CS472 Module 6 Part A - Minimal Spanning Tree Algorithms

Athens State University

February 28, 2016

Outline

Contents

1 Motivation (just one use)

Motivation: Local Area Networks

A *bridge loop* in a local area network occurs when there is more than one link layer (OSI Layer 2) path between two endpoints

- Caused by multiple connections between two network switches or two ports on the same switch connected to each other
- Results in the network suffering from broadcast storms as the switches repeatedly rebroadcast any broadcast or multicast messages out every port
- Results in zombie frames
- Solution is to allow physical loops but detect and avoid the loops

Motivation: Local Area Networks

- Detection of bridge loops is implmented through the *spanning tree protocol*
- This protocol treats the network as a graph whose nodes are bridges and LAN segments and whose edges are the interfaces connecting the bridges to the segments
- The protocol builds a *spanning tree* of this graph

2 Spanning Trees

Spanning Tree: Definition

- *Spanning Tree*: The *spanning tree* of a connected, undirected graph G is a tree that includes all of the vertices and some or all of the edges of G s.t. the set of edges is the minimal set of edges that connect all vertices of G
- The spanning tree is the maximal set of edges of G that contains no cycle
- *Fundamental cycles*: Adding one edge to a spanning tree creates a cycle, such a cycle is defined as a *fundamental cycle* of the graph
  - Each edge will have a distinct fundamental cycle
- A single spanning tree of a graph can be found in linear time by either keeping track of the path taken by a depth-first or breadth-first search of the graph.

Minimal Spanning Tree: Definition and Properties

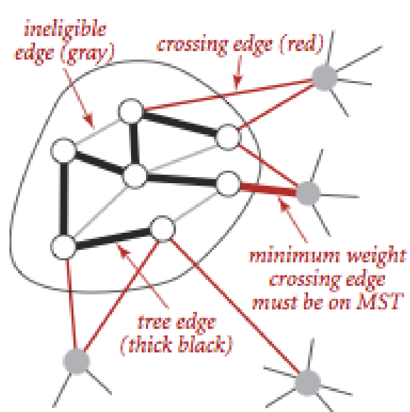
- *Minimal Spanning Tree*: for a weighted graph, the minimal spanning tree is a spanning tree with weight less than or equal to the weight of every other spanning tree of the graph
- There may be several minimum spanning trees of the same weight having a minimum number of edges
- If each edge has a distinct weight, then there will be only one, unique minimum spanning tree
- For any cycle C in a weighted graph G, if the weight of an edge e of C is larger than the weights of all other edges of C, then this edge cannot belong to the MST of G

3 Prim's Algorithm

Prim's Algorithm

- Pick a vextex in the graph as the root of the spanning tree
- Add $|V| - 1$ edges, always taking next the mimimum weight edge that connects a vertex on the tree to a vertex not yet on the tree
- Repeat until all vertices are in the tree

