Math 530 Spring 2022, Lecture 14: Trees

website: `https://www.cip.ifi.lmu.de/~grinberg/t/22s`

1. Trees and arborescences (cont'd)

1.1. Forests and trees (cont'd)

Last time, we introduced trees. Let us quickly recall some of their properties:

If $T = (V, E, \varphi)$ is a tree, then...

- T is a connected forest. (This is how trees were defined.) Thus, T has no cycles. (This is how forests were defined.)
- we have $|E| = |V| - 1$. (This follows from the implication T1$\Longrightarrow$T4 in the tree equivalence theorem.)
- adding any new edge to T creates a cycle. (This follows from the implication T1$\Longrightarrow$T6 in the tree equivalence theorem.)
- removing any edge from T yields a disconnected (i.e., non-connected) graph. (This follows from the implication T1$\Longrightarrow$T7 in the tree equivalence theorem.)
- for each $u \in V$ and $v \in V$, there is a **unique** backtrack-free walk from u to v. (This follows from the implication T1$\Longrightarrow$T3 in the tree equivalence theorem.) Moreover, this backtrack-free walk is a path (since any walk from u to v contains a path from u to v).

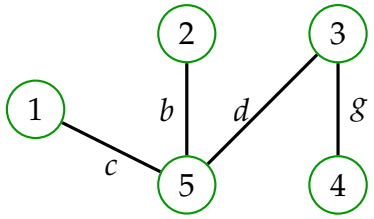
Remark 1.1.1. Computer scientists use some notions of "trees" that are similar to ours, but not quite the same. In particular, their trees often have **roots** (i.e., one vertex is chosen to be called "the root" of the tree), which leads to a parent/child relationship on each edge (namely: the endpoint closer to the root is called the "parent" of the endpoint further away from the root). Often, they also impose a total order on the children of each given vertex. With these extra data, a tree can be used for addressing objects, since each vertex has a unique "path description" from the root leading to it (e.g., "the second child of the fourth child of the root"). But this all is going too far afield for us here; we are mainly interested in trees as graphs, and won't impose any extra structure unless we need it for something.

1.2. Leaves

Continuing with our faux-botanical terminology, we define leaves in a tree:

Definition 1.2.1. Let T be a tree. A vertex of T is said to be a **leaf** if its degree is 1.

For example, the tree

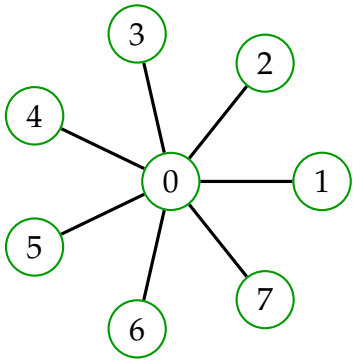


has three leaves: 1, 2 and 4.

How to find a tree with as many leaves as possible (for a given number of vertices)? For any $n \geq 3$, the simple graph

$$(\{0, 1, \ldots, n-1\}, \{0i \mid i > 0\})$$

is a tree (when considered as a multigraph), and has $n-1$ leaves (namely, all of $1, 2, \ldots, n-1$). This tree is called an **n-star graph**, as it looks as follows:



(for $n = 8$).

It is easy to see that no tree with $n \geq 3$ vertices can have more than $n-1$ leaves, so the n-star graph is optimal in this sense. Note that for $n = 2$, the n-star graph has 2 leaves, not 1.

How to find a tree with as few leaves as possible? For any $n \geq 2$, the n-path graph

$P_n =$ 

is a tree with only 2 leaves (viz., the vertices 1 and n). Can we find a tree with fewer leaves? For $n = 1$, yes, because the 1-path graph P_1 (this is simply the graph with 1 vertex and no edges) has no leaves at all. However, for $n \geq 2$, the n-path graph is the best we can do:

Theorem 1.2.2. Let T be a tree with at least 2 vertices. Then:

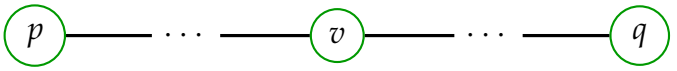
(a) The tree T has at least 2 leaves.

(b) Let v be a vertex of T. Then, there exist two distinct leaves p and q of T such that v lies on the path from p to q.

Note that I'm saying "the path" rather than "a path" here. This is allowed, because in a tree, for any two vertices p and q, there is a **unique** path from p to q. This follows from Statement T2 in the tree equivalence theorem.

Proof of Theorem 1.2.2. **(b)** We apply a variant of the "longest path trick": Among all paths that contain the vertex v, let $\mathbf{w}$ be a longest one. Let p be the starting point of $\mathbf{w}$, and let q be the ending point of $\mathbf{w}$. We shall show that p and q are two distinct leaves.

[Here is a picture of $\mathbf{w}$, for what it's worth:



.

Of course, the tree T can have other edges as well, not just those of $\mathbf{w}$.]

