CS 1010 Discrete Structures
Lecture 15:
Graphs Contd.

Maria Francis

February 22, 2021

22/02/2021

Recap

- Graphs – Matchings and Hall Marriage Problem
- Adjacency lists and adjacency matrices
- Graph isomorphisms – graph invariants
- Connectivity – paths, circuits and connected components

Vertex Connectivity

- Cut vertex is a vertex when on removal of it produces a subgraph that is disconnected.
- Vertex Cut is a set of vertices that make the graph disconnected.
- Vertex Connectivity $\kappa(G)$ is the minimum number of vertices in a vertex cut..
- Connected graphs without cut vertices are called nonseparable graphs – more connected than those with a cut vertex.
- k-connected graphs – i.e. $\kappa(G) \geq k$,
- Analogously, cut edge, Edge cuts and Edge connectivity $\lambda(G)$.

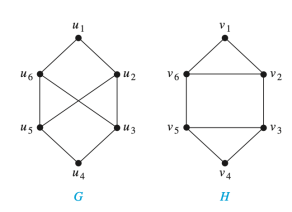
Connectivity in Directed Graphs

- Graph connectivity – key in reliability of networks, particularly in designing computer networks, highway planning.
- Minimum number of routers that can go down to disconnect the network and minimum intersections to block the roads for repair without disconnecting the highway.
- Connectedness in Directed Graphs - two notions.
- Strongly connected –If there is a path from a to b and from b to a whenever a and b are vertices in the graph.
- Weakly connected – if there is a path between every two vertices in the underlying undirected graph.
- Strongly connected $\Rightarrow$ weakly connected.

Paths and Graph Isomorphism

- Recall: Graph invariants help determine if two graphs are isomorphic.
- Paths/Circuits also aid the task of graph isomorphism - the existence of a simple circuit of a particular length is an invariant - Prove!
- Paths can also help us come up with the right mapping like degree did in the example we saw in the last lecture.

Example



- Both graphs have 6 vertices and 8 edges.
- Each has 4 vertices of degree 3 and two vertices of degree 2.
- Thus 3 graph invariants we know of – number of vertices, edges and degrees all agree.
- H has a simple circuit of length 3, v_1, v_2, v_6, v_1 whereas G has only simple circuits of length ≥ 4. so not isomorphic.

Number of Paths using its Adjacency Matrix

Theorem

Let G be a graph with adjacency matrix A w.r.t. the ordering $v_1, v_2, \ldots, v_n$ of the vertices of the graph (with directed or undirected edges, with multiple edges and loops allowed). The number of different paths of length r from v_i to v_j, where $r \in \mathbb{Z}_{\geq 0}$, equals the (i, j)th entry of A^r.

Proof by Induction

- Let G be a graph with the adjacency matrix A.
- The number of paths from v_i to v_j of length 1 is the (i, j)th entry of A - since it gives us the number of edges from v_i to v_j.
- Hypothesis : Assume that the (i, j) th entry of A^r is the number of different paths of length r from v_i to v_j.
- $A^{r+1} = A^r A$, therefore the (i, j)th entry of A^{r+1} equals

$$b_{i1}a_{1j} + b_{i2}a_{2j} + \cdots + b_{in}a_{nj},$$

  where b_{ik} is the (i, k) of A^r.
- By inductive hypothesis, b_{ik} – number of paths of length r from v_i to v_k.

Proof using induction

- A path of length $r+1$ from v_i to v_j – a path of length r from v_i to some intermediate vertex v_k , and then an edge from v_k to v_j.
- By product rule, that is the product of the number of paths of length r from v_i to v_k (b_{ik}) and the number of edges from v_k to v_j (a_{kj}).
- Then we need to add these products for all possible such v_ks.

Euler and Hamilton Paths

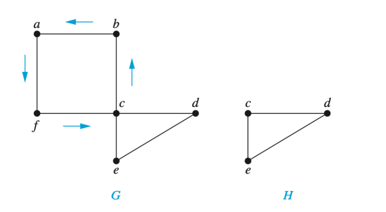
- Can we travel along the edges of a graph starting at a vertex and returning to it by traversing each edge of the graph exactly once? Euler circuit
- Can we travel along the edges of a graph starting at a vertex and returning to it while visiting each vertex of the graph exactly once? Hamilton Circuit
- Looks similar but Euler circuits can be answered by looking at degrees of vertices.
- Hamilton Circuit – quite difficult to solve for most graphs!
- They have many practical applications but they are actually old puzzles!

