DISCRETE STRUCTURE LECTURE NOTES-5

BSc.(H) Computer Science: II Semester

Teacher: Ms. Sonal Linda

SPANNING TREES

Definition

Let G be a simple graph. A spanning tree of G is a subgraph of G that is a tree containing every vertex of G.

Theorem 1: A simple graph is connected if and only if it has a spanning tree.

Example 1: Find a spanning tree of the simple graph G shown in Figure 1.

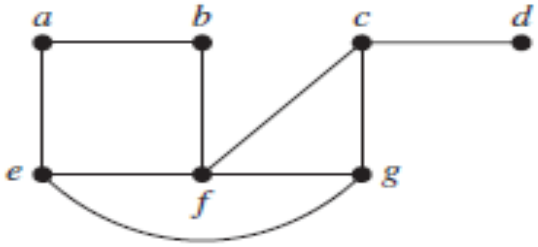


FIGURE 1 The Simple Graph G.

Solution: The graph G is connected, but it is not a tree because it contains simple circuits.

- Remove the edge $\{a, e\}$. This eliminates one simple circuit, and the resulting subgraph is still connected and still contains every vertex of G.
- Next, remove the edge $\{e, f\}$ to eliminate a second simple circuit.
- Finally, remove edge $\{c, g\}$ to produce a simple graph with no simple circuits.

This subgraph is a spanning tree, because it is a tree that contains every vertex of G. The sequence of edge removals used to produce the spanning tree is illustrated in Figure 2.

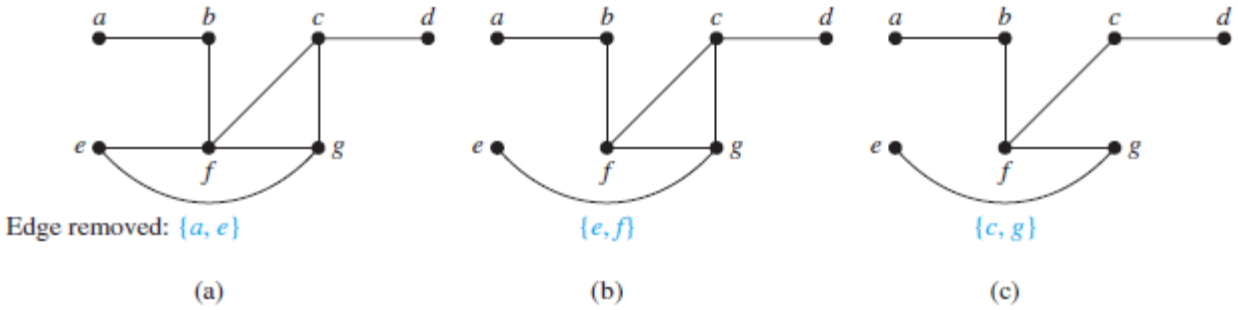


FIGURE 2 Producing a Spanning Tree for G by Removing Edges That Form Simple Circuits.

Spanning Tree

- The algorithm for finding spanning trees by removing edges from simple circuits is inefficient because it requires that simple circuits be identified.

- Instead of constructing spanning trees by removing edges, spanning trees can be built up by successively adding edges.

- Two algorithms based on the above principle are:
  - Depth First Search (DFS)
  - Breadth First Search (BFS)

Depth First Search

ALGORITHM 1 Depth-First Search.

procedure *DFS*(*G*: connected graph with vertices $v_1, v_2, \ldots, v_n$)
T := tree consisting only of the vertex v_1
visit(v_1)

procedure *visit*(*v*: vertex of *G*)
for each vertex *w* adjacent to *v* and not yet in *T*
    add vertex *w* and edge {*v*, *w*} to *T*
    visit(*w*)

Depth First Search

Steps:

1. Arbitrarily choose a vertex of the graph as the root.

2. Form a path starting at this vertex (root) by successively adding vertices and edges, where each new edge is incident with the last vertex in the path and a vertex not already in the path.

