Bartosz Antczak Instructor: Luke Postle January 27, 2017

Review of Last Lecture

We proved a theorem:

G is bipartite if and only if G has no odd cycles

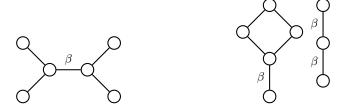
A polynomial-time algorithm for determining if G is bipartite:

- 1. Find a spanning tree T_i of each component G_i of G
- 2. Pick a root r_i for each T_i
- 3. Find $A_i = \{v \in V(T_i) : d(v, r_i) \text{ is even}\}\ \text{and } B_i = \{v \in V(T_i) : d(v, r_i) \text{ is odd}\}\$
- 4. Let $A = \bigcup_i A_i$, $B = \bigcup_i B_i$
- 5. If all edges have one end in A and other end in B: return yes and the bipartition (A, B) to verify this
- 6. Otherwise, there exists an edge e = uv with both ends in same partition: now find a path P from u to v in the tree T_i containing u and v and return no! The odd cycle C = P + e verifies that G is not bipartite

11.1 Bridges

A **bridge** of a graph G is an edge e such that G - e has more components than G.

Example 11.1.1. Various graphs with their respective bridges labelled ' β '



11.1.1 Theorem

An edge e = uv is not a bridge of graph G if and only if there is a cycle C of G that contains e (Proof is on the next page).

Proof:

edge
$$e = uv$$
 is not a bridge

 \iff

u and v are in the same component of G - e

 \iff

there is a path P in G - e from u to v

there is a cycle C of G that contains e

11.1.2 Corollary

Graph G is a forest if and only if all edges are bridges

Also similarly: when G is connected,

G is a tree if and only if all of its edges are bridges

11.2 Weighted Graphs

A weighted graph is a graph G with a weight function w on the edges of G (for the purposed of this class, we assume w is non-negative).

Example 11.2.1. A graph with a weight on each edge



Weighted graphs model

- Time taken to travel between nodes
- Distance between nodes
- Cost required to travel between nodes

Question 1

Finding the shortest paths in weighted graphs is useful, and there are very efficient algorithms that do this. Let's consider the weight of each edge in terms of cost. Can we find a cheapest cost set of edges connecting all vertices efficiently? (We can interpret this question as the edges being roads and vertices being cities, for a practical example)

Observation 1

Let's consider the graph from example 11.2.1, observe that the cheapest set of edges would include 1, 2, and 5. From this observation, we can conclude that a cheapest cost set has no cycles, because a cycle will always contain at least one extra edge not needed to connect all vertices.

Since this edge set must connect all vertices and contain no cycle, the cheapest cost set is actually a *spanning* tree. From this, we derive a definition:

11.2.1 Minimum Weight Spanning Tree (MST)

A tree T is an \mathbf{MST} of a weighted graph G if it has a minimum weight $\left(\sum_{e \in E(T)} w(e)\right)$ among all spanning trees.

How can we find an MST? We can use Prim's Algorithm.

Prim's Algorithm

- Pick a vertex v and put it in our spanning tree T (initially will only contain v)
- \bullet While T is not a spanning tree:
 - Pick an edge e in $\delta(V(T))$ of least weight
 - $\operatorname{Set} T := T + e$

Example 11.2.2. Applying Prim's Algorithm on the weighted graph from example 11.2.1

- Let's pick vertex c to be our tree T. The weight of each edge in $\delta(T)$, $\{ca, cd\}$, is respectively $\{2, 5\}$. We choose the lighter edge ac to be added to T
- Now $\delta(T) = \{ab, ad, cd\}$. We choose ab, which is the lightest
- Finally, $\delta(T) = \{ad, cd\}$, and we choose cd. Now all vertices are connected, and we have constructed an MST

11.2.2 Theorem

The graph constructed using Prim's algorithm is always an MST

Proof:

Let T_0 be the graph constructed using Prim's algorithm, and let e_1, e_2, \dots, e_{n-1} be all of the edges of T_0 in the order they are added to T. Let T be an MST of graph G such that if edge k is the smallest i such that $e_i \notin E(T)$ that k is maximized among all MST's.

Claim: $T_0 = T$ and hence k does not exist.

• Proof of Claim: suppose that k does exist. That is, $e_1, e_2, \dots, e_{k-1} \in E(T)$ but $e_k \notin E(T)$. I claim that there exists a spanning tree T' that uses e_k instead of another edge f in $\delta(V(T))$ and otherwise equals to T. But now $w(T') \leq w(T)$ since e_k had the smallest weight in $\delta(V(T))$. But now T' uses e_1, \dots, e_k and so T is not maximal, which is a contradiction. Ergo, k does not exist. Thus, $T_0 = T$, which proves our statement.