First, we observe that T is connected (since T is a tree), and has at least one vertex u distinct from v (since T has at least 2 vertices). Hence, T has a path $\mathbf{r}$ that connects v to u. This path $\mathbf{r}$ must contain at least one edge (since $u \neq v$). Thus, we have found a path $\mathbf{r}$ of T that contains v and contains at least one edge. Hence, the path $\mathbf{w}$ must contain at least one edge as well (since $\mathbf{w}$ is a longest path that contains v, and thus cannot be shorter than $\mathbf{r}$). Since $\mathbf{w}$ is a path from p to q, we thus conclude that $p \neq q$ (because if a path contains at least one edge, then its starting point is distinct from its ending point).

Now, assume (for the sake of contradiction) that p is not a leaf. Then, $\deg p \neq 1$. The path $\mathbf{w}$ already contains one edge that contains p (namely, the first edge of $\mathbf{w}$). Since $\deg p \neq 1$, there must be another edge f of T that contains $\mathbf{w}$. Consider this f. Let p' be its endpoint distinct from p (if f is a loop, then we set $p' = p$). Appending this edge f (and its endpoint) to the beginning of the path $\mathbf{w}$, we obtain a backtrack-free walk

$$\left(p', f, \underbrace{p, \ldots, v, \ldots, q}_{\text{This is } \mathbf{w}}\right)$$

(this is backtrack-free since f is not the first edge of $\mathbf{w}$). According to Proposition 1.1.2 in Lecture 13, this backtrack-free walk either is a path or contains a cycle. Since T has no cycle (because T is a forest), we thus conclude that this backtrack-free walk is a path. It is furthermore a path that contains v and is longer than $\mathbf{w}$ (longer by 1, in fact). But this contradicts the fact that $\mathbf{w}$ is

a longest path that contains v. This contradiction shows that our assumption (that p is not a leaf) was wrong.

Hence, p is a leaf. A similar argument shows that q is a leaf (here, we need to append the new edge at the end of $\mathbf{w}$ rather than at the beginning). Thus, p and q are two distinct leaves of T (distinct because $p \neq q$) such that v lies on the path from p to q (since v lies on the path $\mathbf{w}$, which is a path from p to q). This proves Theorem 1.2.2 **(b)**.

(a) Pick any vertex v of T. Then, Theorem 1.2.2 **(b)** shows that there exist two distinct leaves p and q of T such that v lies on the path from p to q. Thus, in particular, there exist two distinct leaves p and q of T. In other words, T has at least two leaves. This proves Theorem 1.2.2 **(a)**.

[*Remark:* Another way to prove part **(a)** is to write the tree T as $T = (V, E, \varphi)$, and recall the handshake lemma, which yields

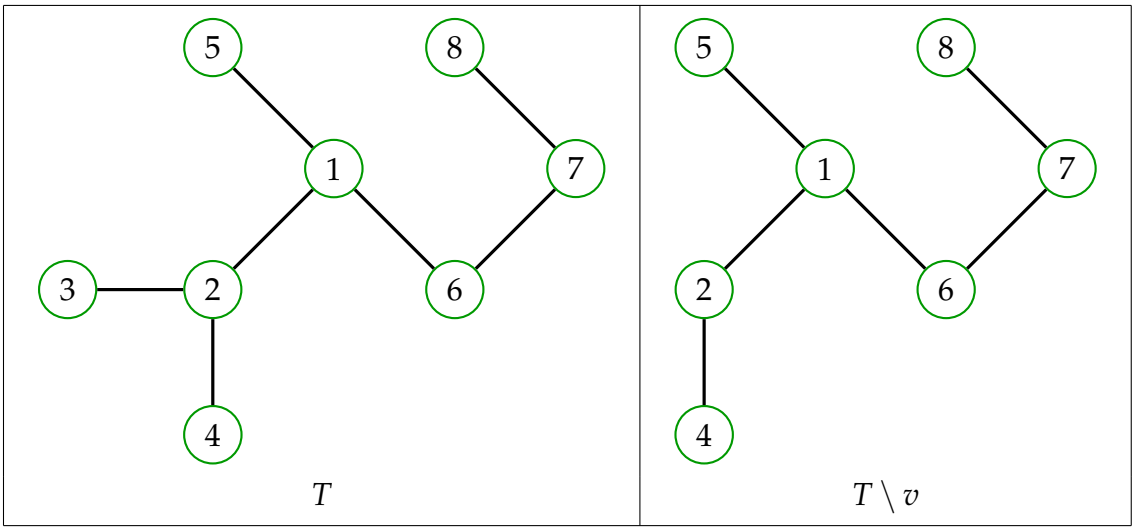
$$\sum_{v \in V} \deg v = 2 \cdot |E| = 2 \cdot (|V| - 1) \qquad (\text{since } |E| = |V| - 1 \text{ in a tree})$$
$$= 2 \cdot |V| - 2.$$

Since each $v \in V$ satisfies $\deg v \geq 1$ (why?), this equality entails that at least two vertices $v \in V$ must satisfy $\deg v \leq 1$ (since otherwise, the sum $\sum_{v \in V} \deg v$ would be $\geq 2 \cdot |V| - 1$), and therefore these two vertices are leaves.] □

Leaves are particularly helpful for performing induction on trees. The formal reason for this is the following theorem:

Theorem 1.2.3 (induction principle for trees)**.** Let T be a tree with at least 2 vertices. Let v be a leaf of T. Let $T \setminus v$ be the multigraph obtained from T by removing v and all edges that contain v (note that there is only one such edge, since v is a leaf). Then, $T \setminus v$ is again a tree.