Euler Paths and Circuits

- An Euler circuit in a graph G is a simple circuit containing every edge of G.
- An Euler path in G is a simple path containing every edge of G.
- Lets find a necessary condition.
- If Euler circuit begins with a vertex a then it leaves with an edge incident with a, (a, b) and it is a different edge.
- Each time the circuit passes through a vertex it contributes 2 to the vertex's degree – enters via one edge and leaves by another.
- As for a – the circuit terminates where it started, contributing one to $deg(a)$, making a of even degree.
- if a connected graph has an Euler circuit, then every vertex must have even degree.

Sufficient Condition

- Is the same condition sufficient too? – Must an Euler circuit exist in a connected multigraph if all vertices have even degree?
- Explanation with an example:



Sufficient Condition

Theorem

A connected multigraph with at least two vertices has an Euler circuit if and only if each of its vertices has even degree.

- You can always have an algorithm that computes the Euler circuit from the preceding discussion on how to construct circuits in G and H.
- It is efficient - $O(m)$, m is the number of edges in G.
- Fleury's algorithm - another algorithm for computing circuits.

Characterization for Euler Path

Theorem

A connected multigraph has an Euler path but not an Euler circuit if and only if it has exactly two vertices of odd degree.

- Suppose that a connected multigraph does have an Euler path from a to b, but not an Euler circuit.
- The first edge of the path contributes one to the degree of a.
- A contribution of two to the degree of a is made every time the path passes through a.
- The last edge in the path contributes one to the degree of b. Every time the path goes through b there is a two to its degree.
- Both a and b have odd degree.
- Every other vertex has even degree!

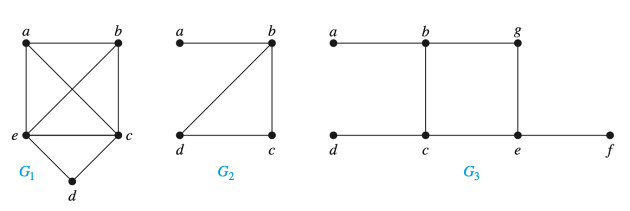
Characterization for Euler Path

- Converse: Suppose that a graph has exactly two vertices of odd degree, say a and b.
- Consider the larger graph made up of the original graph with the addition of an edge $\{a, b\}$.
- Every vertex of this larger graph has even degree, so there is an Euler circuit.
- The removal of the new edge produces an Euler path in the original graph.

Hamilton Paths and Circuits

- A simple path in a graph G that passes through every vertex exactly once is called a Hamilton path.
- A simple circuit in a graph G that passes through every vertex exactly once is called a Hamilton circuit.
- Icosian puzzle, invented in 1857 by the Irish mathematician Sir William Rowan Hamilton.

Examples

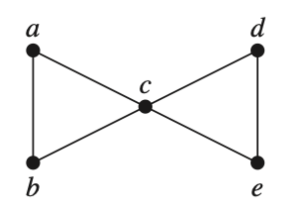


- G_1 has a Hamiltonian circuit, a, b, c, d, e, a.
- G_2 has no Hamiltonian circuit but has a Hamiltonian path a, b, c, d.
- G_3 does not have a Hamiltonian circuit or path.

Characterization possible?

- Is there a simple way to determine whether a graph has a Hamilton circuit or path? No
- There are results for sufficient conditions for the existence of Hamilton circuits.
- Certain properties can be used to show that a graph has no Hamilton circuit.
  - A graph with a vertex of degree one cannot have a Hamilton circuit.
  - If a vertex in the graph has degree two, then both edges that are incident with this vertex must be part of any Hamilton circuit.
  - Once you have considered a vertex and two edges incident on it for a circuit then you can remove all the other edges incident on the vertex.
  - Hamilton circuit cannot contain a smaller circuit within it.

Characterization possible?



