Vg101: Introduction to Computer and Programming

Spring 2021

Haoxiang WANG

MATLAB Task 1: Dijkstra’s Algorithm

1. Background

Dijkstra's algorithm, conceived by Dutch computer scientist Edsger Dijkstra in 1959, is a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree. This algorithm is often used in routing.

Algorithm:

Let's call the node we are starting with an initial node. Let a distance of a node Y be the distance from the initial node to it. Dijkstra's algorithm will assign some initial distance values and will try to improve them step-by-step.

a) Assign to every node a distance value. Set it to zero for our initial node and to infinity for all other nodes.
b) Mark all nodes as unvisited. Set initial node as current.
c) For current node, consider all its unvisited neighbours and calculate their distance (from the initial node). If this distance is less than the previously recorded distance (infinity in the beginning, zero for the initial node), overwrite the distance.
d) When we are done considering all neighbours of the current node, mark it as visited. A visited node will not be checked ever again; its distance recorded now is final and minimal.
e) Set the unvisited node with the smallest distance (from the initial node) as the next "current node" and continue from step (c).

2. Problem to be solved

Problem:

For a given graph (shown as Fig.1), find out the shortest paths and distances from a given node (node A in Fig. 1) to all the other nodes.

a) Input: an undirected connected graph (shown as Fig. 1), with weights (costs) associated with connections.
b) Algorithm: Dijikastra’s algorithm.
c) Output: the shortest paths and corresponding distances from a given node

(node A in Fig. 1) to all the other nodes.

Hints:

a) Using matrix for representing the graph with connections and weights. (How?)
b) By applying the algorithm once, you can obtain all the shortest paths and distance. (Why?)
c) There might exist more than one shortest paths between two nodes. If so, you just need to output one of them.

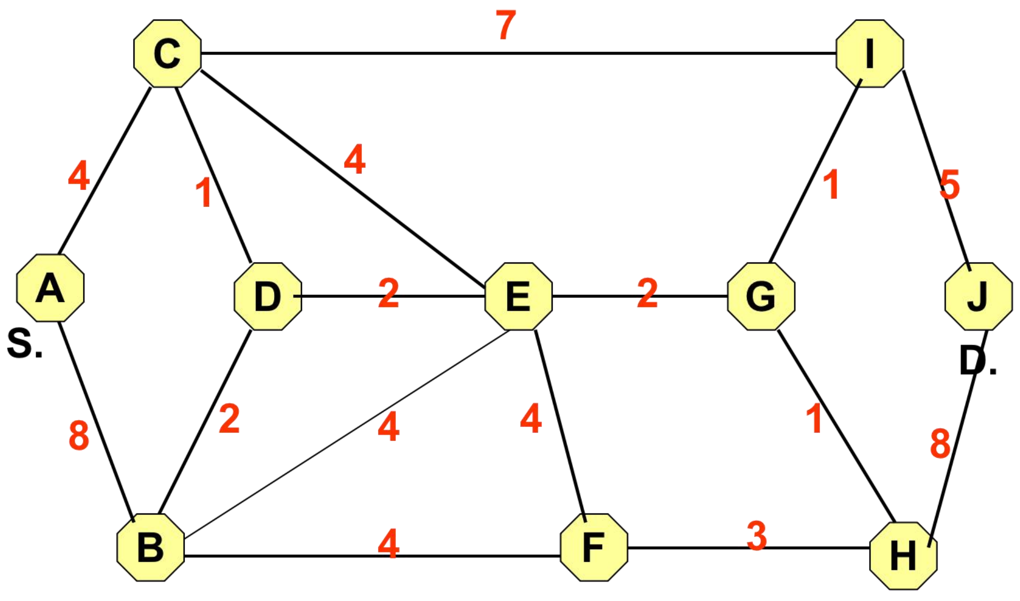


Fig. 1 An undirected connected graph with weights on connections.