Here is an example of a tree T and of the smaller tree $T \setminus v$ obtained by removing a leaf v (namely, $v = 3$):



Proof of Theorem 1.2.3. Write T as $T = (V, E, \varphi)$. Thus, $T \setminus v$ is the induced subgraph $T[V \setminus \{v\}]$.

The graph T is a tree, thus a forest; hence, it has no cycles. Thus, the graph $T \setminus v$ has no cycles either. Hence, it is a forest.

Furthermore, this forest $T \setminus v$ has at least 1 vertex (since T has at least 2 vertices).

We shall now show that any two vertices p and q of $T \setminus v$ are path-connected in $T \setminus v$.

Indeed, let p and q be two vertices of $T \setminus v$. Then, p and q are path-connected in T (since T is connected). Hence, there exists a path $\mathbf{w}$ from p to q in T. Consider this path $\mathbf{w}$. Note that v is neither the starting point nor the ending point of this path $\mathbf{w}$ (since p and q are vertices of $T \setminus v$, and thus distinct from v). Hence, if v was a vertex of $\mathbf{w}$, then $\mathbf{w}$ would contain **two distinct** edges that contain v (namely, the edge just before v and the edge just after v). But this is impossible, since there is only one edge available that contains v (because v is a leaf). Thus, v cannot be a vertex of $\mathbf{w}$. Hence, the path $\mathbf{w}$ does not use the vertex v, and thus is a path in the graph $T \setminus v$ as well. So the vertices p and q are path-connected in $T \setminus v$.

We have now shown that any two vertices p and q of $T \setminus v$ are path-connected in $T \setminus v$. This shows that $T \setminus v$ is connected (since $T \setminus v$ has at least 1 vertex). Hence, $T \setminus v$ is a tree (since $T \setminus v$ is a forest). □

Theorem 1.2.3 has a converse as well:

Theorem 1.2.4. Let G be a multigraph. Let v be a vertex of G such that $\deg v = 1$ and such that $G \setminus v$ is a tree. (Here, $G \setminus v$ means the multigraph obtained from G by removing the vertex v and all edges that contain v.) Then, G is a tree.

Proof. Left to the reader. (The main step is to show that a cycle of G cannot contain v.) □

Theorem 1.2.3 helps prove many properties of trees by induction on the number of vertices. In the induction step, remove a leaf v and apply the induction hypothesis to $T \setminus v$. We move on to another use of trees, though.

1.3. Spanning trees

First we define a concept that makes sense for any multigraphs:

Definition 1.3.1. A **spanning subgraph** of a multigraph $G = (V, E, \varphi)$ means a multigraph of the form $(V, F, \varphi\mid_F)$, where F is a subset of E.

In other words, it means a submultigraph of G with the same vertex set as G.

In other words, it means a multigraph obtained from G by removing some edges, but leaving all vertices undisturbed.

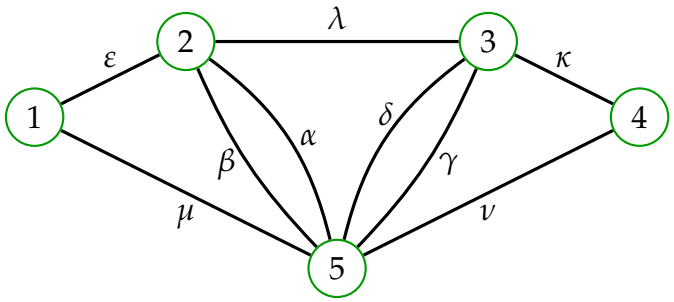
Compare this to the notion of an induced subgraph:

- To build an **induced** subgraph, we throw away some vertices but keep all the edges that we can keep. (As usual in mathematics, the words "some vertices" include "no vertices" and "all vertices".)
- In contrast, to build a **spanning** subgraph, we keep all vertices but throw away some edges.

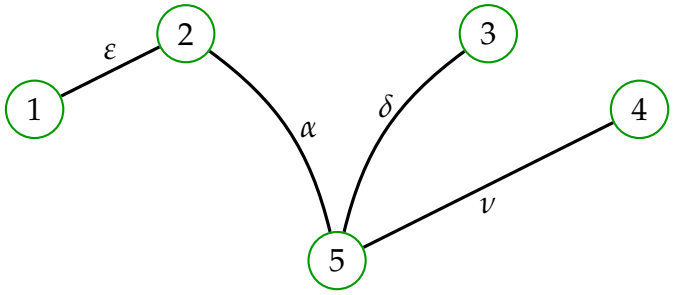
Spanning subgraphs are particularly useful when they are trees:

Definition 1.3.2. A **spanning tree** of a multigraph G means a spanning subgraph of G that is a tree.