- Here vertex a, b, d, e have degree 2 and their edges have to be included.
- But that means c will have to included 4 times, therefore not possible.

Some (Sufficient) Results associated with Hamilton Circuits/Paths

- Show that K_n has a Hamilton circuit whenever $n \geq 3$.
- More edges a graph has, the more likely it is to have a Hamilton circuit.

Theorem (Dirac's)

If G is a simple graph with n vertices $n \geq 3$, s.t. the deg of every vertex is at least $n/2$ then G has a Hamilton circuit.

Theorem (Ore's)

If G is a simple graph with n vertices $n \geq 3$, s.t. $deg(u) + deg(v) \geq n$ for every pair of nonadjacent vertices u and v in G then G has a Hamilton circuit.

Some Observations associated with Hamilton Circuits/Paths

- Dirac's theorem is a corollary to Ore's theorem.
- Not necessary conditions - C_5 has a Hamilton circuit but does not satisfy either of the above results.
- The best algorithms known for finding a Hamilton circuit in a graph or determining that no such circuit exists have exponential worst-case time complexity (in the number of vertices of the graph).
- Finding an algorithm that solves this problem with polynomial worst-case time? - Big deal since the problem is NP-complete!
- Application of Hamilton Circuits – traveling salesperson problem or TSP! shortest route a traveling salesperson should take to visit a set of cities.

Planar Graphs

- The motivation – to design connections where the connecting edges (wires?) do not criss cross each other.
- Examples – circuit boards with no overlapping wires, utility connections in house with no overlapping.

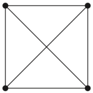


FIGURE 2 The Graph K_4.

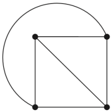


FIGURE 3 K_4 Drawn with No Crossings.

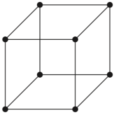


FIGURE 4 The Graph Q_3.

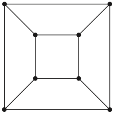


FIGURE 5 A Planar Representation of Q_3.

Planar Graphs

- A graph is called planar if it can be drawn in the plane without any edges crossing (where a crossing of edges is the intersection of the lines or arcs representing them at a point other than their common endpoint).
- Such a drawing is called planar representation of the graph.
- We just saw K_4 and Q_3 are planar. What about $K_{3,3}$ and K_5? No.
- We can use arguments that look at regions divided and then argue that they cannot be planar.

Euler's Formula

A planar representation of a graph splits the plane into regions, including an unbounded region.

Theorem

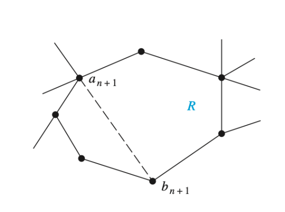
Let G be a connected planar simple graph with e edges and v vertices. Let r be the number of regions in a planar representation of G. Then $r = e - v + 2$.

- First we specify a planar representation of G.
- We construct a sequence of subgraphs $G_1, G_2, \ldots, G_e = G$ by adding an edge at each stage (it has to be incident with a vertex in G_{n-1}).
- Possible because G is connected.
- G_1 will have one edge (some arbitrary edge!)
- Let r_n, e_n, v_n represent the number of regions, edges, and vertices of the planar representation of G_n.

Euler's Formula - Proof by induction

- $r_1 = e_1 - v_1 + 2$ is true for G_1 since $e_1 = r_1 = 1$ and $v_2 = 2$.
- Assume $r_k = e_k - v_k + 2$.
- Let $\{a_{k+1}, b_{k+1}\}$ be the edge that is added to G_k to obtain G_{k+1}.
- Case i: The vertices were already in G_k.
- They must have been on the boundary of a common region R or else $\{a_{k+1}, b_{k+1}\}$ would caused crossing!
- The new edge splits R into two regions $\Rightarrow r_{k+1} = r_k + 1$, $e_{k+1} = e_k + 1$, $v_{k+1} = v_k$ - formula holds!

