Software Science Seminar

Week 8 - Graph Algorithms

Claus Aranha
caranha@cs.tsukuba.ac.jp

College of Information Sciences

2015-06-15

Last updated June 15, 2015

Outline

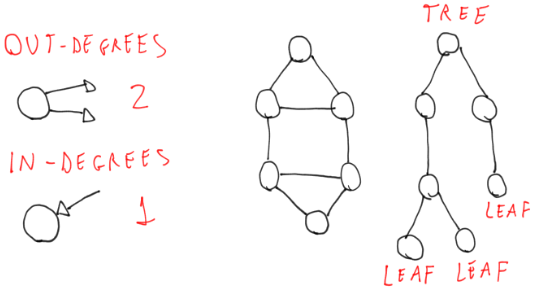
- *Part I:* Graph properties (node properties, cicles, etc)
- *Part II:* Graph algorithms (spanning tree, flow)

Part I: Graph Properties

Degree Properties

Vertex Degrees

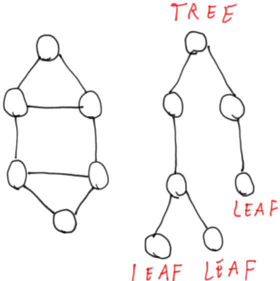
- The degree of a vertex is the total number of edges connected to it.
- For *Undirected Graphs*: Total Degrees = 2*total edges. Why?
- For *Directed Graphs*: We count in-degrees and out-degrees; Total in-degrees = total out-degrees;



Degree Properties

Trees

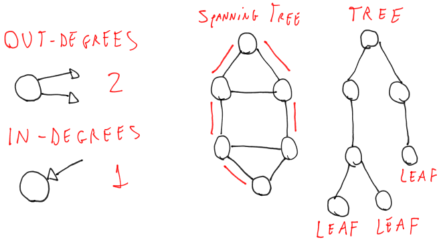
- Trees are undirected graphs with no cycles.
- Leaf nodes are nodes with degree 1;
- Every n-vertex tree contains 1-n edges.
- Rooted Trees are directed graphs where every node except the root has in-degree 1. Leaves have out-degree 0.



Degree Properties

Spanning Trees

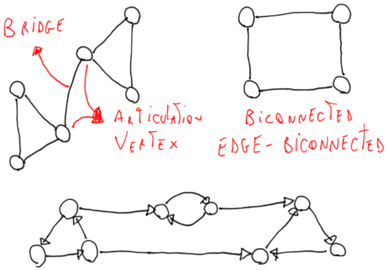
- The spanning tree of graph G(V,E) is the subset G(V,E') that forms a tree in G.
- Any connected graph has a spanning tree.
- The Minimum Spanning Tree is a important characteristic of weighted graphs.



Connectivity

Definitions of Connectivity

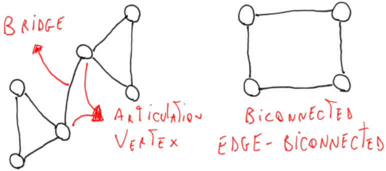
- Connected Graph: There is a path between any pair of vertices;
- The existence of a spanning trees guarantees connectivity;

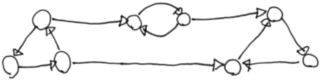


Connectivity

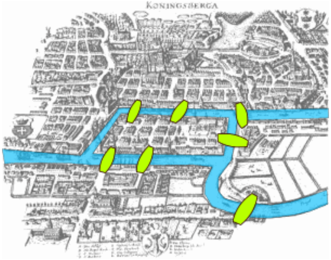
Definitions of Connectivity

- Vertex Connectivity: number of vertices that need to be deleted to "disconnect" the graph;
- If Vertex connectivity is 1, we have an articulation vertex
- If there are no articulation vertices, the graph is biconnected
- Finding by brute force: remove each vertex, and test for connectivity;





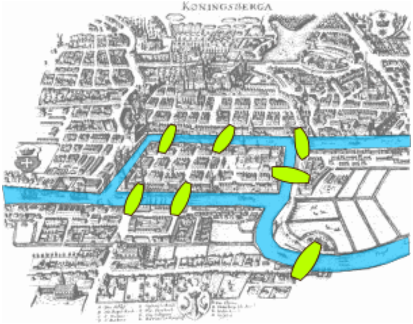
Cycles



Eulerian Cycle

- Each edge of the graph is visited exactly once.
- Connected, undirected graphs are eulerian if every vertex has even degree. Why?
- Directed graphs are Eulerian if in-degree = out-degree for all vertices;
- How can we find an Eulerian cycle?