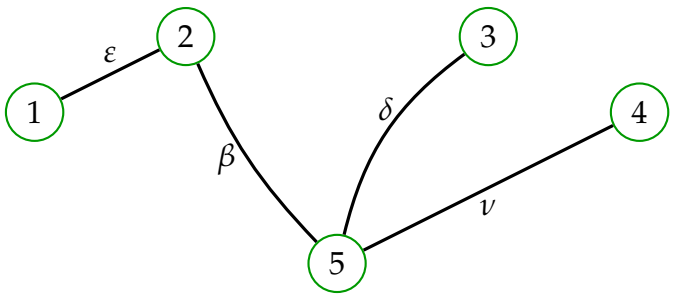
Example 1.3.3. Let G be the following multigraph:



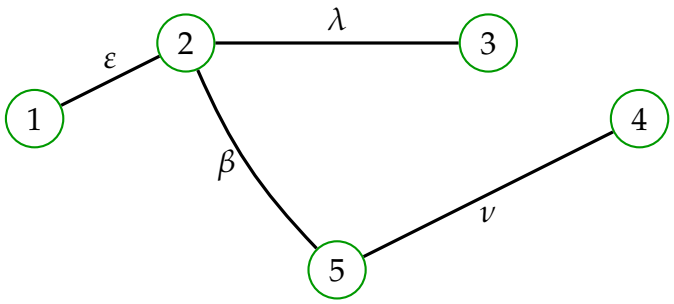
Here is a spanning tree of G:



Here is another:



(Yes, this is a different one, because $\alpha \neq \beta$.) And here is yet another spanning tree of G:



.

A spanning tree of a graph G can be regarded as a minimum "backbone" of G – that is, a way to keep G connected using as few edges as possible. Of course, if G is not connected, then this is not possible at all, so G has no spanning trees in this case. The best one can hope for is a spanning subgraph that keeps each component of G connected using as few edges as possible. This is known as a "spanning forest":

Definition 1.3.4. A **spanning forest** of a multigraph G means a spanning subgraph H of G that is a forest and satisfies $\operatorname{conn} H = \operatorname{conn} G$.

When G is a connected multigraph, a spanning forest of G means the same as a spanning tree of G.

The following theorem is crucial, which is why we will outline several proofs:

Theorem 1.3.5. Each connected multigraph G has at least one spanning tree.

First proof. Let G be a connected multigraph. We want to construct a spanning tree of G. We try to achieve this by removing edges from G one by one, until G becomes a tree. When doing so, we must be careful not to disconnect the graph (i.e., not to destroy its connectedness). According to a result we have seen a while ago (Theorem 1.1.14 in Lecture 8), this can be achieved by making sure that we never remove a bridge (i.e., an edge that appears in no cycle). Thus, we keep removing non-bridges (i.e., edges that are not bridges) as long as we can (i.e., until we end up with a graph in which every edge is a bridge).

So here is the algorithm: We start with G, and we successively remove non-bridges one by one until we no longer have any non-bridges left[1]. This procedure cannot go on forever, since G has only finitely many edges. Thus, after finitely many steps, we will end up with a graph that has no non-bridges any more. This resulting graph therefore has no cycles (since any cycle would have

[1] **Warning:** We cannot remove several non-bridges at once! We have to remove them one by one. Indeed, if e and f are two non-bridges of G, then there is no guarantee that f remains a non-bridge in $G \setminus e$. So we cannot remove both e and f simultaneously; we have to remove one of them and check whether the other is still a non-bridge.

at least one edge, and this edge would be a non-bridge), but is still connected (since G was connected, and we never lost connectivity as we removed only non-bridges). Thus, this resulting graph is a tree. Since it is also a spanning subgraph of G (by construction), it is therefore a spanning tree of G. This proves Theorem 1.3.5. □

Second proof (sketched). In the above first proof, we constructed a spanning tree of G by starting with G and successively removing edges until we got a tree. Now let us take the opposite strategy: Start with an empty graph on the same vertex set as G, and successively add edges (from G) until we get a connected graph.

Here are some details: We start with a graph L that has the same vertex set as G, but has no edges. Now, we inspect all edges e of G one by one (in some order). For each such edge e, we add it to L, but only if it does not create a cycle in L; otherwise, we discard this edge. Notice that adding an edge e with endpoints u and v to L creates a cycle if and only if u and v lie in the same component of L (before we add e). Thus, we only add an edge to L if its endpoints lie in different components of L; otherwise, we discard it. This way, at the end of the procedure, our graph L will still have no cycles (since we never create any cycles). In other words, it will be a forest.