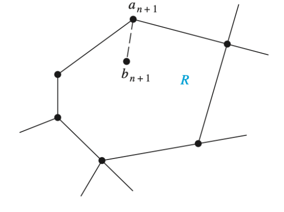
Euler's Formula - Case i



Euler's Formula - Proof by induction

- Case (ii) : One of the two vertices is not already in G_k.
- Assume a_{k+1} is in G_k and b_{k+1} is not.
- Adding this new edge does not produce new regions – b_{k+1} must be in a region that has a_{k+1} on its boundary.
- $r_{k+1} = r_k$, $e_{k+1} = e_k + 1$ and $v_{k+1} = v_k + 1$ - formula still holds
- Completes the induction argument!

Euler's Formula - Case ii



Euler's Formula - Very Important Corollaries

Corollary

If G is a connected planar simple graph with e edges and v vertices, where $v \geq 3$, then $e \leq 3v - 6$.

Corollary

If G is a connected planar simple graph then G has a vertex of degree not exceeding five.

Corollary

If a connected planar simple graph has e edges and v vertices with $v \geq 3$ and no circuits of length 3 then $e \leq 2v - 4$.

Euler's Formula - Very Important Corollaries

- Second corollary is easy to prove. It is used to prove Corollary 1. (Try!)
- First corollary shows that K_5 is nonplanar - It has five vertices and ten edges. $e \leq 3v - 6$ is not satisfied!
- $K_{3,3}$ satisfies Corollary 1 but it is not a sufficient condition – it violates Corollary 3. $e = 9, 2v - 4 = 8$

Idea of Graph coloring

- Coloring of maps - regions with common border different colors.
- One way - a new color every time! - Very inefficient! We would prefer smaller number of colors!
- Consider the problem of determining the least number of colors that can be used to color a map so that adjacent regions never have the same color.
- A coloring of a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices are assigned the same color.
- The chromatic number $\chi(G)$ of a graph is the least number of colors needed for a coloring of this graph.

Four Color Theorem

- Asking for chromatic number of planar graph - same as asking for the minimum number of colors required to color a planar map so that no two adjacent regions are assigned the same color.
- An old question - more than 100 years old!

Theorem (Four Color Theorem)

The chromatic number of a planar graph is no greater than four.

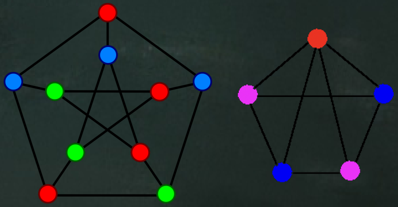
- Posed as a conjecture in 1850s.
- Many incorrect proofs were published, many with hard to find errors that they were thought to be true!
- Finally proved in 1976 - Appel and Haken.

Four Color Theorem

- The proof uses case-by-case analysis by computer.
- First assume the theorem is false, then show that there can be ≈ 2000 types of possible counterexamples.
- Then they show none of these types exist.
- Simpler proofs have come but everything relies on computer.
- Doesn't work for nonplanar graphs – they can have large chromatic number.

Coloring of graphs

- To show chromatic number is k – first color with k colors and then show it cannot be colored with fewer than k colors.
- Examples –



Coloring of graphs

- Chromatic color of K_n –
  - It can be colored by n colors.
  - Can there be fewer colors? - No! since every vertex is connected to every other vertex and therefore needs a different color.
  - $\chi(K_n) = n$.
  - Note K_n is not planar for $n \geq 5$ so four color theorem holds!
- What is the chromatic number of the complete bipartite graph $K_{m,n}$?
- $\chi(K_{m,n}) = 2$ since its bipartite.

Chromatic Numbers

- Chromatic number of cycle C_n, $n \geq 3$
  - Consider C_n when n is even.
  - Pick a vertex, color it say with red.
  - Proceed along in a clockwise direction with a planar representation of the graph – coloring the second vertex blue, the third vertex red and so on.
  - The first and $n-1$ vertices are both colored red and therefore nth vertex can be colored blue.
  - What if n is odd? First and $n-1$st vertex are of different colors - red and blue and therefore nth vertex has to be a third color.
  - $\chi(C_n) = 2$ if n is an even positive integer, $\chi(C_n) = 3$, if n is odd, $n \geq 3$.

