Problem Set 2 – Solutions

Graph Theory 2016 – EPFL – Frank de Zeeuw & Claudiu Valculescu

---

1. *Prove that the following statements about a graph G are equivalent.*
   - G is a tree;
   - G is minimally connected (it is connected and removing any edge disconnects it);
   - G is maximally acyclic (it has no cycles and adding any edge creates a cycle).

   - **tree implies maximally acyclic:** Assume that G is a tree, so certainly it has no cycles. Let uv be an edge not in G. Since G is connected, there is a path between u and v, say $uv_1 \cdots v_k v$. Then adding uv to G creates the cycle $uv_1 \cdots v_k vu$.

   - **maximally acyclic implies minimally connected:** Assume G is maximally acyclic. Let $u, v \in V(G)$ with $uv \notin E(G)$. By maximality, adding uv to G creates a cycle, which implies that there must be a path in G from u to v, so G is connected. To show that it is minimally connected, remove an edge $xy \in E(G)$. If $G - xy$ were still connected, then there would be a path from x to y in $G - xy$, which together with xy would form a cycle in G. So $G - xy$ is not connected.

   - **minimally connected implies tree:** If G is minimally connected, then it is connected, so we just have to prove that it has no cycle. If G contains a cycle, removing any edge from this cycle leaves the graph connected, contradicting minimality. So G does not contain a cycle.

2. *Show that every tree T has at least $\Delta(T)$ leaves (vertices of degree 1).*

   Let v be a vertex with degree $d \geq \Delta(T)$. For every edge vw incident to v, take a longest path starting with vw. By maximality (as in the proof that every tree has a leaf), the last vertex of this path is a leaf. Doing this for each of the d edges incident to v, we get d paths starting at v, which are disjoint except for v (otherwise we would get a cycle). Thus each path gives a different leaf, and we get $d \geq \Delta(T)$ leaves.

   Alternative solution: If you remove v and its incident edges, you are left with d connected components $T_1, \ldots, T_d$, each of which is a tree. By a lemma from class, every tree with at least two vertices has at least one leaf, and the proof of the lemma actually showed that there at least two. Hence the T_i with at least two vertices have at least two leaves, one of which must be a leaf of T (one of the two leaves might have been adjacent to v, but not both because that would give a cycle). Some of the T_i might be single vertices, in which case those vertices were leaves in T (they must have been adjacent to v and to no other vertex).

3. *Let T be a tree with $|V(T)| \geq 2$. Suppose the vertices of T are colored red and blue, such that no two adjacent vertices have the same color. Show that if there are at least as many red vertices as blue vertices, then T has a red leaf.*

   The key is the following observation about any subtree $T' \subset T$ and an outgoing edge $e \in \partial(T')$. The endpoint of e outside T' is either blue, or it is red, in which case it is either a red leaf, or adjacent to a blue vertex outside T'. Thus we either find a red leaf, or we get an imbalance towards blue. This leads to the following algorithmic proof.

   Suppose there are no red leaves. We reconstruct T vertex by vertex starting from a blue vertex, keeping track of which colour class is larger. Pick a blue vertex and repeatedly do the following until we have the whole tree. If the tree that we have so far has a blue neighbor, then we add this neighbor. Otherwise, there must be a red neighbor, which by

assumption is not a leaf, so must have another neighbor, which by the coloring property is blue. Then we add these two vertices.

We start with a tree with more blue vertices than red vertices, and in every step we either add a blue vertex, or a red one and a blue one. This will terminate when we have the entire tree, and the blue class will still be strictly larger. Contradiction.

Alternative solution: There must be $\geq n/2$ red vertices. If none of them are leaves, they all have degree ≥ 2, so the sum of the degrees of the red vertices is $\geq 2 \cdot n/2 = n$. But the tree has $n-1$ edges, and each edge has only one red vertex. Contradiction.

4. *Show that a graph G contains at least $|E(G)| - |V(G)| + 1$ cycles.*

   We prove the statement by induction on $|E(G)|$. For $|E(G)| \leq |V(G)|-1$, there is nothing to prove. Let G be a graph with $|E(G)| > |V(G)| - 1$. The number of edges implies that G cannot be a tree or a forest (by lemmas from class), so it must contain a cycle C. Let uv be any edge from C. By induction, $G - uv$ contains at least $(|E(G)| - 1) - |V(G)| + 1$ cycles, not including C. Adding C gives $|E(G)| - |V(G)| + 1$ cycles.

5. *A vertex is* central *if its greatest distance from any other vertex is as small as possible. Show that a tree has either a single central vertex, or two adjacent central vertices.*

   First observe that in a tree, if u is some vertex and v is at maximal distance from u, then v must be a leaf, because otherwise there is another vertex further away from u.

