Nathan Bickel
MATH 575: Section H01
Professor: Dr. Luo
January 31, 2023

MATH 575 Homework 1

Collaboration:

Problem 1 Prove that a forest with n vertices and c components contains exactly $n-c$ edges.

Solution.

First, let $n=1$. Then, there can only be $c=1$ connected components and 0 edges, so we have $n-c=1-1=0$.

Next, let $n \in \mathbb{N}$, $n>1$, and let $c \in \{1,2,\ldots,n\}$. Assume that for any $n' \in \mathbb{N}$, $n'<n$, a forest with n' vertices and c components contains exactly $n'-c$ edges. Let F be a forest with n vertices and c connected components. By definition, each component is a tree.

We have shown in class that every tree has at least one leaf. Thus, construct an induced subgraph G' of G by removing one leaf from each connected component. Then, $|V(G')| = n' = n-c < n$ since $c>0$, so by the induction hypothesis, $|E(G')| = n'-c$.

Now, add the leaves we removed to G'. We removed c leaves, and since each was incident to exactly one edge, we also removed c edges. Thus, $|E(G)| = |E(G')| + c = n' - c + c = n' = n - c$. Therefore, by strong induction, the claim holds for all n. □

Problem 2 Let G be an n-vertex tree with $\Delta(G) \leq 2$. Prove that G must be isomorphic to the path P_n.

Solution.

Let $Q = \{q_1, q_2, \ldots, q_k\}$ be a longest path in G, and assume G is not isomorphic to P_n (so Q cannot be the whole graph). Then, there exists a non-empty subgraph G' induced by $V(G) - Q$. So there exists some $u, v \in V(G)$, $uv \in E(G)$ such that $u \in Q$ and $v \notin Q$ (if there weren't, there would be no way to get from Q to G' and thus G wouldn't be connected).

Case 1: u is either q_1 or q_k. Then, Q cannot be a longest path, because $Q \cup v$ is longer, a contradiction.

Case 2: $u = q_i$ for $1 < i < k$. Then, u connects to q_{i-1}, q_{i+1}, and v. But then $d(u) = 3$, contradicting $\Delta(G) \leq 2$.

Therefore, G must be isomorphic to P_n. □

Problem 3 Suppose G is a graph with the property that deleting any single vertex (and its incident edges) results in a tree.
(a) What can we say about the number of edges in G and the degrees of its vertices?
(b) Use part (a) to determine G itself.

Solution.

We note that this property holds for K_1, K_2, and $\overline{K_2}$, but we will consider graphs on $n \geq 2$ vertices. We have shown in class that an edge is a cut-edge if and only if it is not contained in a cycle. Because of this, we claim that every edge in G must be contained in one cycle.

If an edge is not contained in a cycle, then it is a cut-edge. Thus, removing either of its endpoints will disconnect the subgraph, meaning it cannot be a tree and contradicting our assumption. Alternatively, suppose an edge is contained in two distinct cycles C and C'. Then, there must exist some $v \in V(G)$ such that $v \in C$ and $v \notin C'$, or C and C' would be the same cycle. But then removing v would not result in a tree, because C' is still in the subgraph and thus it is cyclic, a contradiction. Since each edge is contained in no fewer and no more than one cycle, each edge is contained in exactly one cycle.

We must have that G is connected (if it weren't, then removing a vertex from the largest component would not result in a connected subgraph). But then the only way for every vertex to be a part of exactly one cycle is if G is simply a C_n. Thus, $|E(G)| = n$ and each vertex has degree 2. □

Problem 4 Let G be a connected, weighted graph with n vertices. Let xy be an edge of largest weight in G.
(a) Prove or disprove: there must exist a MST of G that avoids the edge xy.
(b) Prove that if xy is contained in a cycle in G, then there exists a MST of G that avoids the edge xy.
(Hint: The proof of Proposition 2.1.6 in the textbook may have useful ideas. You can also try applying Kruskal's Algorithm.)

Solution.

(a) This is false. Suppose y is only neighbors with x. Then xy must be in any spanning tree so that y is covered, and thus xy is in the MST.