3. Continue adding vertices and edges to this path as long as possible.
   - If the path goes through all vertices of the graph, the tree consisting of this path is a spanning tree.
   - If the path does not go through all vertices, more vertices and edges must be added.

Depth First Search

Steps:

4. Move back to the next to level vertex in the path:
   - If possible form a new path starting at this vertex passing through vertices that were not already visited.
   - If not possible, move back another vertex in the path that is two vertices back in the path and try again.

5. Repeat this procedure,
   - Beginning at the last vertex
   - Moving back up the path one vertex at a time
   - Forming new paths that are as long as possible until no more edges can be added.

Depth First Search

Example1: Use depth-first search to find a spanning tree for the graph G as shown in Figure 1.

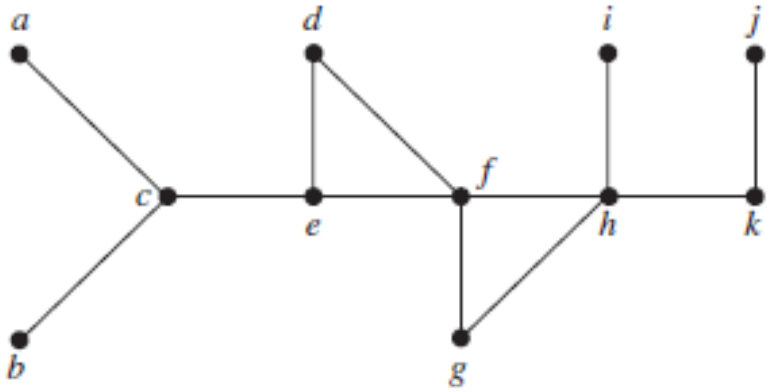


FIGURE 1 The Graph G.

Depth First Search

Solution: The steps used by depth first search to produce a spanning tree of G are shown in Figure 2.

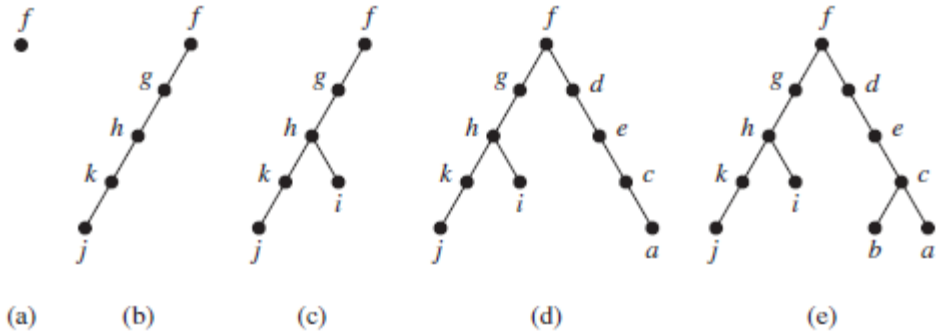


FIGURE 2 **Depth-First Search of G.**

- We arbitrarily start with the vertex f. A path is built by successively adding edges incident with vertices not already in the path, as long as this is possible. This produces a path f, g, h, k, j.
- Next, backtrack to k. There is no path beginning at k containing vertices not already visited. So, we backtrack to h. Form the path h, i.
- Then backtrack to h and then f.From f build the path f, d, e, c, a.
- Then backtrack to c and form the path c, b. This produces the spanning tree.

Breadth First Search

ALGORITHM 2 Breadth-First Search.

procedure *BFS* (*G*: connected graph with vertices $v_1, v_2, \ldots, v_n$)
T := tree consisting only of vertex v_1
L := empty list
put v_1 in the list L of unprocessed vertices
while L is not empty
    remove the first vertex, v, from L
    for each neighbor w of v
        if w is not in L and not in T **then**
            add w to the end of the list L
            add w and edge $\{v, w\}$ to T

Breadth First Search

Steps:

1. Arbitrarily choose a vertex of the graph as the root.

2. Then add all edges incident to the vertex (root).

3. The new vertices added at this stage become the vertices at level 1 in the spanning tree. Arbitrarily order them.

4. Next, for each vertex at level 1, visited in order, add each edge incident to this vertex to the tree as long as it does not produce a simple circuit.