Prim's Algorithm

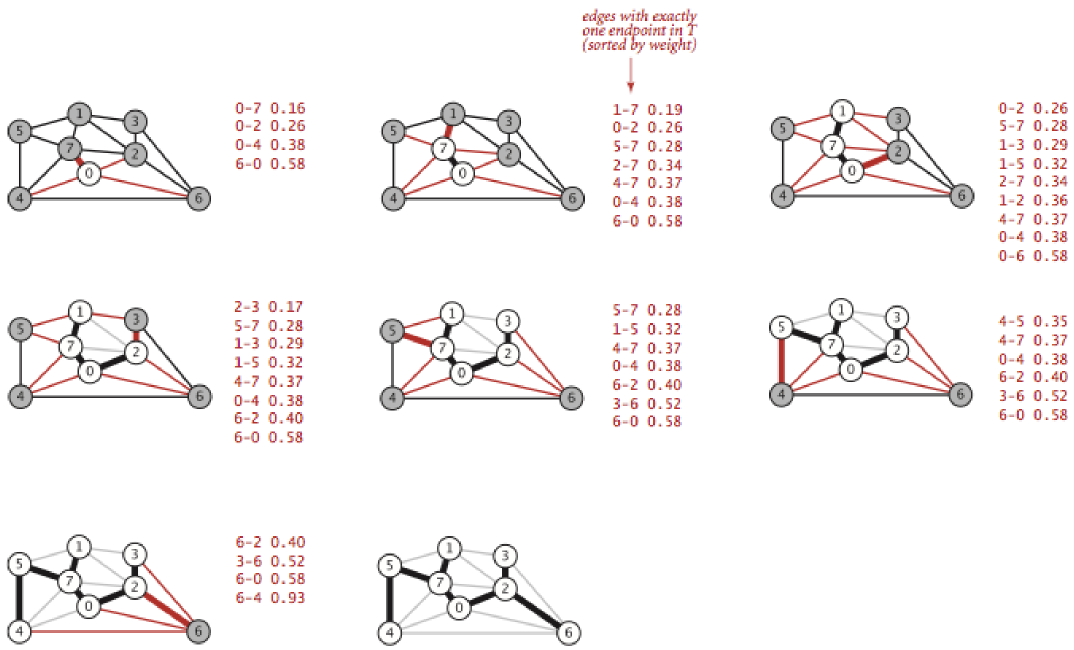


Prim's MST algorithm

Prim's Algorithm: finding that minimum edge

- Lazy implementation:
  - Use a heap to hold the crossing edges and find one of min weight
  - Each time add an edge to the tree, also add a vertex
    - and add all edges to the heaps from that vertex to any non-tree vertex
    - mark as *ineligible* any edge connecting the vertex just added to a tree vertex that is already on the priority queue

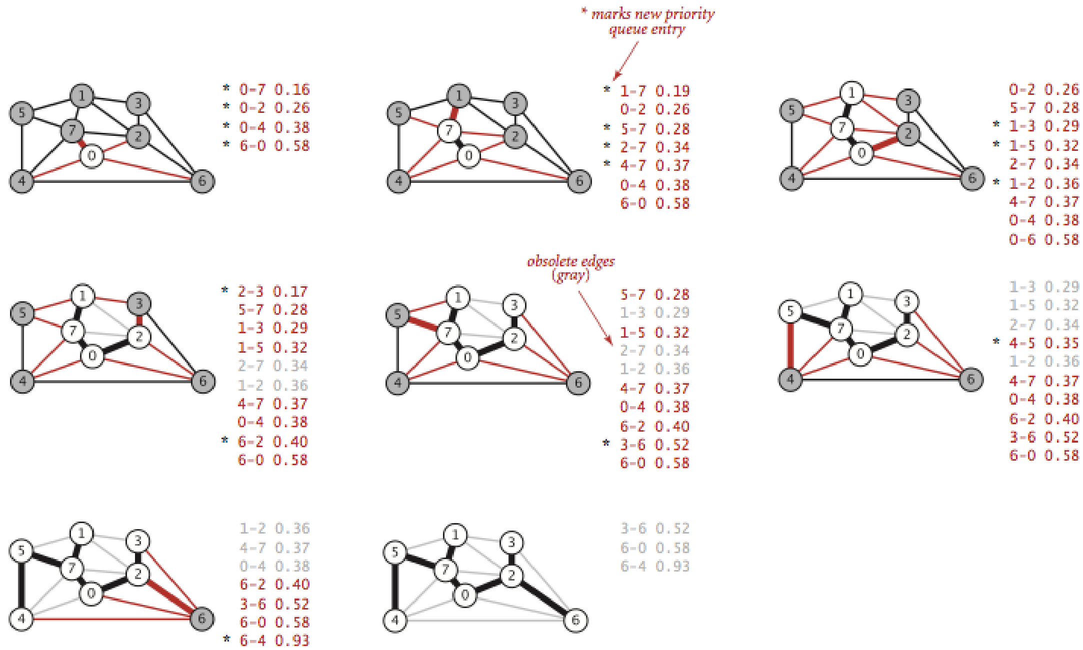
Prim's Algoirthm: finding that mimimum edge



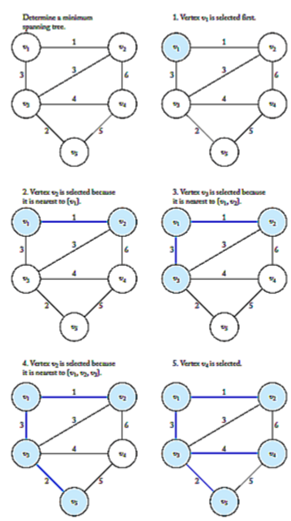
Prim's Algorithm: finding that minimum edge

- Eager implementation
  - Again, use a heap
  - Perhaps we might try to delete ineligible edges from the priority queue
  - But can do much better
    - Only interest is in the mimimal edge
    - So only need to keep track of that min-weight edge and check to see if addition of a vertex requires that we update the min-weight
    - So we only keep the shortest edge on the heap for each non-tree vertex that connects the vertex to the tree

Prim's Algorithm: finding that minimum edge



Prim's Algoirhtm: Example and Analysis



- The lazy version of the algorithm has space efficiency of $O(|E|)$ and time efficiency of $O(|E| * log|E|)$
- The eager version has space efficiency of $O(|V|)$ and time effiency of $O(|E| * log|V|)$