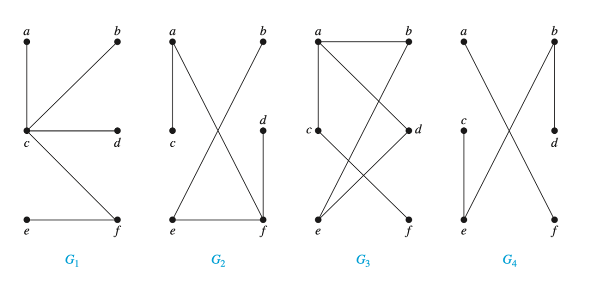
Algorithms for finding Chromatic Numbers

- The best algorithms have exponential worst-case time complexity (in the number of vertices of the graph).
- Finding an approximation to the chromatic number of a graph is difficult.
- Applications - scheduler of exams, no students have two exams at the same time.
- Frequency assignments for TV stations so that within a certain distance two stations wont have the same channel.
- Compilers - assigning index registers for a loop.

Trees

- Tree – A connected graph that contains no simple circuits.
- Applications – chemistry (first use of trees as far back as 1857), computer science – for efficient algorithms for example to locate items in a list, coding theory like Huffman coding, gives efficients codes that save costs of data transmission and storage understand outcomes of games.
- Very important in modeling procedures carrying out sequence of decisions.
- What about graphs containing no simple circuits that are not necessarily connected? forests
- Each of forest's connected components is a tree.

Examples and Non-Examples



Trees

An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.

- Assume that T is a tree. By definition T is a connected graph with no simple circuits.
- Let x, y be two vertices of T, then there is a simple path between x and y since T is connected.
- If the path is not unique then by combining first path from x to y followed by the path from y to x (reverse of the path).
- That is a circuit - not possible, so only a unique simple path is possible.
- Assume there is a unique simple path between any two vertices of a graph T.
- This implies T is connected.

Trees

- All that remains to show that T can have no simple circuits.
- Suppose T had a simple circuit containing x and y.
- Then there would be two simple paths between x and y – that would violate the unique simple path between any two vertices.

Rooted Trees

- A particular vertex is designated as root.
- Every edge is given a direction from root.
- There is after all a unique path from the root to each vertex of the graph.

A rooted tree is a tree in which one vertex has been designated as the root and every edge is directed away from the root.
A different choice of root leads to a different rooted tree.

Terminology of Trees

- Botanical and genealogical origins.
- T is a rooted tree. v is a vertex in T other than the root, the parent of v is the unique vertex u s.t. there is a directed edge from u to v. v is called a child of u.
- Vertices with same parent - siblings.
- The ancestors of a vertex (other than root) are the vertices in the path from the root to this vertex, excluding the vertex itself and including the root.
- The descendants of a vertex v are those vertices that have v as an ancestor.
- A vertex of a rooted tree is called a leaf if it has no children.
- Vertices that have children are called internal vertices. The root is an internal vertex unless it is the only vertex in the graph, in which case it is a leaf.

m-ary Trees and Ordered Trees

- A rooted tree is called an *m-ary tree* if every internal vertex has no more than m children.
- The tree is called a full *m-ary tree* if every internal vertex has exactly m children.
- An m-ary tree with $m = 2$ is called a binary tree.
- Ordered Rooted Tree : is a rooted tree where the children of each internal vertex are ordered.
- Ordered rooted trees are drawn so that the children of each internal vertex are shown in order from left to right.
- Ordered binary tree or just binary tree : if an internal vertex has two children, the first child is called the left child and the second child is called the right child .
- Similarly, left subtree and right subtree.

Properties of Trees

A tree with n vertices has $n-1$ edges.

- Proof by induction. Basis Step. When $n=1$ a tree with $n=1$ vertex has no edges.
- Inductive step . The inductive hypothesis states that every tree with k vertices has $k-1$ edges, where $k \in \mathbb{Z}_{\geq 0}$.
- Suppose that a tree T has $k+1$ vertices and that v is a leaf of T (exists because the tree is finite).
- Let w be the parent of v.
- Removing from T the vertex v and $\{w, v\}$ produces a tree $T^{'}$ with k vertices.
- $T^{'}$ has $k-1$ edges by hypothesis.
- T will have k edges since it includes the edge connecting v and w.