3. Assessment

a) On-site demo and explanation will be required; (20)
b) Code should be well commented and correctly named;(20)
c) Functionalities implemented;(60)

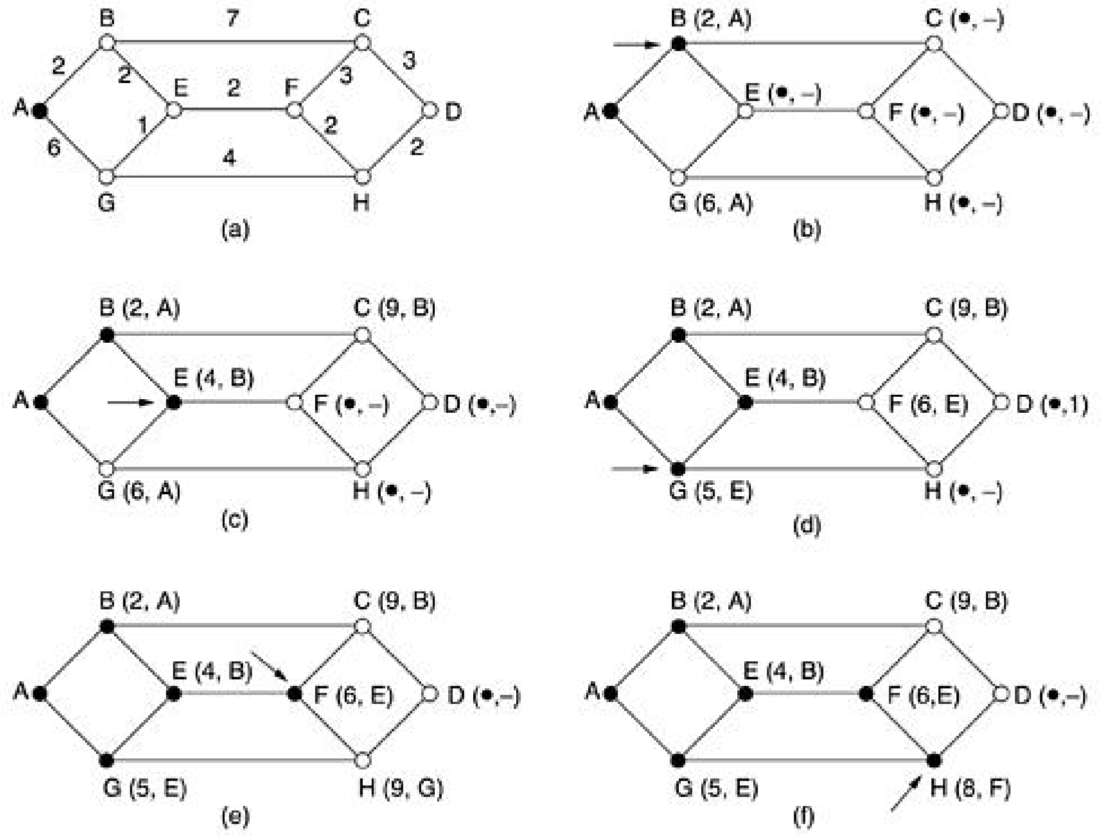
4. Submission

Demo: on the first week of our lab session. (Friday afternoon, Week 9)
Code: should be submitted before the demo. (deadline)
Naming conventions:
[studentID]_[name].m or [studentID]_[name].zip (if you want to hand in multiple files)
e.g. 202012345_张三丰.m or 202012345_张三丰.zip

Expecting to receive an All-In-One zip file for each class.

An example to help you understand the algorithm

a) Assign to every node a distance value. Set it to zero for our initial node and to infinity for all other nodes.
b) Mark all nodes as unvisited. Set initial node as current.
c) For current node, consider all its unvisited neighbours and calculate their distance (from the initial node). If this distance is less than the previously recorded distance (infinity in the beginning, zero for the initial node), overwrite the distance.
d) When we are done considering all neighbours of the current node, mark it as visited. A visited node will not be checked ever again; its distance recorded now is final and minimal.
e) Set the unvisited node with the smallest distance (from the initial node) as the next "current node" and continue from step (c).



A B C D E F G H
2 7 3 2 3 2 2 1 6 4 2
(a)
B (2, A) C (•, –) E (•, –) F (•, –) D (•, –) G (6, A) H (•, –)
(b)
B (2, A) C (9, B) E (4, B) F (•, –) D (•,–) G (6, A) H (•, –)
(c)
B (2, A) C (9, B) E (4, B) F (6, E) D (•,1) G (5, E) H (•, –)
(d)
B (2, A) C (9, B) E (4, B) F (6, E) D (•,–) G (5, E) H (9, G)
(e)
B (2, A) C (9, B) E (4, B) F (6,E) D (•,–) G (5, E) H (8, F)
(f)

From A to D
Distance: 10
Path: A-B-E-F-H-D