11 Trees

11.4 Spanning Tree

1. A spanning tree of a graph G is a subgraph of G that is a tree containing every vertex of G (a spanning tree is connected).

2. A graph is connected $\iff$ it has a spanning tree

3. Identifying spanning trees (note that spanning trees are not unique, unless the graph is a tree itself):

   - Edges Removal by removing edges that form a cycle/simple circuit
   - Depth-first search (or backtracking as the algorithm returns to vertices previously visited to add paths):
   - Breadth-first search

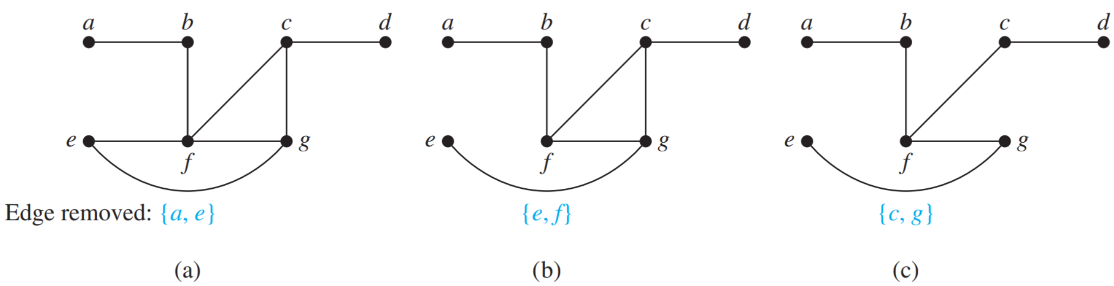


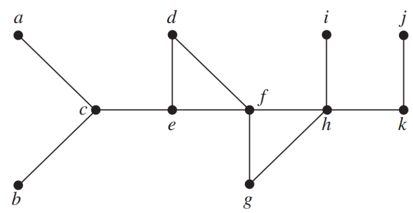
Figure 1: Edge removal example to identify a spanning tree

ALGORITHM 1 Depth-First Search.

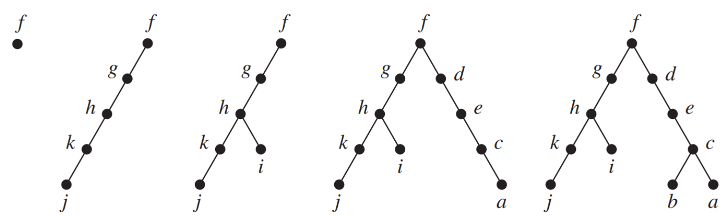
procedure *DFS*(G: connected graph with vertices $v_1, v_2, \ldots, v_n$)
T := tree consisting only of the vertex v_1
visit(v_1)

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)

[Example Graph]

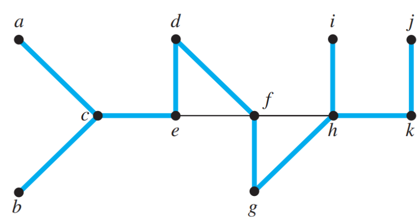


[Algorithm Steps]



(a) (b) (c) (d) (e)

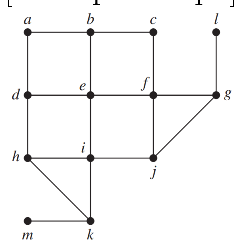
[Output Spanning Tree]



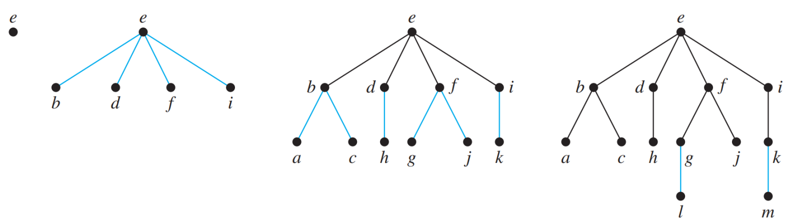
ALGORITHM 2 Breadth-First Search.

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

[Example Graph]



[Algorithm Steps]



[Output Spanning Tree]