Let me denote this forest by H. (Thus, H is the L at the end of the procedure.) I claim that this forest H is a spanning tree of G. Why? Since we know that H is a forest, we only need to show that H is connected. Assume the contrary. Thus, there is at least one edge e of G whose endpoints lie in different components of H (why?). This edge e is therefore not an edge of H. Therefore, at some point during our construction of H, we must have discarded this edge e (instead of adding it to L). As we know, this means that the endpoints of e used to lie in the same component of L at the point at which we discarded e. But this entails that these two endpoints lie in the same component of L at the end of the procedure as well (because the graph L never loses any edges during the procedure, so that any two vertices that used to lie in the same component of L at some point will still lie in the same component of L ever after). In other words, the endpoints of e lie in the same component of H. This contradicts our assumption that the endpoints of e lie in different components of H. This contradiction completes our proof that H is connected. Hence, H is a spanning tree of G, and we have proved Theorem 1.3.5 again. □

Third proof. This proof takes yet another approach to constructing a spanning tree of G: We choose an arbitrary vertex r of G, and then progressively "spread a rumor" from r. The rumor starts at vertex r. On day 0, only r has heard the rumor. Every day, every vertex that knows the rumor spreads it to all its neighbors (i.e., all vertices adjacent to it). Since G is connected, the rumor will eventually spread to every vertex of G. Now, each vertex v (other than r) remembers which other vertex v' it has first heard the rumor from (if it heard it from several vertices at the same time, it just picks one of them), and picks

some edge e_v that has endpoints v and v' (such an edge must exist, since v must have heard the rumor from a neighbor). The edges e_v for all $v \in V \setminus \{r\}$ (where V is the vertex set of G) then form a spanning tree of G (that is, the graph with vertex set V and edge set $\{e_v \mid v \in V \setminus \{r\}\}$ is a spanning tree). Why?

Intuitively, this is quite convincing: This graph cannot have cycles (because that would require a time loop) and must be connected (because for any vertex v, we can trace back the path of the rumor from r to v by following the edges e_v backwards). To obtain a rigorous proof, we formalize this construction mathematically:

Write G as $G = (V, E, \varphi)$. Choose any vertex r of G.

We shall recursively construct a sequence of subgraphs

$$(V_0, E_0, \varphi_0), \quad (V_1, E_1, \varphi_1), \quad (V_2, E_2, \varphi_2), \quad \ldots$$

of G. The idea behind these subgraphs is that for each $i \in \mathbb{N}$, the set V_i will consist of all vertices v that have heard the rumor by day i, and the set E_i will consist of the corresponding edges e_v. The map φ_i will be the restriction of φ to E_i, of course.

Here is the exact construction of this sequence of subgraphs:

- *Recursion base:* Set $V_0 := \{r\}$ and $E_0 := \varnothing$. Let φ_0 be the restriction of φ to the (empty) set E_0.
- *Recursion step:* Let $i \in \mathbb{N}$. Assume that the subgraph (V_i, E_i, φ_i) of G has already been defined. Now, we set

  $$V_{i+1} := V_i \cup \{v \in V \mid v \text{ is adjacent to some vertex in } V_i\}.$$

  For each $v \in V_{i+1} \setminus V_i$, we choose **one** edge e_v that joins[2] v to a vertex in V_i (such an edge exists, since $v \in V_{i+1}$; if there are several, we just choose a random one). Set

  $$E_{i+1} := E_i \cup \{e_v \mid v \in V_{i+1} \setminus V_i\}.$$

  Finally, we let φ_{i+1} be the restriction of the map φ to the set E_{i+1}. This is a map from E_{i+1} to $\mathcal{P}_{1,2}(V_{i+1})$ (because any edge e_v with $v \in V_{i+1} \setminus V_i$ has one endpoint v in $V_{i+1} \setminus V_i \subseteq V_{i+1}$ and the other endpoint in $V_i \subseteq V_{i+1}$). Thus, $(V_{i+1}, E_{i+1}, \varphi_{i+1})$ is a well-defined subgraph of G.

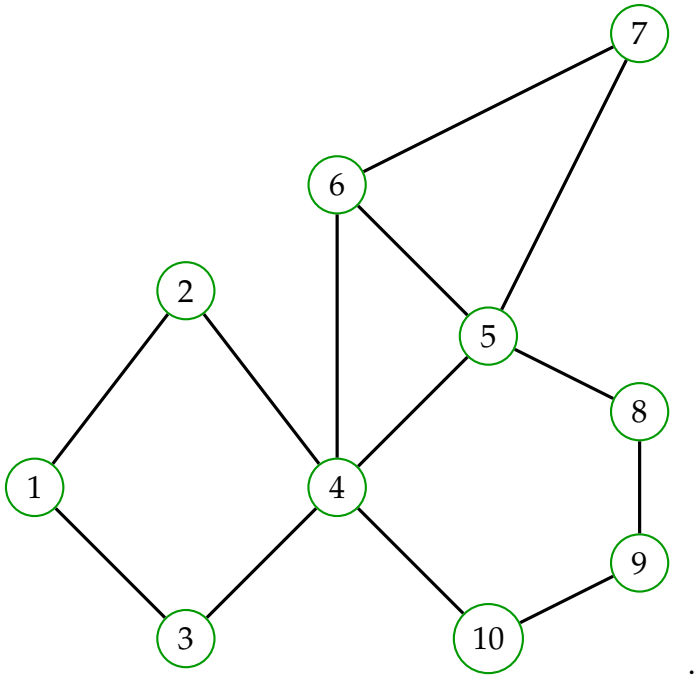
This construction yields that (V_i, E_i, φ_i) is a subgraph of $(V_{i+1}, E_{i+1}, \varphi_{i+1})$ for each $i \in \mathbb{N}$. Hence, $V_0 \subseteq V_1 \subseteq V_2 \subseteq \cdots$, so that $|V_0| \leq |V_1| \leq |V_2| \leq \cdots$. Since a sequence of integers bounded from above cannot keep increasing forever (and the sizes $|V_i|$ are bounded from above by $|V|$, since each V_i is a subset of V), we thus see that there exists some $i \in \mathbb{N}$ such that $|V_i| = |V_{i+1}|$. Consider this i. From $|V_i| = |V_{i+1}|$, we obtain $V_i = V_{i+1}$ (since $V_i \subseteq V_{i+1}$).

