Lec 23:
Graphs and Trees V

Prof. Adam J. Aviv

GW

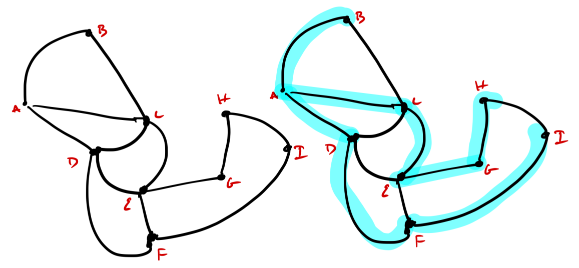
CSCI 1311 Discrete Structures I
Spring 2023

Spanning Trees and Traversals

Spanning Tree

Definition

A spanning tree of a graph G is a subgraph of G that contains ever vertex of G and is a tree.



Every connected graph has a spanning tree

Existence of Spanning Tree Proof (by Algorithm!).

Consider a graph G that is connected.

- If it is acyclic, then it is a spanning tree.
- If it has cycles/circuits, then by the Lemma (from before) we can remove one edge from the circuit (breaking the circuit) to produce a graph G' that is still connected.
  - If G' is acyclic, then it is spanning tree.
  - If G' has cycles, repeatedly remove an edge from each circuit until G'' is reached that is acyclic (which must occur). G'' is a spanning tree

In all cases, a spanning tree can be found.

Traversal algorithms for finding a spanning tree

As a spanning tree contains all the vertexes, we sometimes define the routine for identifying the spanning a tree as a *traversal* of the vertices. The order in which the vertices are enumerated defines the traversal (and the spanning tree),

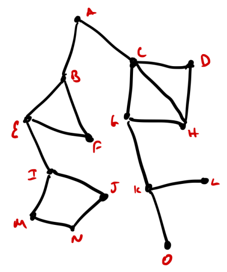
There are two important traversal algorithms that appear frequently in computer science, each starting from a root/start vertex.

- Depth-First-Search (DFS) Traversal
  - Explore entire sub-tree before exploring the next sub-tree
- Breadth-First-Search (BFS) Traversal
  - Explore each level of the tree completely before exploring the next level.

Depth-First Search (1)

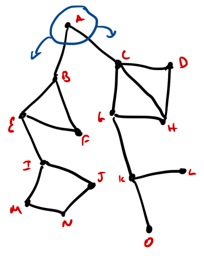
From a start/root vertex, explore the entire sub-tree of the spanning tree first before exploring the next sub-tree

Example: DFS on the following graph starting with *A*. We will break ties by choosing edges that connect to vertices in alphabetic order.

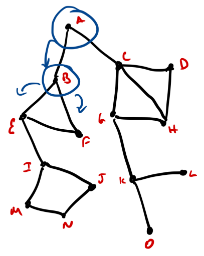


Depth-First Search (2)

Starting at A, we could explore the sub-tree with roots B or C: choose B because of alphabetic ordering.

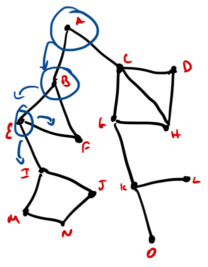


At B, we could explore the sub-tree with roots E or F: choose E because of alphabetic ordering.

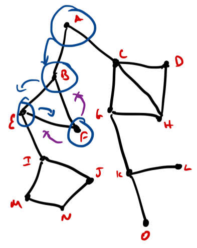


Depth-First Search (3)

At E, the choice is between F and I: choose F.



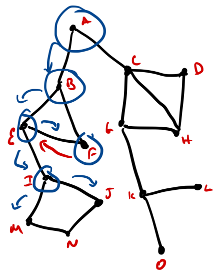
At F, we are stuck. Both B and E have been enumerated/visited.



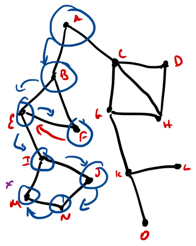
This sub-tree has been full explored, so we back up to E.

Depth-First Search (4)

At I, we choose J over M.



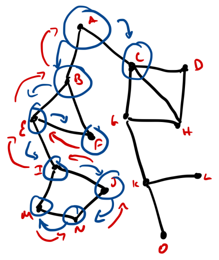
Continuing from J, we choose N. And then are stuck again.



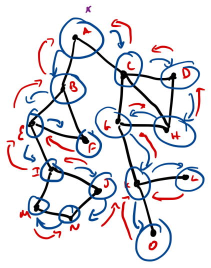
When we back up this time, it takes us all the way to A.

Depth-First Search (5)

From A, we explore the sub-tree with sub-root C.



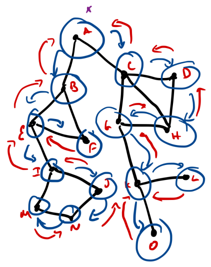
We explore the entire sub-tree, and eventually getting stuck at 0.



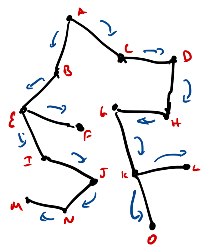
When we back up this time, it takes us all the way to A. But, there is no where else to go. We are done!

Depth-First Search (6)

We've embedded the traversal in the graph, with arrows.