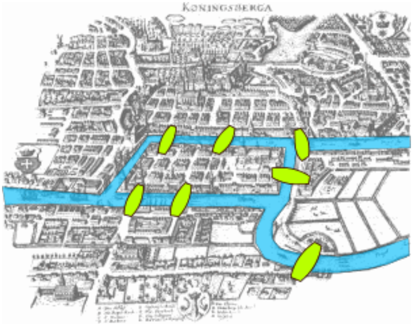
Cycles



Hamiltonian Cycles

- Each vertex of the graph is visited exactly once;
- Not so easy to find a solution as the Eulerian cycle;

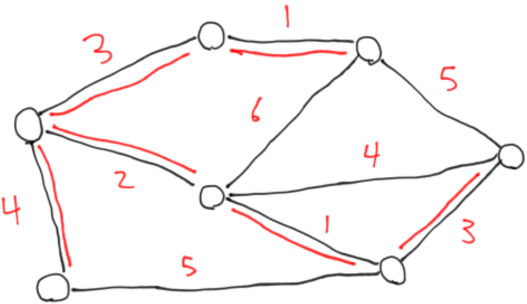
Cycles



Part II: Graph Algorithms

Minimum Spanning Trees

- Spanning Tree: A subset tree of a graph that includes all nodes.
- Minimum Spanning Tree: The spanning Tree that has minimal weight.

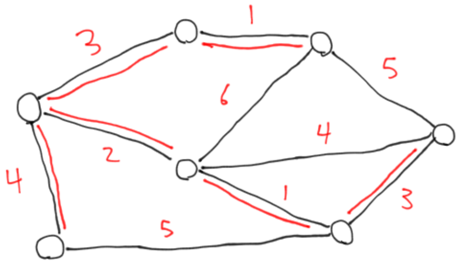


Minimum Spanning Tree

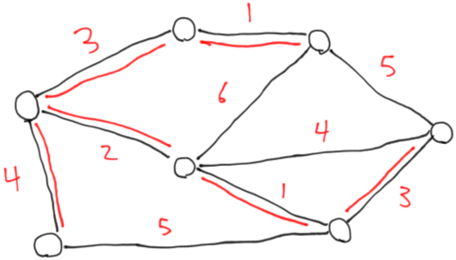
Prim's Algorithm to find the MST

Greedy algorithm:

```
1- Sort all edges by weight.
2- Add the smallest edge to MST.
3- Sort edges that are connected to the MST.
4- Add the smallest edge with one new node to MST.
5- Go to 3.
```



Minimum Spanning Tree



- Maximum Spanning Tree: -1*weight;
- Multiplication Spanning Tree: log(weight);

Articulated Vertex Problem

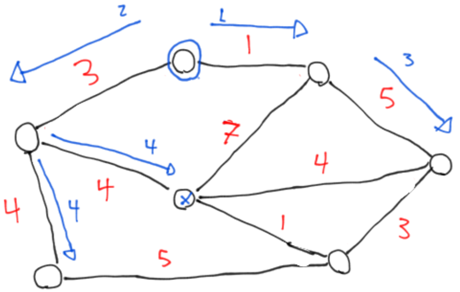
An algorithm to find an articulated vertex in a graph was discussed in the last class.

```
1- Create a spanning tree using DFS
2- Rank each node by search order.
3- Back edges are not part of the DFS

4- Give a second rank to each node, based on the
   highest rank they can reach using a back node.

5- Nodes where all children have second rank
   smaller than themselves are articulation nodes.
```

Shortest Path

Djikstra Algorithm

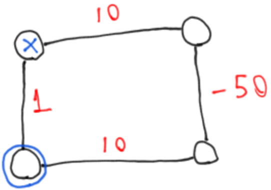
- Very similar to Prim: What is the difference?
- When does this algorithm encounter problems?



Shortest Path

Djikstra Algorithm

- Very similar to Prim: What is the difference?
- When does this algorithm encounter problems?



All-Pairs Shortest Path

Floyd's Algorithm (A bit similar to DP)

- We want to find the shortest distance from all nodes to all nodes;
- Needs an adjacency Matrix (code/idea is simple);
- Iterate over nodes usable in the path;
- usable to test for unreachable nodes (infinite distance at the end);

```
weight[][] is adjacency matrix, unconnected nodes are INF

for (k = 1 to n-vertices):
  for (i = 1 to n-vertices):
    for (j = 1 to n-vertices):
      path_k = weight[i][k]+weight[k][j]
      if path_k < weight[i][j]:
        weight[i][j] = path_k
```

Union-Find Problem

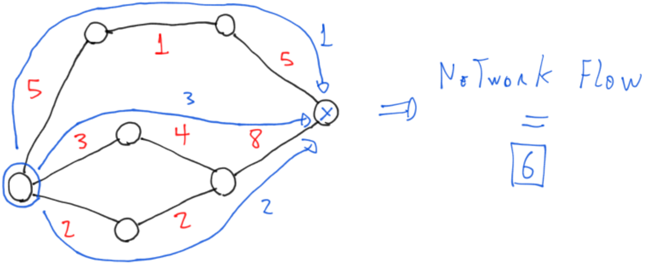
Union-find is an algorithms that creates a structure to construct a set from disjoint nodes. It has two main functions:

- Union(Set A, Set B) - Merges set A with set B
- Find(Node A) - finds what set node A belongs to.

In the beginning, all nodes belong to a set containing only themselves. Progressive application of Union will create the different sets using double linked lists.

Network Flow

- How much flow can we get from node *A* to node *B* at once?
- Ford-Fulkerson "Augmenting Path" algorithm:
- Basic idea: Do a BFS, remove "flow" from weight, repeat the BFS with new weights.



This Week's Problems

- Freckles
- Fire Station
- Tourist Guide
- War