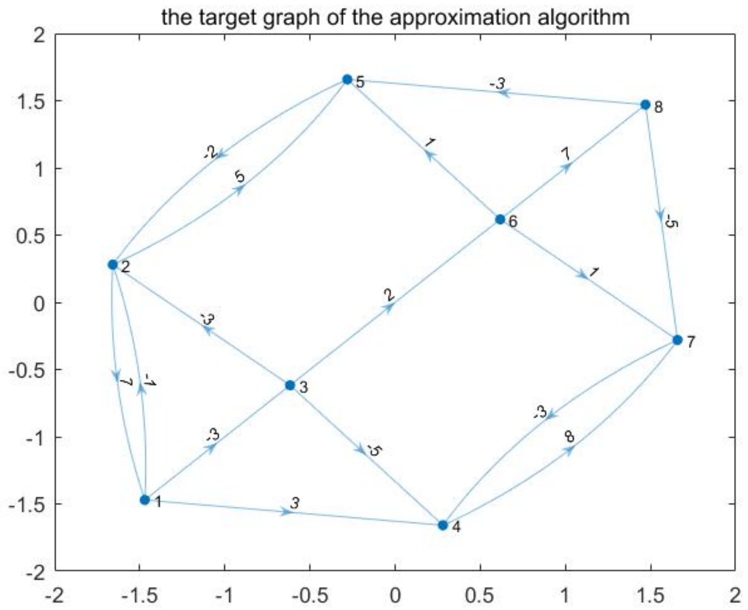

Figure 2: The plotted G_2

Then apply the successive approximation algorithm to find the minimal distance between V_1 and other vertices in G_2.

1.2.2 Solution

The result is listed below.

Vertices	v1	v2	v3	v4	v5	v6	v7	v8	t=1	t=2	t=3	t=4
v1	0	-1	-3	3	Inf	Inf	Inf	Inf	0	0	0	0
v2	7	0	Inf	Inf	5	Inf	Inf	Inf	-1	-6	-6	-6
v3	Inf	-3	0	-5	Inf	2	Inf	Inf	-3	-3	-3	-3
v4	Inf	Inf	Inf	0	Inf	Inf	8	Inf	3	-8	-8	-8
v5	Inf	-2	Inf	Inf	0	Inf	Inf	Inf	Inf	4	-1	-1
v6	Inf	Inf	Inf	Inf	1	0	1	7	Inf	-1	-1	-1
v7	Inf	Inf	Inf	-3	Inf	Inf	0	Inf	Inf	11	0	0
v8	Inf	Inf	Inf	Inf	-3	Inf	-5	0	Inf	Inf	6	6

Table 3: The result of the algorithm

2 Appendix: Matlab Code

2.1 function: main

Contents

- set the target graph of successive approximation method
- plot the graph
- the dijkstra algorithm
- set the target graph of successive approximation method
- plot the graph
- the successive approxiamtion algoritm

```
clc,clear
```

set the target graph of successive approximation method

```
%(mainly the adjacent matrix)
M=[1 6 3 0 0 0 0 0 0;
0 1 2 0 3 2 0 0 0;
0 0 1 1 0 3 0 0 0;
0 0 0 1 5 7 3 0 0;
0 0 0 0 1 4 3 0 2;
0 0 0 0 0 1 0 6 0 ;
0 0 0 0 0 0 1 5 0;
0 0 0 0 0 0 0 1 2;
0 0 0 0 0 0 0 0 1];
```

plot the graph

```
figure(1)
Grph= graph(M,'upper','OmitSelfLoops');
plot(Grph,'EdgeLabel',Grph.Edges.Weight)
title('the target graph of dijkstra algoritm')
```

the dijkstra algorithm

```
D=M+M';
D(find(D==0))=inf;
D=D-diag(diag(D));
for i=2:9
[mydistance mypath]=mydijkstra(D,1,i);
end
```

set the target graph of successive approximation method

```
%(mainly the adjacent matrix)
M=[1 -1 -3 3 0 0 0 0;
7 1 0 0 5 0 0 0;
0 -3 1 -5 0 2 0 0;
0 0 0 1 0 0 8 0;
0 -2 0 0 1 0 0 0;
0 0 0 0 1 1 1 7;
0 0 0 -3 0 0 1 0;
0 0 0 0 -3 0 -5 1];
```

plot the graph

```
figure(2)
Grph= digraph(M,'OmitSelfLoops');
plot(Grph,'EdgeLabel',Grph.Edges.Weight)
title('the target graph of the approximation algorithm')
```

the successive approxiamtion algoritm

```
S=M;
S(find(S==0))=inf;
S=S-diag(diag(S));
stepmat=mystepsapprox(S,1,8)
```

```
stepmat =

0     0     0     0
-1    -6    -6    -6
-3    -3    -3    -3
3    -8    -8    -8
Inf     4    -1    -1
Inf    -1    -1    -1
Inf    11     0     0
Inf   Inf     6     6
```

2.2 function: mydijkstra

```
function [mydistance mypath]=mydijkstra(a,sb,db)
% input:aładjacent matrix
% sbłinitial point, dbłterminal point
% out putmydistancełminmun distance, mypathłminmum path
n=size(a,1); visited(1:n)=0;
distance(1:n)=inf; % store the distance
distance(sb)=0; parent(1:n)=0;
for i=1:n-1
temp=distance;
id1=find(visited==1); %find the marked point
temp(id1)=inf; %put the distance to the marked point to infinity
[t, u] = min(temp); %find the point with minmum marked number
visited(u) = 1; %remember the marked point
id2=find(visited==0); %find the unmarked point
for v = id2
if a(u, v) + distance(u) < distance(v)
distance(v) = distance(u) + a(u, v); %change the number of the point
parent(v) = u;
end
end
end
mypath=[];
if parent(db)˜= 0 %if there exists the path
t = db; mypath = [db];
while t˜= sb
p=parent(t);
mypath=[p mypath];
t=p;
end
end
mydistance=distance(db);
return
end
```

2.3 function: mystepsapprox

```
function stepmat = mystepsapprox( ad, spoint, tpoint )
% input:adładjacent matrix
% spointłinitial point, tpointłterminal point
% outputstepmat- the matrix in the procedure to find the minimal distance
gdegree=length(ad(1,:));
```

```
stepmat=ad(spoint,:)';
step=1;
while 1
tempvector=inf*ones(gdegree,1);
canreach=find(stepmat(:,step)<inf);
%templength=length(canreach);
for i=1:gdegree
canback=find(ad(:,i)<inf);
findex=intersect(canreach,canback);
if ~isempty(findex)
flagvector=stepmat(findex,step)+ad(findex,i);
tempvector(i)=min(flagvector);
else
tempvector(i)=inf;
end
end
stepmat=[stepmat tempvector];
if norm(stepmat(:,step)-tempvector)==0
break
end
step=step+1;
end
```