Properties of Trees

Tree is a connected undirected graph with no simple circuits. This means, consider when G is an undirected graph with n vertices,

1. G is connected,
2. G has no simple circuits,
3. G has $n-1$ edges.

From previous theorem we have *i* and *ii* implies *iii*.

Exercise:

1. When *i* and *iii* hold, this implies *ii* holds.
2. When *ii* and *iii* hold, *i* must hold.

Counting vertices in m-ary trees

- A full m-ary tree with i internal vertices contains $n = mi + 1$ vertices. Proof: Root is counted in $+1$.
- A full m-ary tree with
  1. n vertices has $i = (n-1)/m$ internal vertices and $l = [(m-1)n + 1]/m$ leaves,
  2. i internal vertices $n = mi + 1$ vertices and $l = (m-1)i + 1$ leaves,
  3. l leaves has $n = (ml - 1)/(m-1)$ vertices and $i = (l-1)/(m-1)$ internal vertices.
- All of it can be solved by $n = mi + 1$ and $n = l + i$. For eg : in 1, $i = (n-1)/m$ from $n = mi + 1$, insert this in $n = l + i$ to get $l = [(m-1)n + 1]/m$.

Exercise Question

Suppose that someone starts a chain letter. Each person who receives the letter is asked to send it on to 4 other people. Some people do this, but others do not send any letters. How many people have seen the letter, including the first person, if no one receives $\gneq$ 1 letter and if the chain letter ends after there have been 100 people who read it but did not send it out? How many people sent out the letter?

- We can use a 4-ary tree.
- Internal vertices - people who sent out the letter, leaves- people who didn't.
- No of leaves $l = 100$.
- From previous result we have $n = (4 \cdot 100 - 1)/(4 - 1) = 133$ - these many people saw the letter
- Number of internal vertices is $133 - 100 = 33$ - they sent out the letter.

Terminology related to m-ary trees

- Level of a vertex v in a rooted tree – length of the unique path from the root to this vertex.
- Level of the root – 0.
- Height of a rooted tree – maximum of levels, i.e. length of the longest path from the root to any vertex.
- Balanced - A rooted m-ary tree of height h is balanced if all leaves at levels h or $h-1$.

Theorem

There are at most m^h leaves in an m-ary tree of height h.

Proof by induction on the height of the tree.

m-ary trees

Corollary

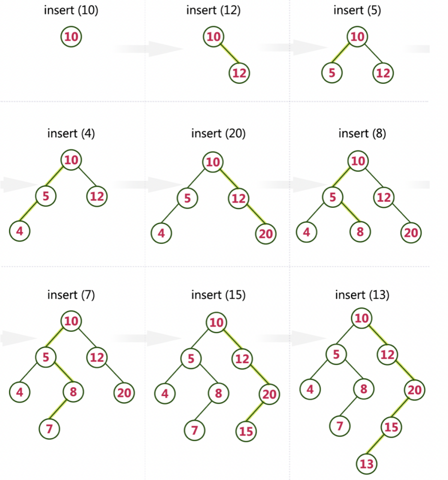
If an m-ary tree of height h has l leaves, then $h \geq \lceil log_m l \rceil$. If the m-ary tree is full and balanced, then $h = \lceil log_m l \rceil$.

- $l \leq m^h$ from previous theorem.
- $log_m l \leq h$, h is an integer, $h \geq \lceil log_m l \rceil$.
- Suppose that the tree is balanced, each leaf is at level h or $h - 1$.
- The height of the tree is h there is at least one leaf is at level h.
- There are therefore more than m^{h-1} leaves.
- We have $m^{h-1} \lneq l \leq m^h$, taking log $h - 1 \lneq log_m l \leq h \Rightarrow h = \lceil log_m l \rceil$.