Breadth First Search

Steps:

5. Arbitrarily order the children of each vertex at level 1. This produces the vertices at level 2 in the tree.

6. Follow the same procedure until all the vertices in the tree have been added.

7. The procedure ends because there are only a finite number of edges in the graph.

A spanning tree is produced because we have produced a tree containing every vertex of the graph.

Breadth First Search

Example 2: Use breadth first search to find spanning tree for the graph shown in Figure 3.

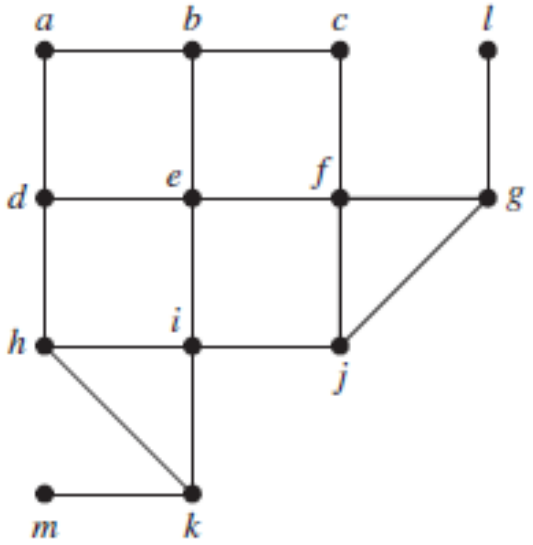


FIGURE 3 A Graph G.

Breadth First Search

Solution: The steps of the breadth-first search procedure are shown in Figure 4.

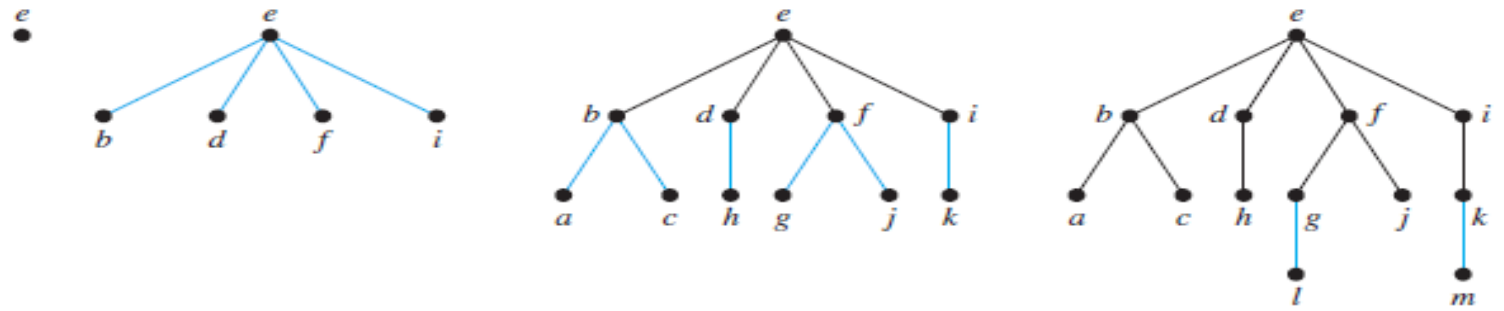


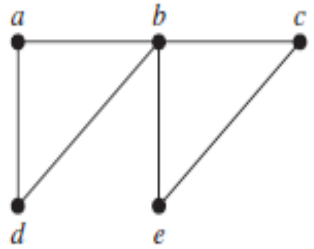
FIGURE 4 **Breadth-First Search of G.**

- We choose the vertex e to be the root.
- Then we add edges incident with all vertices adjacent to e, so edges from e to $b, d, f,$ and i are added. These four vertices are at level 1 in the tree.
- Next, add the edges from these vertices at level 1 to adjacent vertices not already in the tree.
- Hence, the edges from b to a and c are added, are edges from d to h, from f to j and g, and from i to k. The new vertices $a, c, h, j, g,$ and k are at level 2.
- Next, add edges from these vertices to adjacent vertices not already in the graph. This adds edges from g to l and from k to m.