   Therefore, if we remove all leaves at once, all greatest distances are reduced by 1, and the set of central vertices remains the same. We can repeatedly remove all leaves until this is no longer possible, which must be because we are left with a single degree 0 vertex, or we are left with no vertices. In the first case, the degree 0 vertex is central, and it must have been the only central vertex in each previous step, including in the original tree. In the second case, we must have had a single edge in the previous step, whose vertices must have been the central ones in the original tree.

6. *Determine the diameter of the following graphs.*
   - A hypercube*, whose vertex set is $\{0,1\}^d$, with an edge between two vectors if they differ in exactly one entry;*
   - The Petersen graph*, whose vertices are the 2-element subsets of $\{1,2,3,4,5\}$, with an edge between two subsets if they are disjoint;*
   - A visibility graph*, whose vertex set is a finite set $X \subset \mathbb{R}^2$, with an edge between two points if they can see each other, i.e., if the line segment between them contains no other point of X.*

   - The diameter of a hypercube is d. This is because any two vertices can differ in at most d entries, which implies that the distance between them is at most d. Moreover, for any vertex u, there exists exactly one vertex v that differs from u in exactly d entries, so the distance between u and v is d.

   - The diameter of the Petersen graph is 2. If two subsets are adjacent, they are at distance 1. If two subsets are not adjacent, they share one element, so they look like $\{1,2\}, \{1,3\}$. Then both are adjacent to $\{4,5\}$, so the distance is 2.

   - If no three points are on a line, then every two points can see each other, so the diameter is 1. If all the points are on one line, then the graph is a path of length $|X| - 1$, and the diameter is $|X| - 1$. In all other cases, for any two points that are not adjacent, there must be another point that both can see, so they are at distance 2 and the diameter is 2. To see that such a point exists, let ℓ be the line through the two points, and take a point that is at minimum distance from this line.

7. *Prove that the following Greedy Forest-Growing algorithm returns a minimum-weight spanning tree, given a graph G and $w : E(G) \to \mathbb{R}_+$.*

Greedy Forest-Growing Algorithm

(1) *Start with the empty graph F, and set $S = E(G)$;*

(2) *Find $e \in S$ with minimum $w(e)$; if $S = \emptyset$ go to (4);*

(3) *Add e to F, unless that creates a cycle; remove e from S; go back to (2);*

(4) *If $|E(F)| = |V(G)| - 1$, return F, else return "disconnected".*

If G is not connected, then the algorithm will reach step (4) with a forest with $< |V(G)|-1$ edges, and return "disconnected". Any edge that was not added would create a cycle in this forest, and thus cannot connect the components of the forest. Thus G is indeed not connected.

Assume that G is connected. Suppose the algorithm returns the forest F. We must have $|E(F)| = |V(G)| - 1$, so F is a spanning tree.

Let T^* be a minimum-weight spanning tree that shares as many initial edges as possible with F (i.e., if the algorithm adds the edges to F in the order $e_1, \ldots, e_{n-1}$, then T^* contains a longer list of the form $e_1, e_2, \ldots, e_{k-1}, e_k$ than any other minimum-weight spanning tree). We can assume $T^* \neq F$, otherwise we are done. Let e be the first edge not in T^* that was put into F during the algorithm, and let F' be the forest just before that. There must be a cycle C in $T^* + e$, and $C - e$ must contain an edge f that could have been added to F' without creating a cycle.

To see this last fact, observe that if $F' + xy$ contains a cycle, then x and y must be in the same connected component of F'. Thus, if every edge of $C - e$ creates a cycle in F', then both endpoints of e would be in the same component, contradicting the fact that e does not create a cycle in F'.

So when the algorithm chose e, f was also a candidate, and it chose e over f, so $w(e) \leq w(f)$. Then $T^* + e - f$ is also a minimum-weight spanning tree, which has more initial edges in common with F than T^*, a contradiction.

8. *Design a "greedy removal" algorithm for minimum-weight spanning trees, which starts with the whole graph and removes heavy edges. Prove that it works.*

Greedy Removal Algorithm

(1) Start with $H = G$ and $S = \emptyset$;

(2) Pick $e \in E(H) \backslash S$ with maximum $w(e)$; if $E(H) \backslash S = \emptyset$, go to (4);

(3) If $H - e$ is connected, then remove e from H; add e to S; go to (2);

(4) If H is connected, return H; else return "disconnected".

If G is not connected, then the algorithm will just put every edge in S and correctly return "disconnected". If G is connected, then H will be connected throughout.

Just for variation, we'll use a different type of argument than we used for the tree-growing and forest-growing algorithms.

Call a connected subgraph of G *good* if it contains a minimum-weight spanning tree of G. We'll show that the subgraph returned by G is good. Since no edge removal could disconnect an intermediate H, the final H is connected. It contains no cycles, since otherwise an edge of that cycle could still be removed (and could have been removed in any earlier step, so cannot have been put in S). Hence the final H is a tree, and spanning because $V(H) = V(G)$ all along. So it is a spanning tree that is good, which implies that it is a minimum-weight spanning tree itself.