[2] We say that an edge **joins** a vertex p to a vertex q if the endpoints of this edge are p and q.

In our colloquial model above, $V_i = V_{i+1}$ means that no new vertices learn the rumor on day $i+1$; it is reasonable to expect that at this point, every vertex has heard the rumor. In other words, we claim that $V_i = V$. A rigorous proof of this can be easily given using the fact that G is connected[3].

Now, we claim that the subgraph (V_i, E_i, φ_i) is a spanning tree of G. To see this, we must show that this subgraph is a forest and is connected (since $V_i = V$ already shows that it is a spanning subgraph). Before we do this, let us give an example:

Example 1.3.6. Let G be the following multigraph:



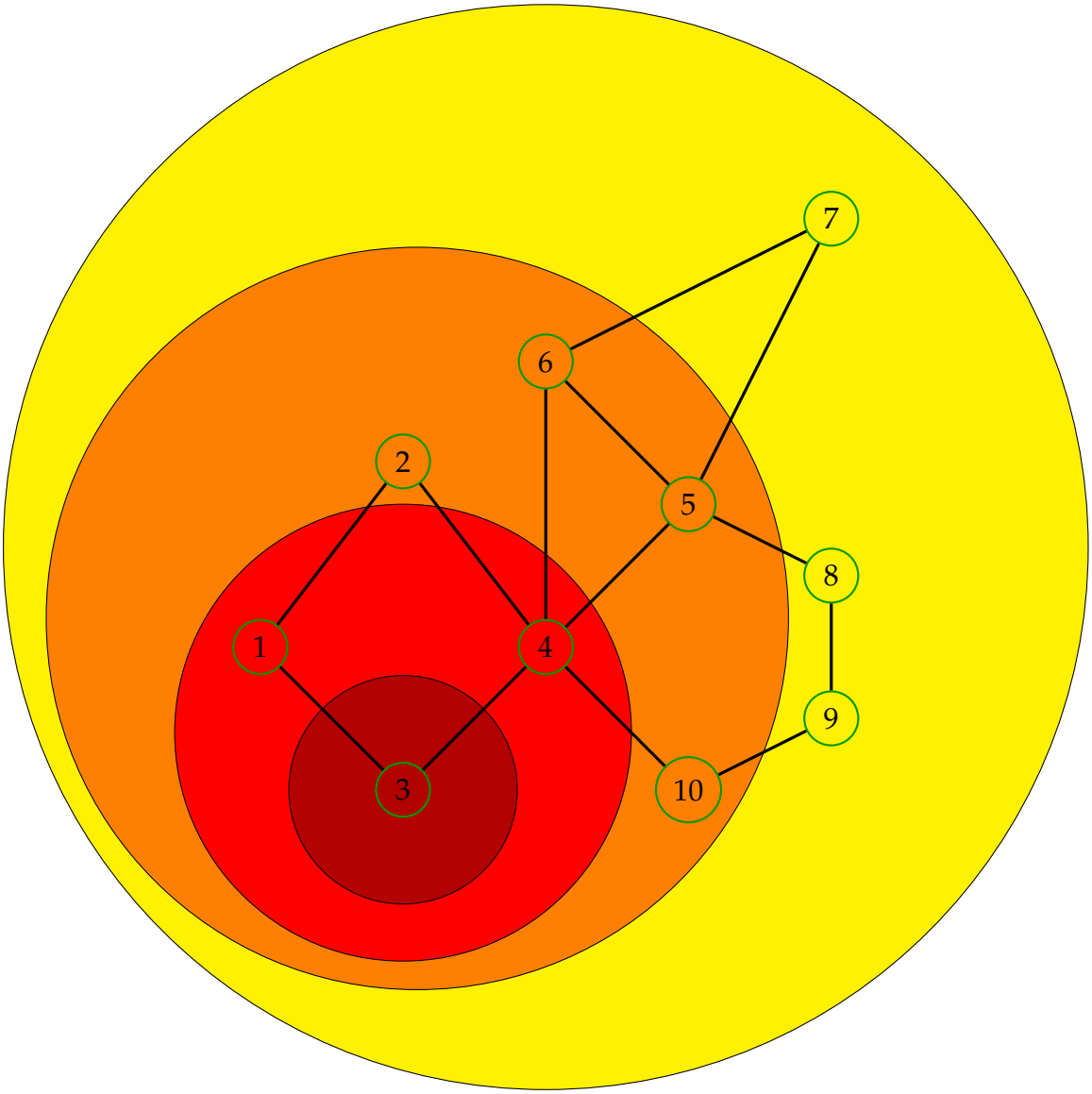
.

