Lecture 6: trees and dijkstra's algorithm, part 1

Thursday, January 28, 2016 12:32 PM

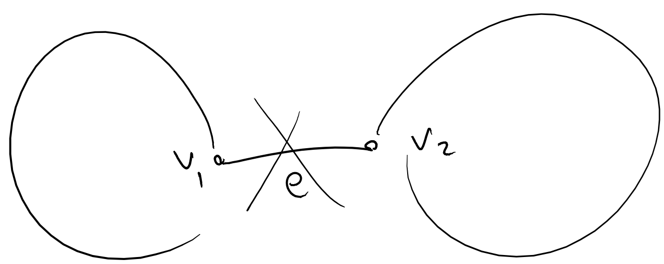
Def If G is a tree, $v \in V_G$ is called a leaf if $d_G(v) = 1$

Lem 1 If G is a tree, then $e(G) + 1 = v(G)$

Lem 2 If $e \in E_G$, then $c(G-e) = c(G)$ or $c(G)+1$

$c(G) = \#$ components of G

Pf: Assume first, $c(G) = 1$, suppose $G-e$ not connected



want $c(G-e) = 2$

let V_i = vertices equiv to v_i (connected by a walk) in $G-e$

- $G[V_i]$ connected ✓ transitivity of ~
- $G[V_1] \cup G[V_2] = G-e$
  - $V_1 \cup V_2 = V_{G-e}$?
  - $E_{G[V_1]} \cup E_{G[V_2]} = E_{G-e}$

given $v \in V_{G-e} = V_G$, why is $v \in V_1$ or V_2

choose a path in G from v to v_1

either e is in the path or not.

if not, then $v \in V_1$

if e is in the path, then

path looks like $v e_2 w_2 e_3 w_3 \cdots v_2 e v_1$

$\Rightarrow v e_2 w_2 \cdots v_2$ is a path not involving e from v to $v_2 \Rightarrow v \in V_2$

$\square$

general graph

$G = \bigcup G_i$ G_i components, say $e \in G_1$

$$G - e = \underbrace{(G_1 - e)}_{\substack{1 \text{ or } 2 \\ \text{components}}} \cup \underbrace{\bigcup_{i=2}^{c(G)} G_i}_{c(G)-1 \text{ components}}$$

$\Rightarrow c(G)$ or $c(G)+1 = c(G-e)$

$\square$

Pf of lem 1

Let G be a tree. choose $e \in E_G$ (else done) consider $G-e$.

Claim $G-e$ is not connected

if e incident to v_1, v_2 $\Rightarrow$ $v_1 e v_2$ is the unique path in G from v_1 to v_2

$\Rightarrow$ in $G-e$ $\nexists$ (v_1,v_2)-path.

$\Rightarrow$ $c(G-e)=2$, $G-e = G_1 \cup G_2$

G_i trees since connected & acyclic.

we'll prove statement by induction on $\underbrace{\#\text{edges in } G}_{e(G)}$

know true for G_1, G_2

$$
\begin{aligned}
\underbrace{v(G)}_{\#V_G} = v(G_1)+v(G_2) &= e(G_1)+1+e(G_2)+1 \\
&= e(G_1)+e(G_2)+2 \\
&= e(G-e)+2 \\
&= e(G)-1+2 = e(G)+1.
\end{aligned}
$$

$\triangleright$

Question: given a list of cities, certain roads b/t them. want to calculate between ...

shortest path betwen two cities.

math

G vertices $\longleftrightarrow$ cities

edges $\longleftrightarrow$ roads

"cost function" $w: E_G \longrightarrow \mathbb{R}_{>0}$

given a walk $W = v_1 e_2 v_2 e_3 \cdots e_n v_n$

define: $l(W) = \sum_{i=2}^{n} w(e_i)$

$d(v,w) = \min \{ l(W) \mid W \text{ a } (v,w)\text{-walk} \}$

Goal: Given G, w, $v, w \in V_G$, find W a (v,w)-walk s.t. $d(v,w) = l(W)$

Procedure: inductively produce subgraphs $T_0, T_1, T_2 \ldots < G$ which consist of vertices whose distance to v has been explicitly computed by producing paths.

either stop when we get $w \in T_i$ or keep going if we want and get minimal paths to each vertex.

$N(T_i)^0 = \{ v \in V_G \setminus V_{T_i} \mid v \text{ adjacent to some } v' \in V_{T_i} \}$

Algorithm

- Start with T_0 just consist of v & no edges
- Given T_i, choose $u \in N(T_i)^0$, $v' \in V_{T_i}$ adjacent to u via edge e such that $d(v,v') + w(e)$ minimal

(in this case $d(v,u) = d(v,v') + w(e)$)