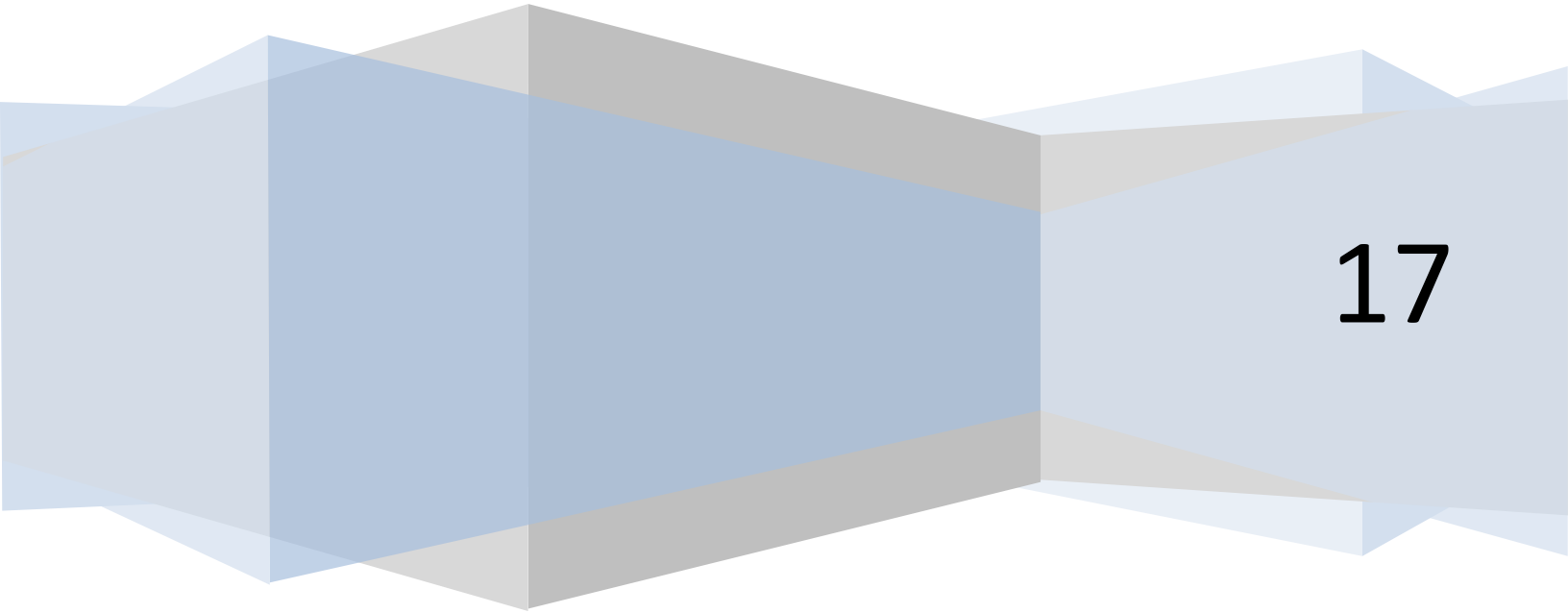
Communication Network 1 Project

Prim's algorithm visualization

Student name: Thanh Long Pham

Student number: 2764687



17

Contents

I. Prim's algorithm

Related to graph algorithms, Prim's algorithm in finding minimum spanning tree operates, somehow, like Dijkstra's algorithm in finding the shortest paths from a vertex to all vertices. They are both greedy algorithms. Starting from an arbitrary vertex, the tree which is assumed naming T grows continuously until all the vertices of graph are added to the tree. For each iteration, a light edge will be added to the tree T. Light edge is an edge connected a vertex of the tree T to one in the remained vertices of the graph which has the smallest weight. At the end of algorithm, all the edges in tree T form the minimum spanning tree. Prim's algorithm is a greedy algorithm because for each step, Prim's algorithm adds to the tree T an edge which has the minimum amount of weight.

