11.4 Spanning Trees

Spanning Tree

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.

Depth-First Search

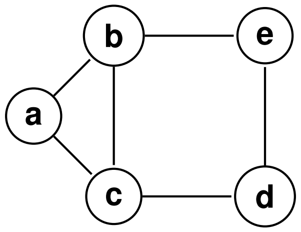
A spanning tree can be built by doing a *depth-first search* of the graph.

- Start with arbitrarily chosen vertex of the graph as the root.
- Form a path from the root by successively adding vertices and edges where
  - Each new edge is incident with the last vertex in the path.
  - Vertices are not already in the path.
  - Continue adding vertices and edges to this path as long as possible,
- If all vertices are included in the path, then “Done”.
- Otherwise, move back to the next to last vertex in the path and, if possible, form a new path starting at this vertex passing through vertices that were not already visited. If this cannot be done, move back another vertex and try again.
- Repeat this procedure until no more edges can be added.

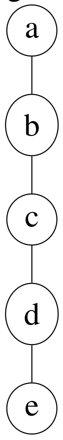
```
Algorithm DFS($G$ : connected graph with vertices $v_1, v_2, \ldots, v_n$)
1: $T$ = tree consisting only of the vertex $v_1$
2: visit($v_1$)
```

```
Procedure visit($v$ : vertex of $G$)
1: for each vertex $w$ adjacent to $v$ and not yet in $T$ do
2:     add vertex $w$ and edge $\{v, w\}$ to $T$
3:     visit($w$)
4: end for
```

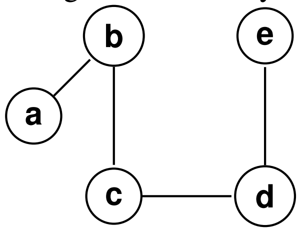
A connected graph



DFS tree starting from a as the root



Spanning tree created by DFS



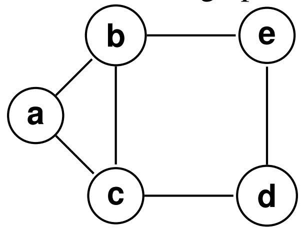
Breadth-First Search

- Start with arbitrarily chosen vertex of the graph as the root.
- Add all edges incident to the vertex along with the other vertex connected to each edge
  - The new vertices added become the vertices at level 1 in the spanning tree
  - Arbitrarily order the new vertices
- For each vertex at level 1, visited in order, add each edge incident to this vertex and the other vertex connected to the edge to the tree as long as it does not produce a simple circuit
  - Arbitrarily order the children of each vertex at level 1
  - The new vertices added become the new vertices at level 2 in the spanning tree
- Repeat the procedure until all vertices in the graph have been added

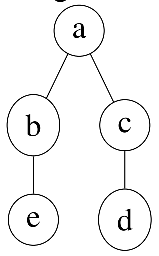
Algorithm 2 *BFS*(G : connected graph with vertices $v_1, v_2, \ldots, v_n$)

```
1:  $T$ = tree consisting only of vertex $v_1$
2:  $L$ = empty list
3:  put $v_1$ in the list $L$ of unprocessed vertices
4:  while $L$ is not empty do
5:      remove the first vertex, $v$, from $L$
6:      for each neighbor $w$ of $v$ do
7:          if $w$ is not in $L$ and not in $T$ then
8:              add $w$ to the end of the list $L$
9:              add $w$ and edge $\{v, w\}$ to $T$
10:         end if
11:     end for
12: end while
```

