Trees

Arash Rafiey

20 October, 2015

Definition

Let $G = (V, E)$ be a loop-free undirected graph. G is called a **tree** if G is connected and contains no cycle.

Trees

Definition

Let $G = (V, E)$ be a loop-free undirected graph. G is called a **tree** if G is connected and contains no cycle.

Theorem

If a, b are distinct vertices in tree $T = (V, E)$, then there is a unique path from a to b in T.

Trees

Definition

Let $G = (V, E)$ be a loop-free undirected graph. G is called a **tree** if G is connected and contains no cycle.

Theorem

If a, b are distinct vertices in tree $T = (V, E)$, then there is a unique path from a to b in T.

Proof.

Since T is connected, there is a path from a to b. If there is another path from a to b then we get a cycle by union the two paths. □

Theorem

In every tree $T = (V, E)$, $|V| = |E| + 1$.

Theorem

In every tree $T = (V, E)$, $|V| = |E| + 1$.

Proof.

We use induction on the number of edges. If $|E| = 0$ then it is clear that T has only one vertex. Let $e = uv$ be an edge in T. By removing e we would have two subtrees $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$. Observe that $|E_1| < |E|$ and $|E_2| < |E|$. By induction hypothesis $|T_1| = |E_1| + 1$ and $|T_2| = |E_2| + 1$. We have $|T| = |T_1| + |T_2|$ and $|E| = |E_1| + |E_2| + 1$. Therefore $|T| = |E| + 1$.

Theorem

Every tree $T = (V, E)$ with at least two vertices has at least two leaves (vertex of degree 1).

Theorem

Every tree $T = (V, E)$ with at least two vertices has at least two leaves (vertex of degree 1).

Proof.

Set $n = |T|$. We know that $|E| = n - 1$. If there are k leaves then $2(n-1) = 2|E| = \sum_{v \in V} deg(v) \geq k + 2(n-k)$. Therefore $k \geq 2$. □

Labeling Trees

A labeled tree with n vertices is a tree where each vertex has a label from $\{1, 2, \ldots, n\}$ (distinct labels).

The number of non-isomorphic trees with n labels is n^{n-2}.

Labeling Trees

A labeled tree with n vertices is a tree where each vertex has a label from $\{1, 2, \ldots, n\}$ (distinct labels).

The number of non-isomorphic trees with n labels is n^{n-2}.
This can be viewed as the number of sequences $x_1, x_2, \ldots, x_{n-2}$ where each $x_i \in \{1, 2, \ldots, n\}$.

Labeling Trees

A labeled tree with n vertices is a tree where each vertex has a label from $\{1, 2, \ldots, n\}$ (distinct labels).

The number of non-isomorphic trees with n labels is n^{n-2}.
This can be viewed as the number of sequences $x_1, x_2, \ldots, x_{n-2}$ where each $x_i \in \{1, 2, \ldots, n\}$.
From each labeling we obtain a sequence as follows.

1) Start with $i = 1$.

2) Set x_i to be the parent of the smallest labeled leaf y_j.

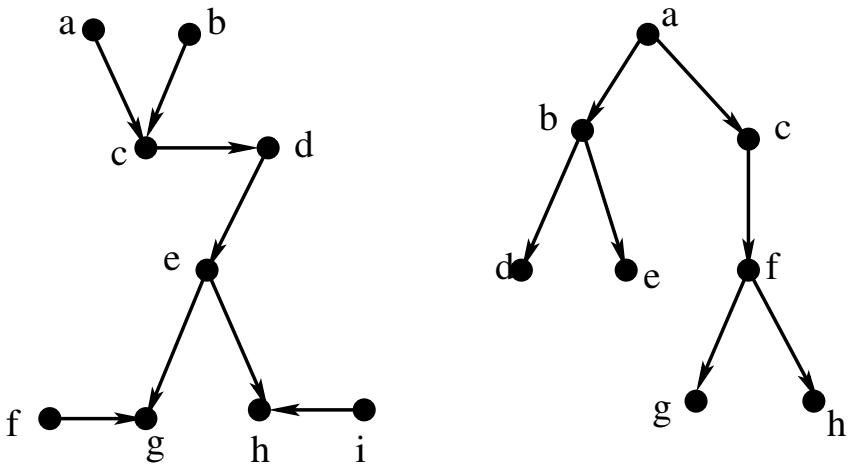
3) Remove the leaf and increase i by one. If $i = n - 2$ stops otherwise go to (2).

Definition

If G is a directed graph, then G is called a *directed tree* if the undirected graph associated with G is a tree. When G is a directed tree, G is called rooted tree if there is a unique vertex r, called *root* , in G with indegree zero, and for all other vertices v, the in degree of v is 1.