```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

Figure 1: Pseudo code for Prim's algorithm [1]

In the pseudo code above, a graph G containing its' all edges' weights and an arbitrary vertex as a root are the input for Prim's algorithm. The initialization part is the first four lines. From the first line to the third line, the key of all vertices are set to infinite and their parents are null. At the forth line, the key of the root is set to 0. Then a min-priority queue Q is declared containing all vertices of graph G.

The **while** loop starting from the sixth line finishes if queue Q is empty. For each iteration of the **while** loop, a vertex u is extracted from the min-priority queue Q basing on the *key* value. Therefore, all vertices in the set G.V – Q are placed into the minimum spanning tree while all vertices in the set Q are the remained vertices waiting for placing to the tree. The **for** loop from eighth line to eleventh line is for

updating information of vertices of queue Q. For each vertex v which is the adjacent with the just extracted vertex u and hasn't added to tree yet is updated information if its key is greater than the weight connected it with vertex u. This updating of **for** loop is for making sure that, in the next iteration of the **while** loop, a vertex containing the light edge is added to the tree.

II. Prim's algorithm visualization application

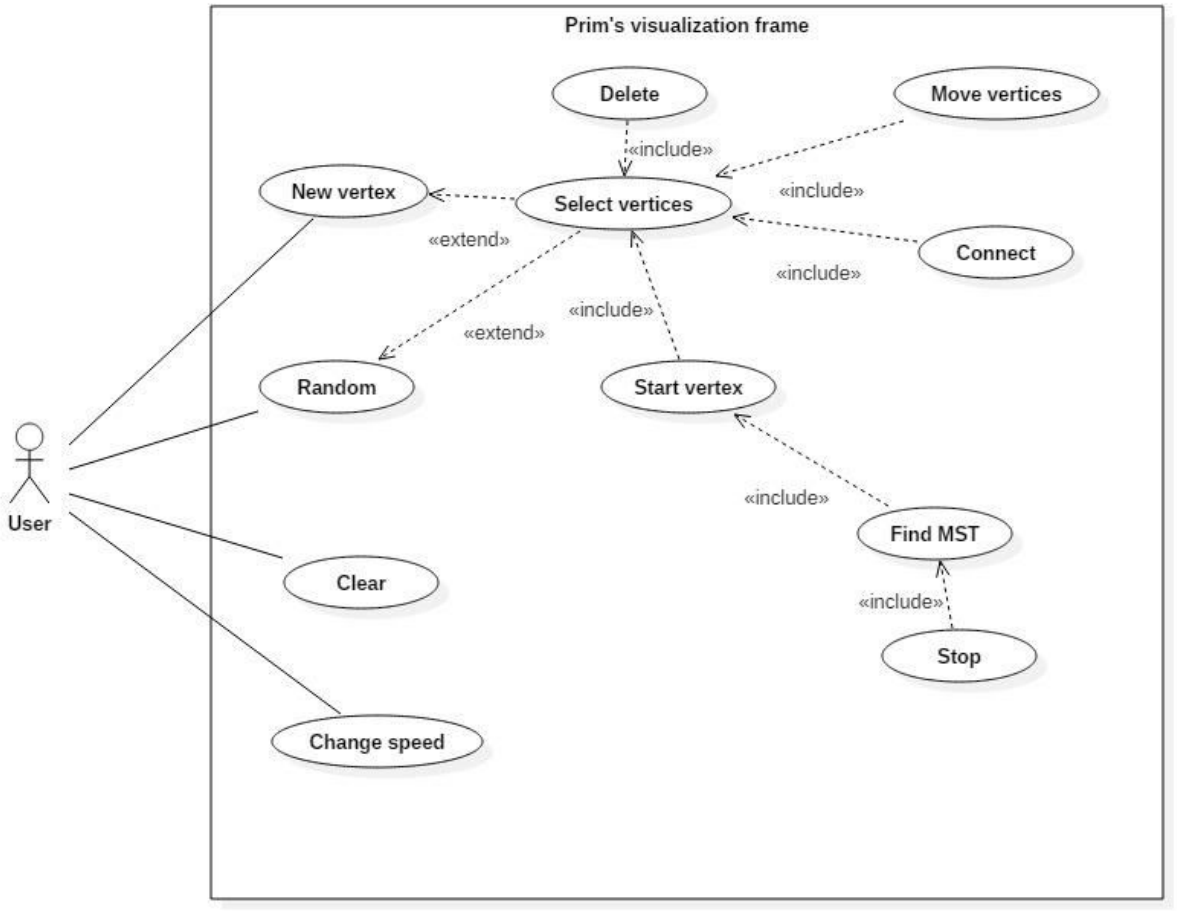
1. Overview

This application is written for visualizing Prim's algorithm. Mostly the code for drawing vertices and edges is written based on GraphPanel [2] which is an example for drawing object in Java. This application supports user to visualize a process of finding a minimum spanning tree for an undirected weight graph starting from a selected vertex. It also supports user to change the speed of visualization process for better observing how Prim's algorithm is executed.