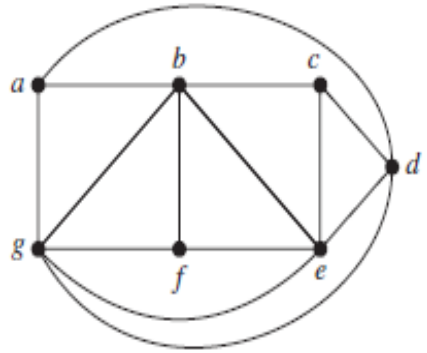
Exercises

In Exercises 2–6 find a spanning tree for the graph shown by removing edges in simple circuits.

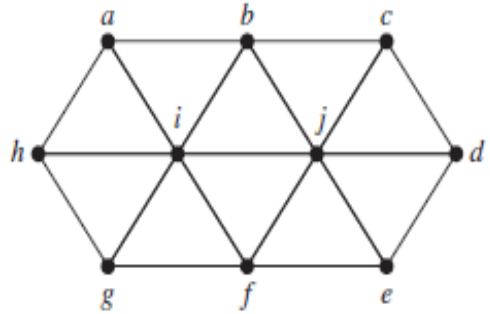
2.



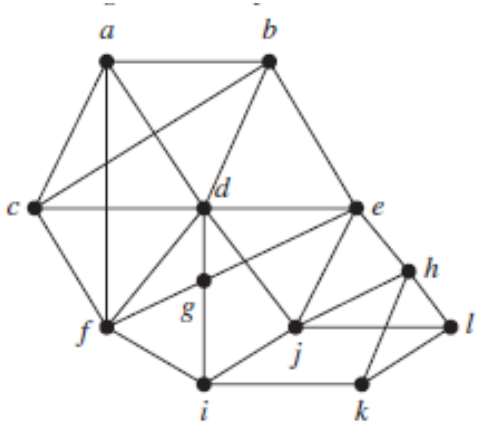
3.



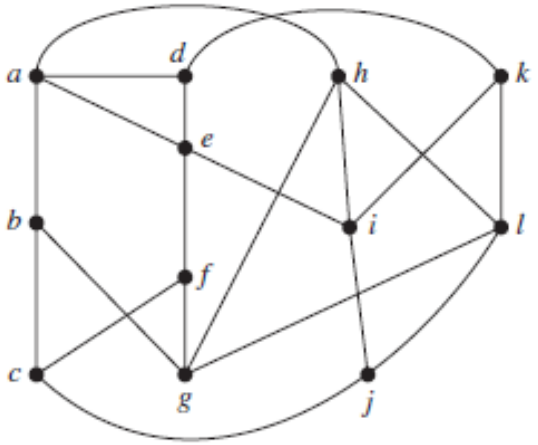
4.



5.



6.



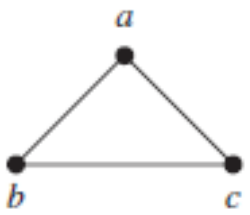
Exercises

7. Find a spanning tree for each of these graphs.

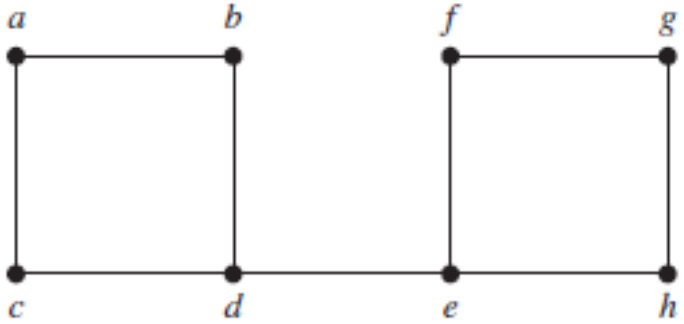
a) K_5 **b)** $K_{4,4}$ **c)** $K_{1,6}$
d) Q_3 **e)** C_5 **f)** W_5

In Exercises 8–10 draw all the spanning trees of the given simple graphs.