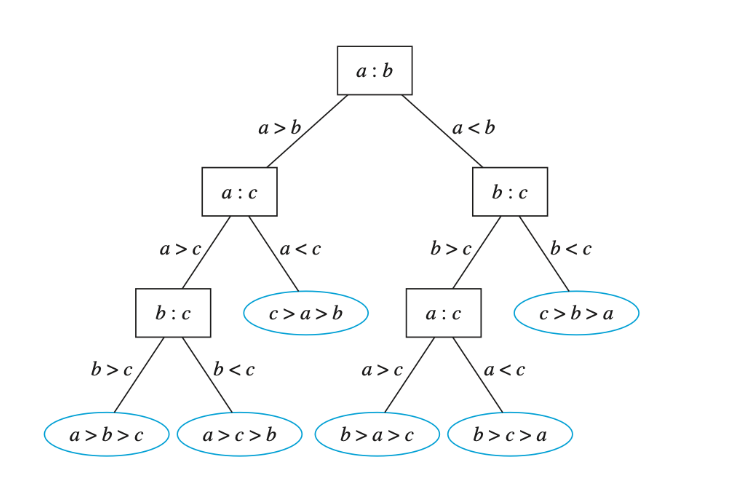
Applications of Trees - Binary Search Tree (BST)

- Searching for items in a list - an important task in computer science.
- Each child of a vertex is designated a right or left child.
- Each vertex is assigned a key – key of a vertex is both larger than the keys of all vertices in its left subtree and smaller than the keys of all vertices in its right subtree.

Applications of Trees - Binary Search Tree (BST)



Decision Trees



Complexity of Comparison based sorting algorithms

- Using decisions trees we can find a lower bound for the worst-case complexity of sorting algorithms.
- Given a list of n elements sorting algorithms are based on binary comparisons.
- A binary decision tree in which each internal vertex represents a comparison of two elements.
- Each leaf represents one of the $n!$ permutations of n elements.
- Complexity is based on number of binary comparisons, worst case complexity is based on largest number of binary comparisons needed to sort a list with n elements.
- That is the height of the decision tree with $n!$ leaves - at least $\lceil log n! \rceil$

Complexity of Comparison based sorting algorithms

Theorem

A sorting algorithm based on binary comparisons requires at least $\lceil log n! \rceil$ comparisons.

Exercise : $\lceil log n! \rceil$ is $\Theta(nlogn)$.

Therefore we have,

Theorem

The number of comparisons used by a sorting algorithm to sort n elements based on binary comparisons is $\Omega(nlogn)$.

So if you have a comparison sorting algorithm that uses $\Theta(nlogn)$ comparisons in the worst case you have an optimal algorithm.

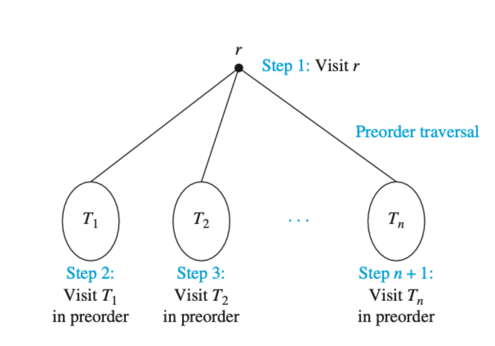
Tree Traversal

- Ordered rooted trees are used to store information and therefore we need procedures for visiting each vertex.
- Traversal algorithms – preorder, inorder and postorder traversal.

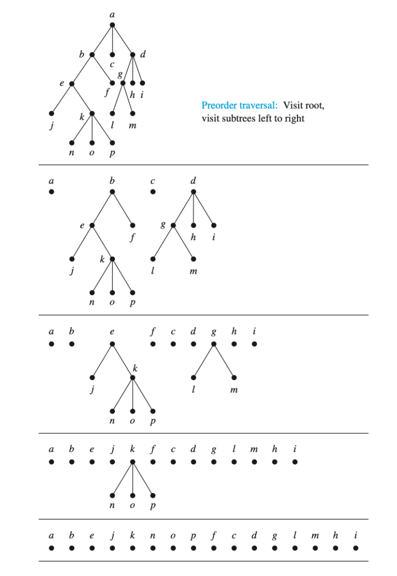
Definition (Preorder Traversal)

Let T be an ordered rooted tree with root r. If T contains only of r, then r is the preorder traversal of T. Or else suppose $T_1, T_2, \ldots, T_n$ are the subtrees at r from left to right in T. The preorder traversal begins by vising r, continues by traversing T_1 in preorder, then T_2 in preorder and so on until T_n is traversed in preorder.