2. List of use cases

- *New vertex*: A new vertex is created at the mouse pointer.
- *Clear*: All vertices and edges in the frame are cleared.
- *Random*: 8 vertices and 14 edges between vertices with random weights ranging from 1 to 20 are created at random locations in the frame.
- *Start vertex*: A vertex is chosen as the start vertex for executing algorithm.
- *Change speed*: Speed of visualization is changed from 1s to 10s.
- *Find MST*: Algorithm is started visualization.
- *Stop*: visualization process is stopped.
- *Select vertices*: User can select vertex by clicking on the vertex or multiple vertices by holding Ctrl button.
- *Delete*: A selected vertex and all connected edges are deleted.
- *Move vertices*: Selected vertices and connected edges are moved by mouse dragging.
- *Connect*: A weighted edge between 2 selected vertices is created.

3. Use case diagram



4. Use case scenario

a. New vertex

Name	New vertex
Actor	User
Trigger	User clicks on New vertex button on top of Frame or right click on Frame and then choose New vertex option.
Preconditions	
Post conditions	A new vertex with its name is drawn at mouse location.
Normal flow	1. User clicks on New vertex button on top of Frame or right click on Frame and then choose New vertex option. 2. System displays a new vertex with its name at the mouse location
Alternative flows	
Exception	

b. Clear

Name	Clear
Actor	User
Trigger	User clicks on Clear button on top of the Frame.
Preconditions	
Post conditions	All drawn vertices and edges are cleared.
Normal flow	1. User clicks on Clear button on top of the Frame. 2. All drawn vertices and edges are cleared.
Alternative flows	
Exception	

c. Random

Name	Random
Actor	User
Trigger	User clicks on Random button on top of the Frame.
Preconditions	
Post conditions	8 vertices and 14 edges with random weights ranging from 1 to 20 are drawn.
Normal flow	1. User clicks on Random button. 2. System generates 8 vertices with ID from 1 to 8 at 8 random locations. 3. System chooses a random vertex as the start vertex. 4. System generates 14 edges which connects 8 generated vertices with random weight ranging from 1 to 20.
Alternative flows	
Exception	

d. Start vertex

Name	Start vertex
Actor	User
Trigger	User clicks on start vertex button at the top of the Frame or right click and then chooses start vertex option from pop up.

Preconditions	A vertex is chosen already.
Post conditions	The selected vertex's color is changed from black to green.
Normal flow	1. User clicks on start vertex button at the top of the Frame or right click and then chooses start vertex option from pop up. 2. The selected vertex's color is changed from black to green.
Alternative flows	
Exception	2a. In the step 2 of normal flow, if there is no vertex which is selected. 1. System will displayed a failed message requiring user to choose a vertex. 2b. In the step 2 of normal flow, if there is more than one vertices are selected. 1. System will displayed a failed message requiring user to choose just a vertex.