4 Kruskal's Algorithm

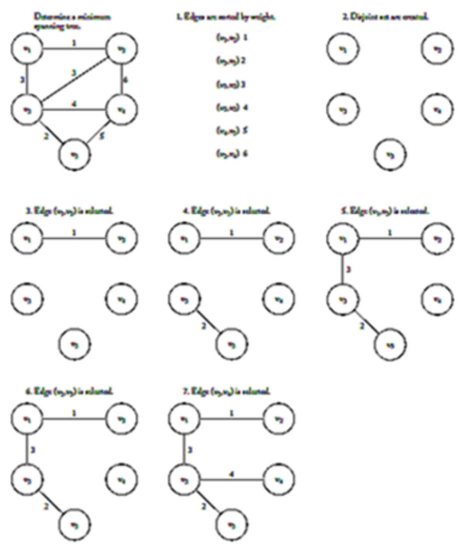
Kruskal's Algorithm

- Algorithm

- Sort the edges of the graph into a set S in increasing order of weight of each edge
- Build a forest where each vertex in the graph is a separate tree
- While S is not empty and F is not yet a spanning tree
  * Remove an edge with minimum weight from S
  * If that edge connects two different trees, add it to the forest and combine the two trees into a single tree

- Analysis: $O(|E| * log|E|)$

Kruskal's Algorithm: Example



5 A variation: shortest paths in a graph

A variation: shortest path in a graph

- *Single source shortest paths problem*: Given a weighted connected graph G, find shortest paths from a source vertex s to each of the other vertices in the graph.
- Common problem when trying to figure out how to route packets in a network
- *Shortest-path tree*: For a connected undirected graph G, a shortest-path tree from vertex v is a spanning tree T of G, s.t. the path distance from root v to any other vertex u in T is the shortest path distance from v to u in G
- Note the difference between shortest-path tree and minimal spanning trees

Dijkstra's Algorithm

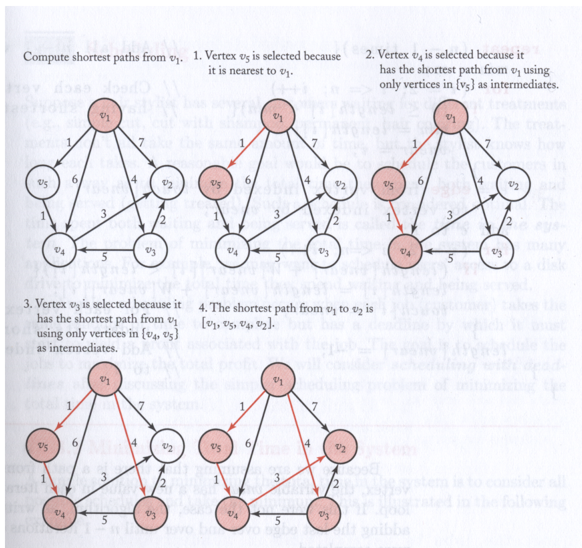
We start at some *initial node* I and define $D(Y)$ be the distance from the initial node to Y

- Assign every node a tenative distance value
  - Set the $D(I) = 0$
  - All other nodes set $D(n) = \infty$
  - Keep a list of a all unvisited nodes. Put all nodes except I on this list. Set the current node to the initial node

Dijkstra's Algorithm

- For the current node, compute the distance of all unvisited neighbors of this node
  - Compare the computed distance to that nodes currently assigned value and save the smaller of the two results
  - Once all unvisited neighbors have been considered, mark the current node as visited and remove it from the unvisited list
    * If the destination node has been marked as visited or the smallest tenative distance in the unvisited set is infinity, then stop
  - Select the unvisted node node with the smallest tenative distance and set it as the current node. Repeat.

Dijkstra's Algorithm: Example



Dijkstra's Algorithm: Notes

- Doesn't work for graphs with negative weights
- Works for both undirected and directed graphs
- Analysis
  - $O(|V|^2)$ if using a adj. matrix with weights and an array implmentation of a heap
  - $O(|E| * log|V|)$ if using adj. lists and a min-heap implementation of a priority queue
- Don't confuse Dijkstra's algorithm with Prim's Algorithm

6 Key Points

Key Points

- Definitions

  - Spanning tree
  - Minimal Spanning Tree
  - Shortest-Path Tree

- Algorithms
  - Prim's Algorithm for Minimal Spanning Tree
  - Kruskal's Algorithm for Minimal Spanning Tree
  - Dijkstra's Algoirthm for Single-Source Shortest Path