Preorder Traversal



Example

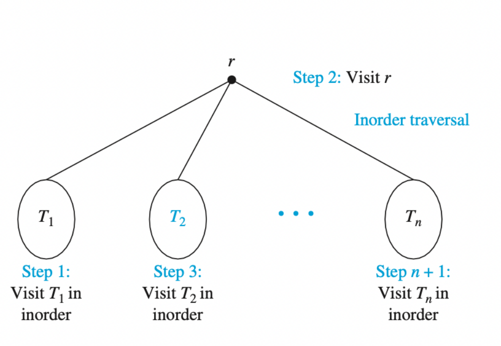

Preorder traversal: Visit root,
visit subtrees left to right
a b e j k n o p f c d g l m h i

Tree Traversal

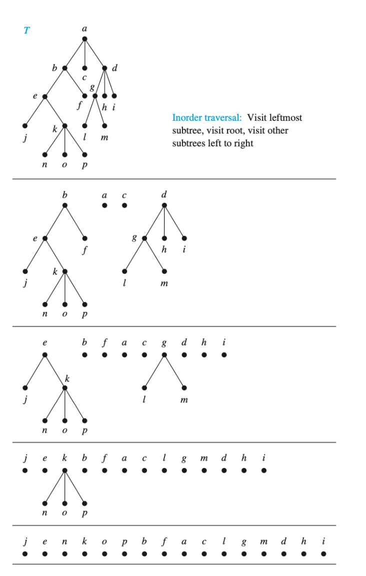
Definition (Inorder Traversal)

Let T be an ordered rooted tree with root r. If T contains only of r, then r is the inorder traversal of T. Or else suppose $T_1, T_2, \ldots, T_n$ are the subtrees at r from left to right in T. The inorder traversal begins by traversing T_1 in inorder, then visiting r, continues by traversing T_2 in inorder, then T_3 in inorder and so on until T_n is traversed in inorder.

Inorder Traversal



Example


T
Inorder traversal: Visit leftmost subtree, visit root, visit other subtrees left to right
j e n k o p b f a c l g m d h i

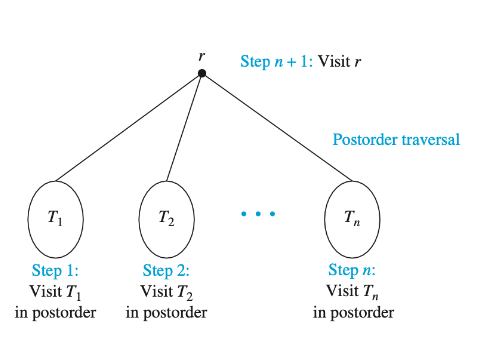
Tree Traversal

Definition (Postorder Traversal)

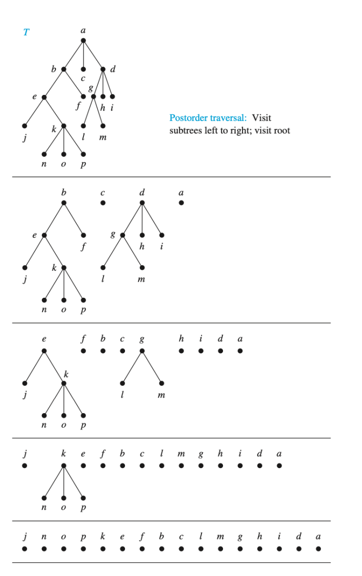
Let T be an ordered rooted tree with root r. If T contains only of r, then r is the postorder traversal of T. Or else suppose $T_1, T_2, \ldots, T_n$ are the subtrees at r from left to right in T. The postorder traversal begins by traversing T_1 in postorder, continues by traversing T_2 in postorder, then T_3 in postorder and so on then T_n is traversed in postorder and finally ends by visiting r,.

- Exercise – Design recursive algorithms for these traversals.
- Inorder traversal of a BST gives ——-.

Postorder Traversal



Example


T
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
Postorder traversal: Visit subtrees left to right; visit root
j n o p k e f b c l m g h i d a