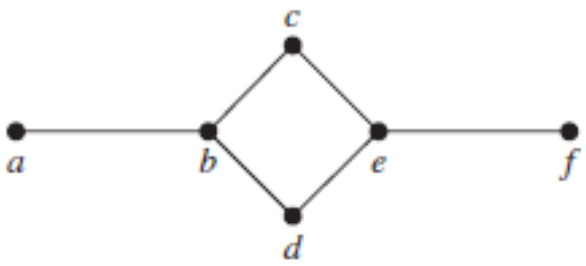
8.



9.



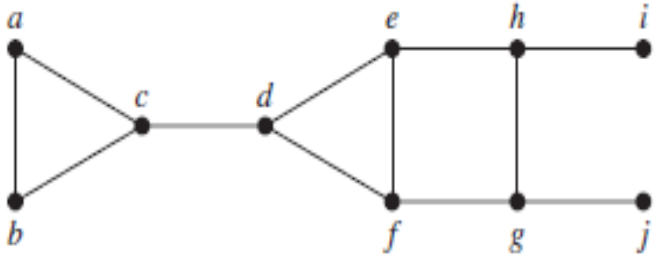
10.



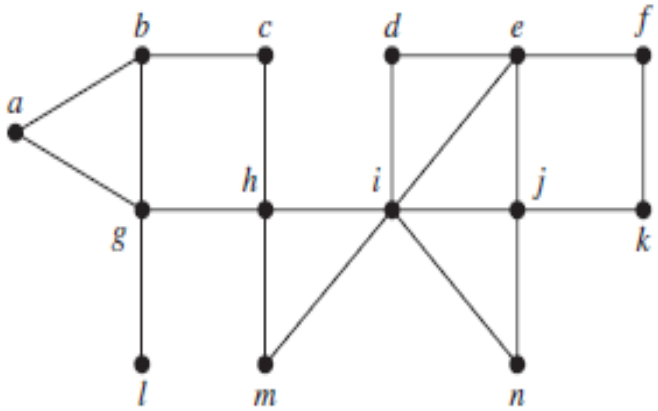
Exercises

In Exercises 13–15 use depth-first search to produce a spanning tree for the given simple graph. Choose a as the root of this spanning tree and assume that the vertices are ordered alphabetically.

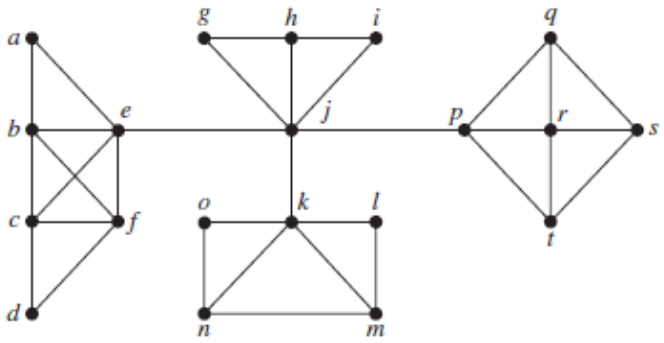
13.



14.



15.



Exercises

16. Use breadth-first search to produce a spanning tree for each of the simple graphs in Exercises 13–15. Choose a as the root of each spanning tree.

17. Use depth-first search to find a spanning tree of each of these graphs.
    a) W_6 (see Example 7 of Section 10.2), starting at the vertex of degree 6
    b) K_5
    c) $K_{3,4}$, starting at a vertex of degree 3
    d) Q_3

18. Use breadth-first search to find a spanning tree of each of the graphs in Exercise 17.