e. Change speed

Name	Change speed
Actor	User
Trigger	User clicks on small arrow buttons at the top of Frame or changes the number inside the box then hits Enter.
Preconditions	
Post conditions	A number in speed field is changed.
Normal flow	1. User clicks on small arrow buttons at the top of Frame or changes the number inside the box then hits Enter. 2. A number in speed field is changed.
Alternative flows	
Exception	

f. Find MST

Name	Find MST
Actor	User
Trigger	User clicks on Find MST button at the top of Frame
Preconditions	A vertex is chosen as start vertex already.
Post conditions	A successful dialog is displayed.

Normal flow	1. User clicks on Find MST button at the top of Frame 2. Disable showing popup, others function, enabled Stop function. 3. The just extracted vertex's color is changed to green. 4. The colors of the remained vertices of Graph and edges connecting them to the tree are changed to blue. 5. The light edge and the vertex which is connected with the tree by the light edge is changed to red. 6. All the edges connected vertices which are already in MST are changed to black. 7. After finding MST, a successful popup is displayed.
Alternative flows	
Exception	2a. In the step 2a of normal flow, if user hasn't choose a vertex as start vertex yet. 1. A message is displayed asking for user to choose a vertex as a start vertex.

g. Stop

Name	Stop
Actor	User
Trigger	User click on Stop button on the top of Frame
Preconditions	Stop button is enabled when user clicks on find MST button for running algorithm.
Post conditions	Reset all colors of edges and graphs to black as default and color of start vertex as green.
Normal flow	1. User clicks on Stop button on the top of Frame. 2. All colors of edges and graphs is changed to blacks and the start vertex's color is changed to green.
Alternative flows	
Exception	

h. Select vertices

Name	Select vertices
Actor	User
Trigger	User clicks on a vertex or holds ctrl button then clicks on a vertex
Preconditions	
Post conditions	A black square border surrounding the vertex is appeared.
Normal flow	1. User holds Ctrl button. 2. User clicks to the vertex. 3. A black square border is appeared surrounding each later selected vertex.
Alternative flows	1. User clicks to the vertex. 2. A black square border surrounding vertex is appeared.
Exception	

i. Delete

Name	Delete
Actor	User
Trigger	User clicks the right button of the mouse and then chooses the Delete option.
Preconditions	A vertex or multiple vertices are chosen.
Post conditions	All selected vertices and associated edges with them are deleted from frames.
Normal flow	1. User clicks the right button of the mouse and then chooses the Delete option. 2. All selected vertices and associated edges with them are deleted from frames.
Alternative flows	
Exception	2a. In step 2 of normal flow, if there is no vertex selected. 1. Nothing happen.

j. Move vertices

Name	Move vertices
Actor	User
Trigger	User drags the mouse from a location to another location.