Set $r = 3$. Then, the above construction yields

$$\begin{aligned} V_0 &= \{3\}, \\ V_1 &= \{3,1,4\}, \\ V_2 &= \{3,1,4,2,5,6,10\}, \\ V_3 &= \{3,1,4,2,5,6,10,8,9,7\} = V, \end{aligned}$$

[3]Here is the *proof* in detail: We must show that $V_i = V$. Assume the contrary. Thus, there exists a vertex $u \in V \setminus V_i$. Consider this u. The path from r to u starts at a vertex in V_i (since $r \in V_0 \subseteq V_i$) and ends at a vertex in $V \setminus V_i$ (since $u \in V \setminus V_i$). Thus, it must cross over from V_i into $V \setminus V_i$ at some point. Therefore, there exists an edge with one endpoint in V_i and the other endpoint in $V \setminus V_i$. Let v and w be these two endpoints, so that $v \in V_i$ and $w \in V \setminus V_i$. Then, w is adjacent to some vertex in V_i (namely, to v), and therefore belongs to V_{i+1} (by the definition of V_{i+1}). Hence, $w \in V_{i+1} = V_i$. But this contradicts $w \notin V \setminus V_i$. This contradiction shows that our assumption was wrong, qed.

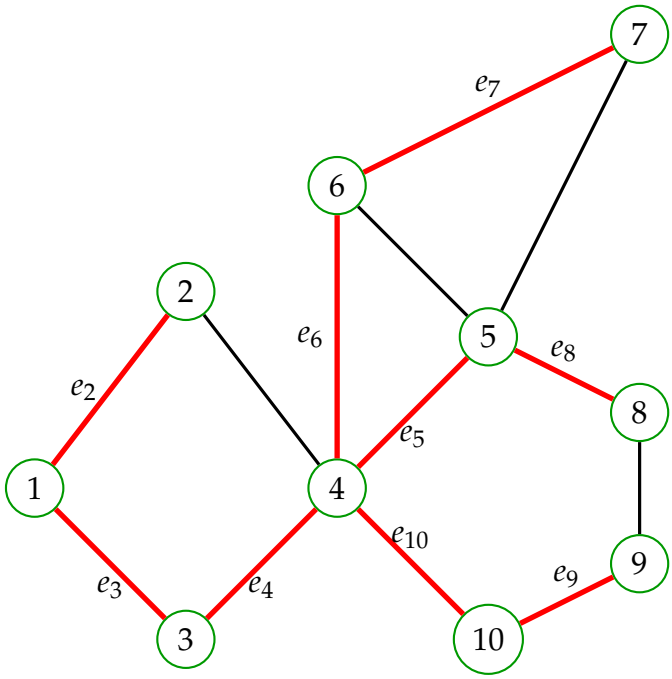
so that $V_k = V$ for all $k \geq 3$. Thus, we can take $i = 3$. Here is an image of the V_k as progressively growing circles:



.

(The dark-red inner circle is V_0; the red circle is V_1; the orange circle is V_2; the yellow circle is $V_3 = V_4 = V_5 = \cdots = V$.) Finally, the edges e_v can be

chosen to be the following (we are painting them red for clarity):



(Here, we have made two choices: We chose e_2 to be the edge joining 2 with 1 rather than the edge joining 2 with 4, and we chose e_7 to be the edge joining 7 with 6 rather than 7 with 5. The other options would have been equally fine.)

We now return to the general proof. Let us first show the following:

> *Claim 1:* Let $j \in \mathbb{N}$. Each vertex of the graph (V_j, E_j, φ_j) is path-connected to r in this graph.

[*Proof of Claim 1:* We induct on j:

Base case: For $j = 0$, Claim 1 is obvious, since $V_0 = \{r\}$ (so the only vertex of the graph in question is r itself).

Induction step: Fix some positive integer k. Assume (as the induction hypothesis) that Claim 1 holds for $j = k - 1$. That is, each vertex of the graph $(V_{k-1}, E_{k-1}, \varphi_{k-1})$ is path-connected to r in this graph.