Following the arrows, we get the DFS spanning tree

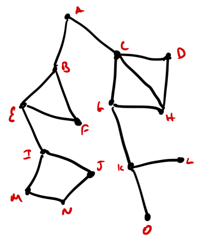


The order of the traversal is the order in which we visited the vertexes in forming the tree.

$$A, B, E, F, I, J, N, M, C, D, H, G, K, L, O$$

Exercise

Find the DFS spanning trees starting with E and H.

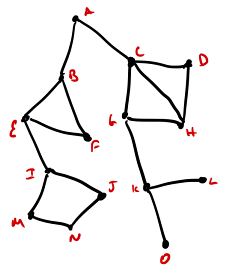


What is the order of each traversal.

Breadth-First Search (1)

From a start/root vertex, explore the next level of the spanning tree before exploring the following level.

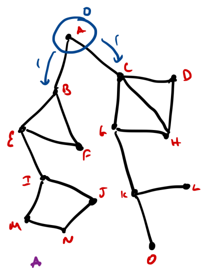
Example: BFS on the following graph starting with *A*. We will break ties by choosing edges that connect to vertices in alphabetic order.

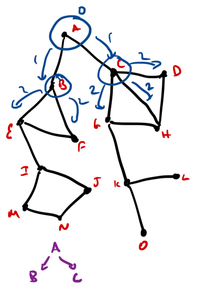


Breadth-First Search (2)

Starting with *A* (at level 0), the next level of the tree would include *B* and *C*.

At *B* and *C*, the next level (level 2), would include *E*, *F*, *G*, *D*, and *H*

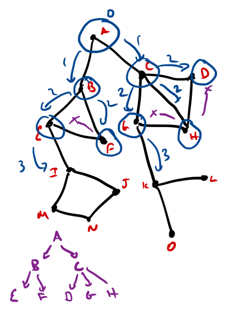




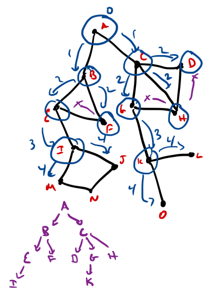
Ties are broken by Alphabetic order. We first visit *B* and then *C*, left to right across level 1 in the spanning tree.

Breadth-First Search (3)

Exploration F, D and H are stuck as all children have been visited.



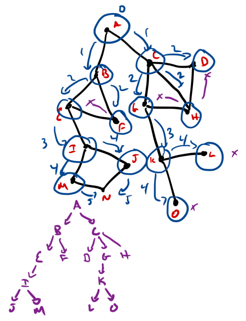
We continue to explore the next level from the children of E and G.



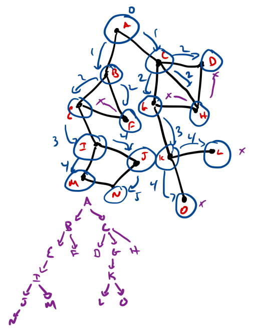
Ties are still broken by alphabetic ordering across the level of the spanning tree.

Breadth-First Search (4)

Continuing on the next level, L and O are stuck

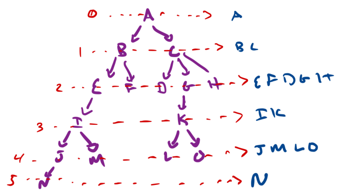


Since N can be reached by both M and J, we break the tie alphabetically, and N is a child of J.



Breadth-First Search (5)

We've tracked the spanning tree during the traversal.

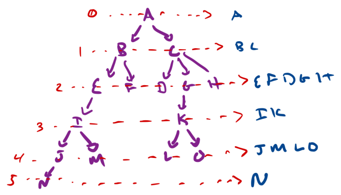


The traversal order is now a *level-order traversal* of the spanning tree, where each level is enumerated left-to-right, top-to-bottom.

$$A, B, C, E, F, D, G, H, I, I, K, J, M, L, O, N$$

Shortest Path and BFS

The spanning tree for BFS is also a minimum spanning tree, which defines the smallest distance (in terms of number of edges in the path) between the root and other vertices.

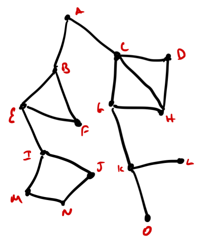


For example, the distance between A and J is 4 "hops" as j is on level 4 of the tree.

Why does BFS define the minimum spanning tree?

Exercise

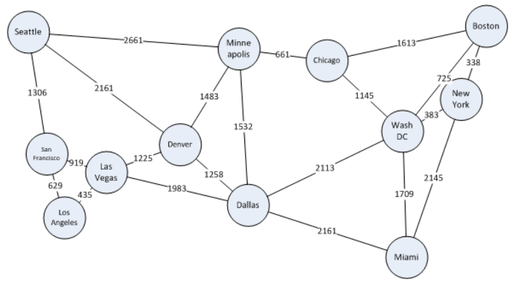
Find the BFS spanning trees starting with E and H.



What is the order of each traversal.

Minimum Spanning Trees with Weighted Edges

If a graph have weighted edges, a minimum spanning tree (MST) is the spanning tree with minimum total weight.



For example, a weighted graph with distances between cities in the USA, the (MST) from Washington would define the shortest path via other connecting cities.

`http://hansolav.net/sql/graphs.html`

Solving shortest path problems

Solutions for the shortest path in a graph (or network) is extremely important to computer science. There are number of seminal algorithms.

- Dijkstra's Algorithm
- Prim's Algorithm
- Kruskal's Algorithm

The book discusses each of these in detail, but they will likely be covered in your Algorithms or Computer Network classes.