Introduction to Algorithms : Third Assignment

Liu Chengjun, Student ID: 0490303961

Question 1:

Step 1: Convert Graphic Map into mathematical elements.

Using either adjacency list or matrix is okay. Due to no restriction about efficiency and relatively complex edges, we here use matrix to describe graphic paths.

The matrix below describes paths of the graph. The vertical indexes are starting points of arrow edges, while the horizontal indexes are ending points of arrow edges. The lengths of edges are put between axes.

	1	2	3	4	5	6	7	8	9	10
1	0	5			2			8		
2		0	1							
3			0	4	2	1				
4				0		3	10			
5		3			0	7			3	
6						0	8		5	1
7							0			
8					1			0		
9								2	0	4
10							6			0

Step 2: Present draft of Dijkstra Algorithm.

Before Dijkstra Algorithm the algorithm of RELAX should be introduced first:
If there exists an edge from vertex A to vertex B, compare if the edge shortens distance from starting point to end B by adding edge up on distance from starting point to end A:

$RELAX(A, B, \omega)$
  if $d[B] > d[A] + \omega$
    $d[B] = d[A] + \omega$
    $\pi(B) = A$

If indeed the distance to A plus length of edge is shorter, make distance to B equal to that distance and set predecessor to be A.

Besides, we need initialize data structure used in algorithm.
Denote set of distances to be D , and set of predecessors to be P .

$Initialize(G,s)$

For each point v in V:

$D[v]=\inf$

$P[v]=0$

$D[s]=0$

We make distances to each vertex the maximum such that we can relax the distances, and make predecessors uncertified to fill predecessors from relax process.

For an input of graph set $G=(V,E)$ and starting point s (here we start from 1):

$Dijkstra(G,s)$

$Initialize(G,s)$

$T=\varnothing$

$R=G$

While $R\neq\varnothing$

$NewP=\min(R)$

$T=T\cup NewP$

$R=R-NewP$

For each v adjacent to $NewP$(Search in row of $NewP$)

$RELAX\,(NewP,v,\omega)$

The theory is for every loop to select the nearest point to starting point and renew the distances to starting points so that we can select new point and its nearest distance in next round, and finish with all points found in nearest distances.

Step 3: Realization in Matlab programming

For the input of graphic matrix, we program in Matlab to solve the problem:

```
a=zeros(10);% create graphic matrix
%input edge lengths into matrix
a(1,2)=5;a(1,5)=2;a(1,8)=8;a(1,6)=10;
a(2,3)=1;
a(3,4)=4;a(3,5)=2;a(3,6)=1;
a(4,6)=3;a(4,7)=10;
a(5,2)=3;a(5,6)=7;a(5,9)=3;
a(6,7)=8;a(6,9)=5;a(6,10)=1;
a(8,5)=1;
a(9,8)=2;a(9,10)=4;
a(10,7)=6;
a(a==0)=inf; %set all other edges to be inf(no route)
a(1,1)=0;a(2,2)=0;a(3,3)=0;a(4,4)=0;a(5,5)=0;a(6,6)=0;a(7,7)=0;a(8,8)=0;a(9,9)=0;a(10
,10)=0;%reach itself is zero distance
pb(1:length(a))=0;pb(1)=1;  %record the state whether the point is regarded min(NewP)
and set into T
index1=1; %order the vertices are put into T
index2=ones(1,length(a)); %the array to store predecessors.
d(1:length(a))=inf;d(1)=0;  %store the distances
temp=1;  %indicates the vertex from which the routes are searched and relaxed
while sum(pb)<length(a)  %while not all points are included
    tb=find(pb==0); %from starting point find all unincluded points that are possibly
```

```
reached by edges
    d(tb)=min(d(tb),d(temp)+a(temp,tb));%relax the edges to find min edge.
    tmpb=find(d(tb)==min(d(tb)));  %find the shortest edge and nearest point to reach
    temp=tb(tmpb(1));%take one of the shortest points
    pb(temp)=1;%mark the point as included.
    index1=[index1,temp];  %renew the order.
    temp2=find(d(index1)==d(temp)-a(temp,index1));%find the predecessor
    index2(temp)=index1(temp2(1)); %record the predecessor
end
```

With the algorithm above, the answer is get:

d = %the distance to each point (from 1 to 10)

0 5 6 10 2 7 14 7 5 8

index1 = %order the algorithm searches in

1 5 2 9 3 6 8 10 4 7

index2 = %the predecessor of each point

1 1 2 3 1 3 10 9 5 6