Now, let v be a vertex of the graph (V_k, E_k, φ_k). We must show that v is path-connected to r in this graph. If $v \in V_{k-1}$, then this follows from the induction hypothesis (since $(V_{k-1}, E_{k-1}, \varphi_{k-1})$ is a subgraph of (V_k, E_k, φ_k)). Thus, we WLOG assume that $v \notin V_{k-1}$ from now on. Hence, $v \in V_k \setminus V_{k-1}$. According to the recursive definition of E_k, this entails that there is an edge $e_v \in E_k$ that joins v to some vertex $u \in V_{k-1}$. Consider this latter vertex u. Then, v is path-connected to u in the graph (V_k, E_k, φ_k) (since the edge e_v provides a length-1 path from v to u). However, u is path-connected to r in the graph

$(V_{k-1}, E_{k-1}, \varphi_{k-1})$ (by the induction hypothesis, since $u \in V_{k-1}$), hence also in the graph (V_k, E_k, φ_k) (since $(V_{k-1}, E_{k-1}, \varphi_{k-1})$ is a subgraph of (V_k, E_k, φ_k)). Since the relation "path-connected" is transitive, we conclude from the previous two sentences that v is path-connected to r in the graph (V_k, E_k, φ_k).

So we have shown that each vertex v of the graph (V_k, E_k, φ_k) is path-connected to r in the graph (V_k, E_k, φ_k). In other words, Claim 1 holds for $j = k$. This completes the induction step, and Claim 1 is proved.]

Claim 1 (applied to $j = i$) shows that each vertex of the graph (V_i, E_i, φ_i) is path-connected to r in this graph. Since the relation "path-connected" is an equivalence relation, this entails that any two vertices of this graph are path-connected. Thus, the graph (V_i, E_i, φ_i) is connected (since it has at least one vertex). It remains to prove that this graph (V_i, E_i, φ_i) is a forest.

Again, we do this using an auxiliary claim:

Claim 2: Let $j \in \mathbb{N}$. Then, the graph $\left(V_j, E_j, \varphi_j\right)$ has no cycles.

[*Proof of Claim 2:* We induct on j:

Base case: The graph (V_0, E_0, φ_0) has no edges (because $E_0 = \varnothing$) and thus no cycles. Thus, Claim 2 holds for $j = 0$.

Induction step: Fix some positive integer k. Assume (as the induction hypothesis) that Claim 2 holds for $j = k - 1$. That is, the graph $(V_{k-1}, E_{k-1}, \varphi_{k-1})$ has no cycles.

Now, let $\mathbf{c}$ be a cycle of the graph (V_k, E_k, φ_k). Then, $\mathbf{c}$ must use at least one edge from $E_k \setminus E_{k-1}$ (since otherwise, $\mathbf{c}$ would be a cycle of the graph $(V_{k-1}, E_{k-1}, \varphi_{k-1})$, but this is impossible, since $(V_{k-1}, E_{k-1}, \varphi_{k-1})$ has no cycles). However, each edge from $E_k \setminus E_{k-1}$ has the form e_v for some $v \in V_k \setminus V_{k-1}$ (because of how E_k was defined). Thus, $\mathbf{c}$ must have an edge of this form. Consider the corresponding vertex $v \in V_k \setminus V_{k-1}$. The cycle $\mathbf{c}$ contains the edge e_v and therefore also contains its endpoint v. However, (again by the definition of E_k) the edge e_v is the **only** edge in E_k that contains the vertex v. Since the edge e_v is not a loop (because it joins the vertex $v \in V_k \setminus V_{k-1}$ with a vertex in V_{k-1}), we thus conclude that the vertex v has degree 1 in the graph (V_k, E_k, φ_k). Thus, the vertex v cannot be contained in any cycle of (V_k, E_k, φ_k) (because a cycle cannot contain a vertex of degree 1). This contradicts the fact that the cycle $\mathbf{c}$ contains v.

Forget that we fixed $\mathbf{c}$. We thus have obtained a contradiction for each cycle $\mathbf{c}$ of the graph (V_k, E_k, φ_k). Hence, the graph (V_k, E_k, φ_k) has no cycles. In other words, Claim 2 holds for $j = k$. This completes the induction step, and Claim 2 is proved.]

Applying Claim 2 to $j = i$, we see that the graph (V_i, E_i, φ_i) has no cycles. In other words, this graph is a forest. Since it is connected, it is therefore a tree. Since it is a spanning subgraph of G, we thus conclude that it is a spanning tree of G. Hence, we have constructed a spanning tree of G.

We note an important property of this construction:

Claim 3: For each $k \in \mathbb{N}$, we have

$$V_k = \{v \in V \mid d(r,v) \leq k\},$$

where $d(r,v)$ means the length of a shortest path from r to v.

This is easily proved by induction on k. Thus, the spanning tree (V_i, E_i, φ_i) we have constructed has the following property: For each $v \in V$, the path from r to v in this spanning tree is a shortest path from r to v in G. For this reason, this spanning tree is called a **breadth-first search ("BFS") tree**. Note that the choice of root r is important here: It is usually not true that the path from an arbitrary vertex u to an arbitrary vertex v along our spanning tree is a shortest path in G. No spanning tree of G has this property, unless G itself is "more or less a tree" (more precisely, unless G^{simp} is a tree)! □