(b) Suppose we have used Kruskal's algorithm to construct a MST T' of G that includes the edge xy:

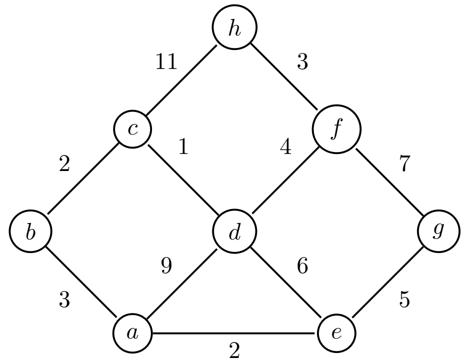
1. Let F be a forest with $V(F) = V(G)$ and $E(F) = \emptyset$.
2. Let $E' = (e_1, e_2, \ldots, e_k)$ be an ordering of $E(G)$ such that for all $i, j \in \{1, 2, \ldots, k\}$, we have that $i < j \implies \mathrm{w}(e_i) \leq \mathrm{w}(e_j)$. Since order does not matter among edges with the same weight, choose an ordering where xy comes after all the edges with the same weight. We note that since xy has largest weight, this choice will always result in $e_k = xy$.
3. Let $i = 1$. If adding e_i to F decreases the number of connected components, add e_i to F.
4. Repeat step 3 for $i = 2, 3, \ldots, k$ (in that order) until F has only one connected component.

Since xy is in a cycle, let $C \subseteq G$ be the cycle with $E(C) = \{xy, f_1, f_2, \ldots, f_m\}$. Since we chose in the algorithm to add xy, adding xy decreased the number of connected components. So before adding xy, there were two components G_1 and G_2 with $x \in G_1$, $y \in G_2$.

Because we have C, there is another xy-path in G, and there must be another edge $f \in C$ such that one endpoint is in G_1 and the other endpoint is in G_2. But since xy is the last element in E', f would have been considered in the algorithm before xy. Adding the edge f to the forest F would have decreased the number of components by connecting G_1 and G_2, so it would have been added. But then by the time xy is considered, G_1 and G_2 are already connected, a contradiction since we assumed they were separate components.

So Kruskal's algorithm will never produce such a T' on such a G. Since we proved in class that Kruskal's algorithm always generates a MST, simply apply the algorithm above to find a MST that avoids xy. □

Problem 5 Use Dijkstra's Algorithm to find a shortest path tree starting with the vertex a. At each step of the algorithm, write the edge that was added to the tree.

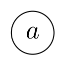


Solution.

We define $\mathrm{t}(v) : V(T) \to \mathbb{Z}$ to be the tentative distance from a to v, and we define $\mathrm{prev}(v) : V(T) \to V(T)$ to be the vertex that v connects to to get the tentative distance. At each step, we will show a table of these updated values and the tree T we are constructing. In the table, the vertices already in T will be bolded.

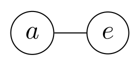
1.

	a	b	c	d	e	f	g	h
t()	0	3	∞	9	2	∞	∞	∞
prev()	-	a	-	a	a	-	-	-



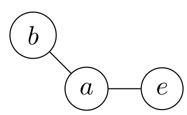
2.

	a	b	c	d	**e**	f	g	h
t()	0	3	∞	8	2	∞	7	∞
prev()	-	a	-	e	a	-	e	-



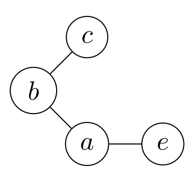
3.

	a	**b**	c	d	**e**	f	g	h
t()	0	3	5	8	2	∞	7	∞
prev()	-	a	b	e	a	-	e	-



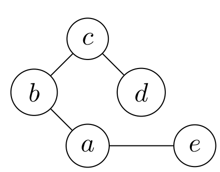
4.

	a	**b**	**c**	d	**e**	f	g	h
t()	0	3	5	6	2	∞	7	16
prev()	-	a	b	c	a	-	e	c



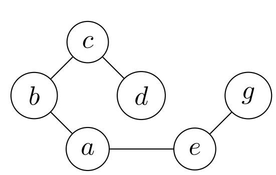
5.

	a	**b**	**c**	**d**	**e**	f	g	h
t()	0	3	5	6	2	10	7	16
prev()	-	a	b	c	a	d	e	c



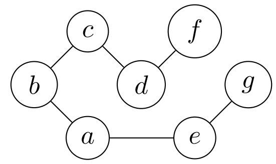
6.

	a	**b**	**c**	**d**	**e**	f	**g**	h
t()	0	3	5	6	2	10	7	16
prev()	-	a	b	c	a	d	e	c



7.

	a	**b**	**c**	**d**	**e**	**f**	**g**	h
t()	0	3	5	6	2	10	7	13
prev()	-	a	b	c	a	d	e	f



8.

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**
t()	0	3	5	6	2	10	7	13
prev()	-	a	b	c	a	d	e	f