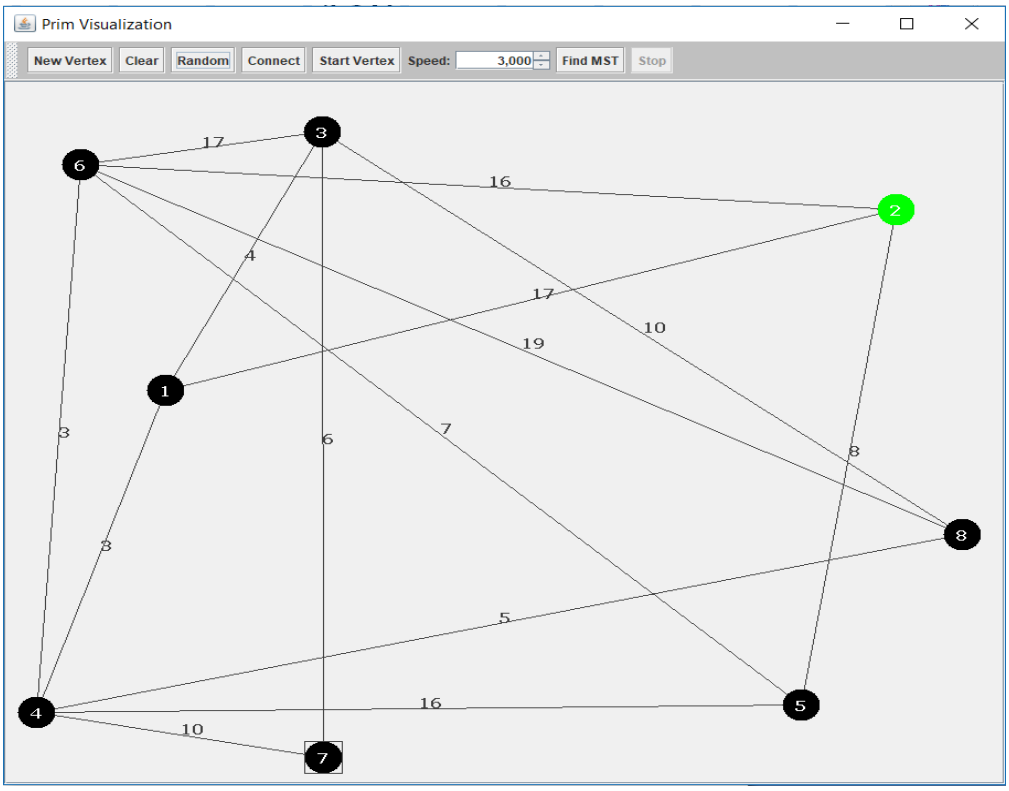
Preconditions	A vertex or vertices are selected.
Post conditions	All vertices and associated edges are moved to the new locations.
Normal flow	1. User drags the mouse from a location to another location. 2. All vertices and associated edges are moved to the new locations.
Alternative flows	
Exception	2a. In step 2 of normal flow, if there is no vertex selected. 1. Nothing happen.

k. Connect

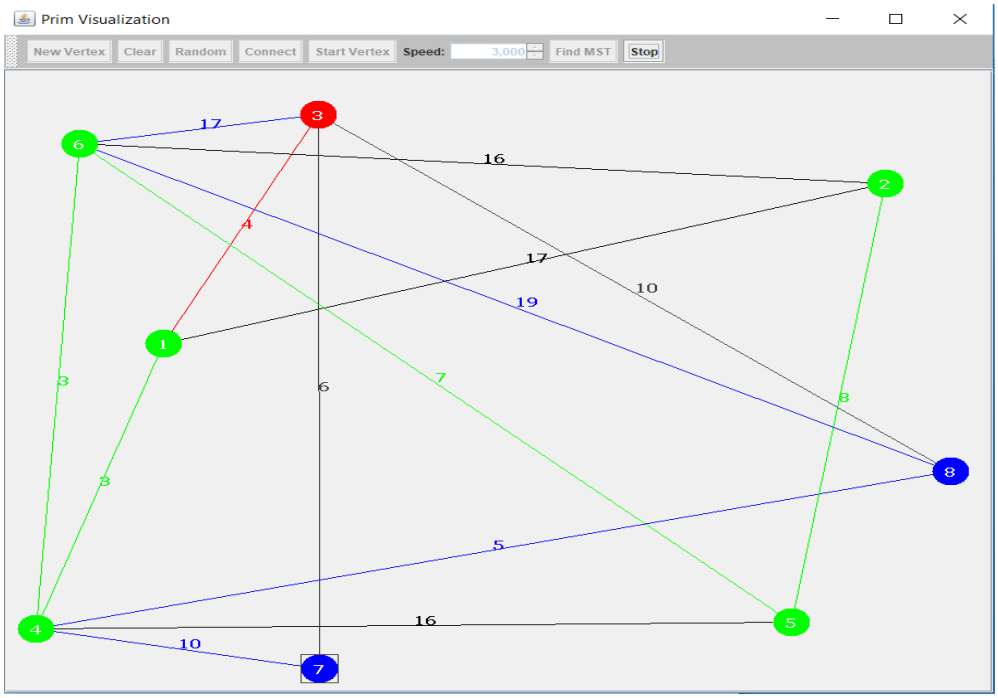
Name	Connect
Actor	User
Trigger	User selects the right button of the mouse and then choose the connect option.