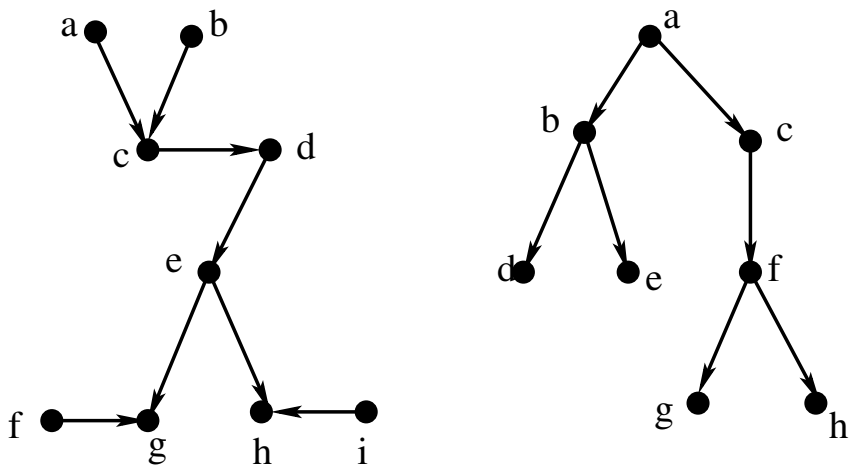
Definition

If G is a directed graph, then G is called a *directed tree* if the undirected graph associated with G is a tree. When G is a directed tree, G is called rooted tree if there is a unique vertex r, called *root* , in G with indegree zero, and for all other vertices v, the in degree of v is 1.



Definition

If G is a directed graph, then G is called a *directed tree* if the undirected graph associated with G is a tree. When G is a directed tree, G is called rooted tree if there is a unique vertex r, called *root* , in G with indegree zero, and for all other vertices v, the in degree of v is 1.



For simplicity we may ignore the direction once the root identified.

Traversing a tree starting from root

Breadth-First Search Method : Visiting the vertices of the tree level by level starting from root r

Breadth-First Search (BFS). Start with node r and set $L_0 = r$.

At step i let L_i be the set of nodes that are not in any of

$L_0, L_1, \ldots, L_{i-1}$ and have a neighbor in L_{i-1}.

Lemma

L_j is the set of nodes that are at distance exactly j from r.

Traversing a tree starting from root

Breadth-First Search Method : Visiting the vertices of the tree level by level starting from root r

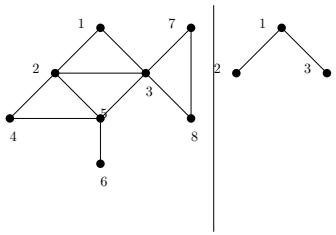
Breadth-First Search (BFS). Start with node r and set $L_0 = r$.

At step i let L_i be the set of nodes that are not in any of

$L_0, L_1, \ldots, L_{i-1}$ and have a neighbor in L_{i-1}.

Lemma

L_j is the set of nodes that are at distance exactly j from r.



Depth-First Search (backtracking approach)

We don't visit the nodes level by level! As long as there is an unvisited node adjacent to the current visited node we continue! Once we are stuck, trace back and go to a different branch!

Depth-First Search (backtracking approach)

We don't visit the nodes level by level! As long as there is an unvisited node adjacent to the current visited node we continue! Once we are stuck, trace back and go to a different branch!

PreorderDFS (r)

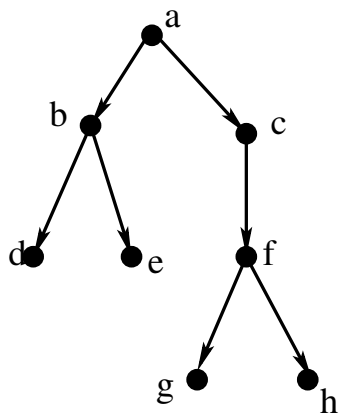
1. Mark r as Explored and print r.
2. For every edge rv
3. If v is not Explored then call PreorderDFS (v)

Depth-First Search (backtracking approach)

We don't visit the nodes level by level! As long as there is an unvisited node adjacent to the current visited node we continue! Once we are stuck, trace back and go to a different branch!

PreorderDFS (r)

1. Mark r as Explored and print r.
2. For every edge rv
3. If v is not Explored then call PreorderDFS (v)



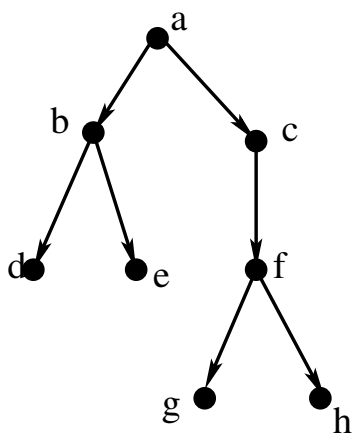
Pre–DFS : a, b, d, e, c, f, g, h

DFSpostorder (r)

1. Mark r as Explored
2. For every edge rv
3. If v is not Explored then call DFSpostorder (v)
4. Print r.

DFSpostorder (r)

1. Mark r as Explored
2. For every edge rv
3. If v is not Explored then call DFSpostorder (v)
4. Print r.



Post–DFS : d, e, b, g, h, f, c, a