Suppose we have a good H in step (2), containing a minimum-weight spanning tree T^* of G. If we remove $e \notin T^*$, then clearly $H - e$ is good, so assume $e \in T^*$. Write $e = uv$. Removing e from T^* must disconnect it into two components, say U containing u and V containing v. Since $H - e$ is connected, there must be a path in it from u to v, which must contain an edge f with one endpoint in U and one in V (and $f \notin T^*$). We do not have $f \in S$, since S is always disjoint from $E(H)$. So $f \in E(H) \backslash S$, which implies $w(e) \geq w(f)$. Therefore $T^{**} = T^* - e + f$ is a minimum-weight spanning tree contained in $H - e$, so $H - e$ is good.

*9. *Let T be a tree with t edges and G a graph. Prove that if $|E(G)| \geq t \cdot |V(G)|$, then T is a subgraph of G.*

Write $n = |V(G)|$. We use induction on t and n, with the induction statement "Every graph G with n edges and $|E(G)| \geq tn$ contains every tree with t edges". We assume that any graph on n vertices with $|E(G)| \geq (t-1)n$ contains any tree with $t-1$ edges, and that any graph with $n-1$ vertices and $|E(G)| \geq t(n-1)$ contains any tree with t edges. When $t = 1$ or $n = 1$ the statement is trivial.

Let G be a graph with at least tn edges, and T a tree with t edges. Since T is a tree, it has a leaf u, adjacent to a vertex v. Note that $T - u$ is a tree on $t-1$ vertices.

Suppose that $\delta(G) \geq t$. By induction and the fact that $|E(G)| \geq tn > (t-1)n$, G contains $T - u$ as a subgraph. Let us fix one embedding of $T - u$ in $G - x$ (there could be more than one), so in particular we can view v as a vertex in G. By the assumption that $\delta(G) \geq t$, we have $d_G(v) \geq t$. On the other hand, $T - u$ has $t-1$ edges, so we have $d_{T-u}(v) \leq t-1$. Hence v is connected to at least one vertex w in G that is not in (this embedding of) $T - u$. Then adding vw to $T - u$ gives a subgraph of G isomorphic to T.

Suppose that $\delta(G) < t$, so there is a vertex $y \in V(G)$ with $d_G(y) \leq t-1$. Then $G - y$ is a graph on $n-1$ vertices with at least $tn - (t-1) > t(n-1)$ edges. By induction, $G - y$ contains T, hence so does G.

Alternative solution: Instead of the induction, we could first prove the statement for any G with $\delta(G) \geq t$, with the argument above. Then we could use the fact that every graph G contains a subgraph with $\delta(H) \geq |E(G)|/|V(G)|$, which can be proved by repeatedly removing a vertex of lowest degree and some calculation.

Note: With the proof above we can do a little better, and weaken the condition to $|E(G)| \geq (t-1)|V(G)|$. The Erdős-Sós Conjecture claims that even the condition $|E(G)| \geq \frac{1}{2}(t-1)|V(G)|$ suffices; this seems to have recently been proved by Ajtai, Komlós, Simonovits, and Szemerédi for sufficiently large $|V(G)|$.

*10. *Let $X = \{1, \ldots, n\}$ and let $\mathcal{S}$ be a set of n subsets of X. Use trees to prove that X contains an element x such that adding it to the sets of $\mathcal{S}$ gives n distinct subsets. In other words, there is an $x \in X$ such that $|\{S \cup \{x\} : S \in \mathcal{S}\}| = n$.*

Define a graph G with vertex set $\mathcal{S}$, and connect S and S' by an edge if $S \cup \{x\} = S'$ for some $x \in X$. Moreover, give the edge between S and $S \cup \{x\}$ the label x.

We claim that one of the labels does not occur at all. This is what we want, because if x does not occur as a label, there is no $S \in \mathcal{S}$ for which $S \cup \{x\} \in \mathcal{S}$, which implies that $|\{S \cup \{x\} : S \in \mathcal{S}\}| = n$.

To prove the claim we need the following observation. If there is a cycle in G, every label occurs an even number of times in that cycle. Indeed, moving around the cycle we have a sequence of additions and removals. To get from a set S back to S in that way, every addition has to be undone by a removal of the same element, and vice versa.

Now we run the following algorithm. For any cycle in the graph, we remove an edge, until there are no more cycles left. Observe that after each step, the label of the removed edge still occurs elsewhere in the cycle, by the observation above. When the algorithm finishes, we have a forest on $V(G)$ with the same set of labels as G. But the forest has at most $|V(G)| - 1$ edges, hence it has at most that many labels. Since G has the same set of labels, one label must be missing.