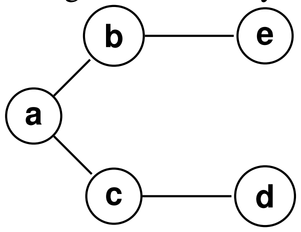
A connected graph



BFS tree starting from a as the root

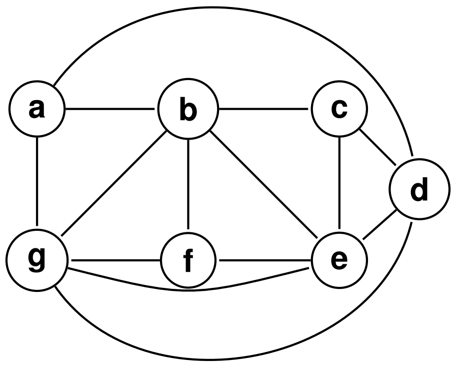


Spanning tree created by BFS

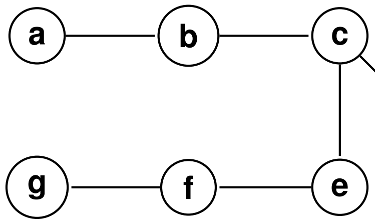


11.4 pg. 795 # 3

Find a spanning tree for the graph shown by removing edges in simple circuits.

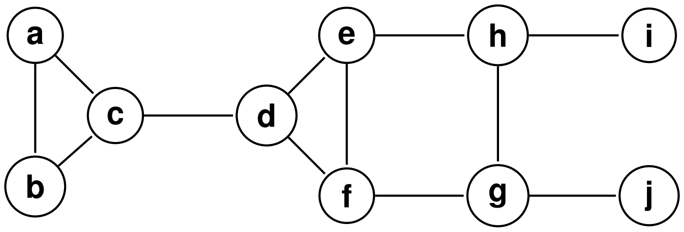


We will remove an edge one at time to remove simple circuits. We remove $\{a, d\}$, $\{a, g\}$, $\{b, e\}$, $\{b, f\}$, $\{b, g\}$, $\{d, e\}$, $\{d, g\}$, and $\{e, g\}$

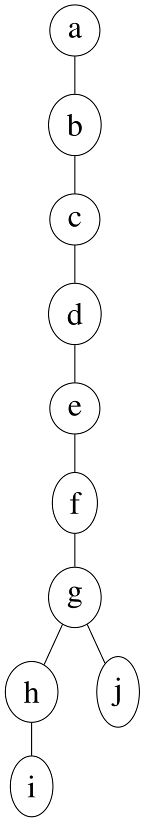


11.4 pg. 795 # 13

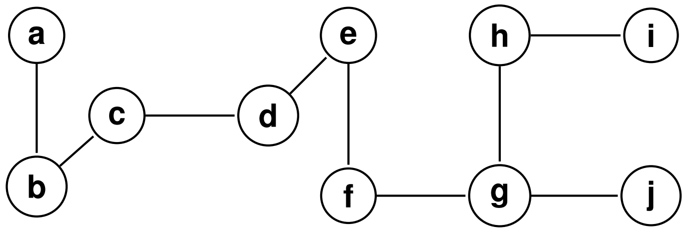
Use depth-first search and breadth-first search to produce a spanning tree for the given simple graph. Choose a as the root of this spanning tree and assume that the vertices are ordered alphabetically.



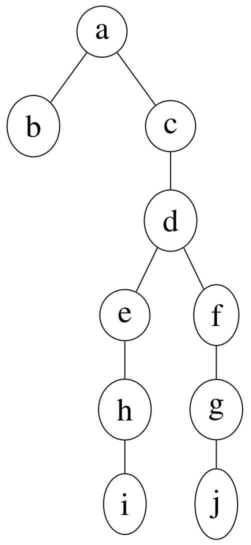
The DFS tree is as follows:



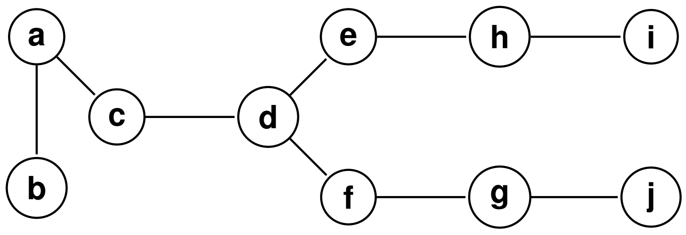
The spanning tree by DFS:



The BFS tree is as follows:



The spanning tree by BFS:


a
e
h
i
c
d
b
f
g
j