Preconditions	2 vertices are selected.
Post conditions	An edge with weight connecting 2 selected vertices is drawn.
Normal flow	1. User selects the right button of the mouse and then choose the connect option. 2. A popup is occurred requiring user to enter the weight of the edge. 3. An edge with weight connecting 2 selected vertices is drawn.
Alternative flows	
Exception	2a. In the step 2 of normal flow, if there are not exactly 2 vertices selected. 1. Nothing happen. 3a. In the step 3 of normal flow, if user enters a not numeric character. 1. A message is displayed requiring users to enter just numeric characters.

5. Some screenshots

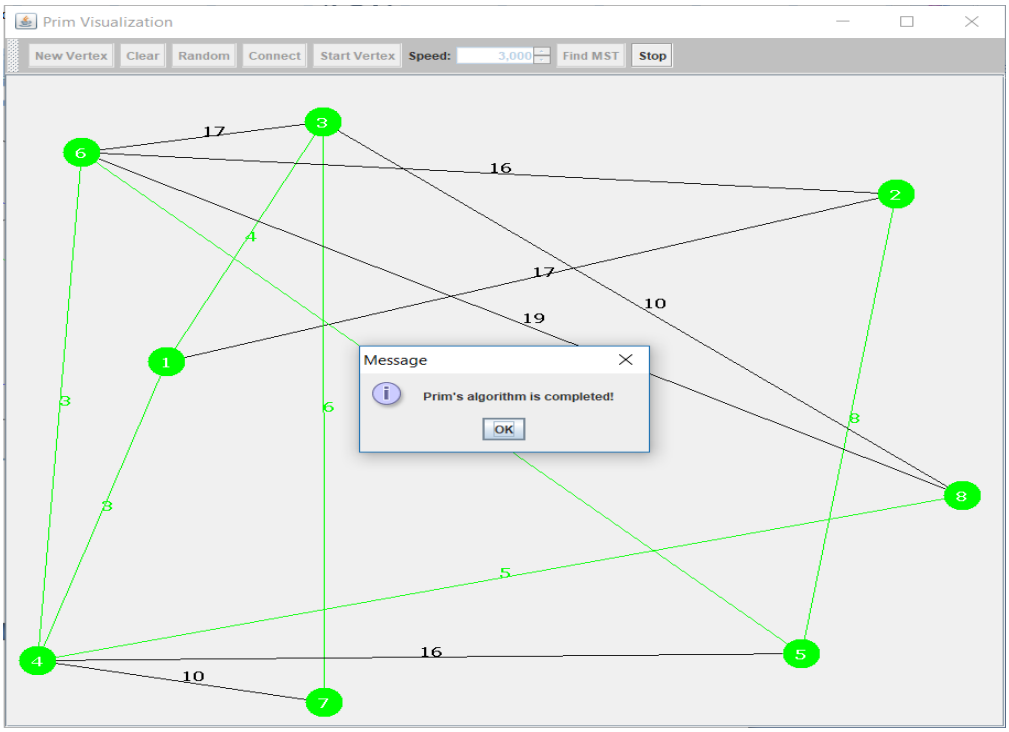
a. After clicking random button.



b. When running algorithm



c. When algorithm is completed.



III. References

[1] THOMAS H. CORMEN, CHARLES E. LEISERON, RONALD L. RIVEST, CLIFFORD STEIN, *Introductions to ALGORITHMS*, 3rd ed., 2009, pp. 634

[2] JOHN B. MATTHEWS, *GraphPanel*, https://sites.google.com/site/drjohnbmatthews/graphpanel