Graph Theory and Its Applications

Dr. G.H.J. Lanel

Lecture 6

Outline

Outline

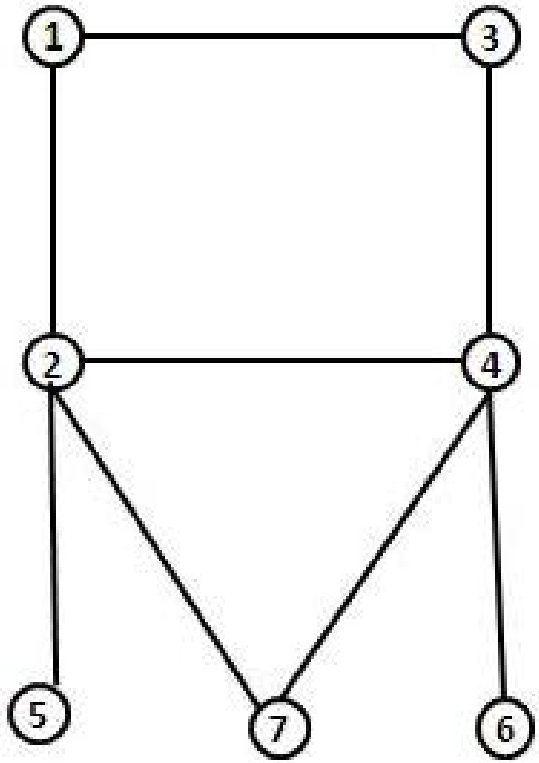
There are two different efficient algorithms for finding spanning trees.

1. Depth first search Algorithm (DFS)
2. Breadth first search Algorithm (BFS)

Depth first search Algorithm (DFS)

Depth first search is a recursive algorithm for visiting all the vertices of a connected graph G.

Example:



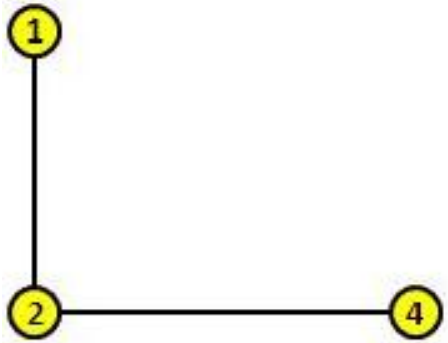
Step 1:

Start at vertex 1.

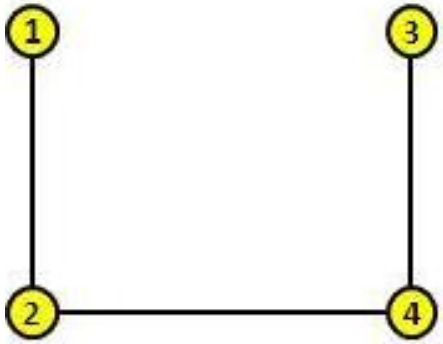
Step 2:

Look at 1's first neighbor nearly vertex 2 has not get been visited we visit it.

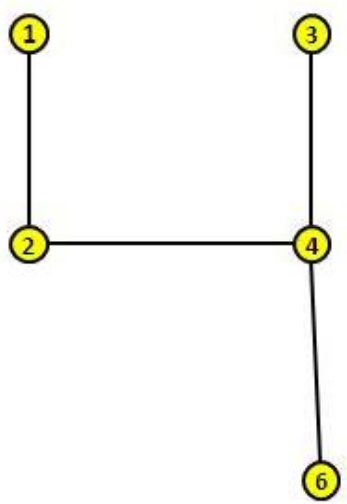
Step 3:



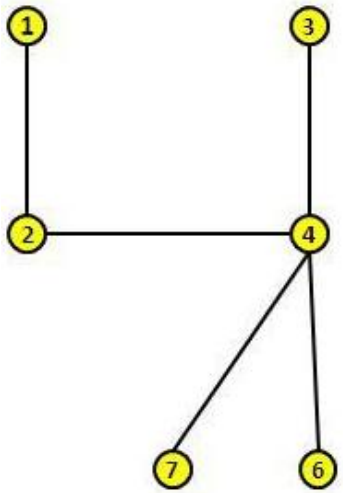
Step 4:



Step 5:

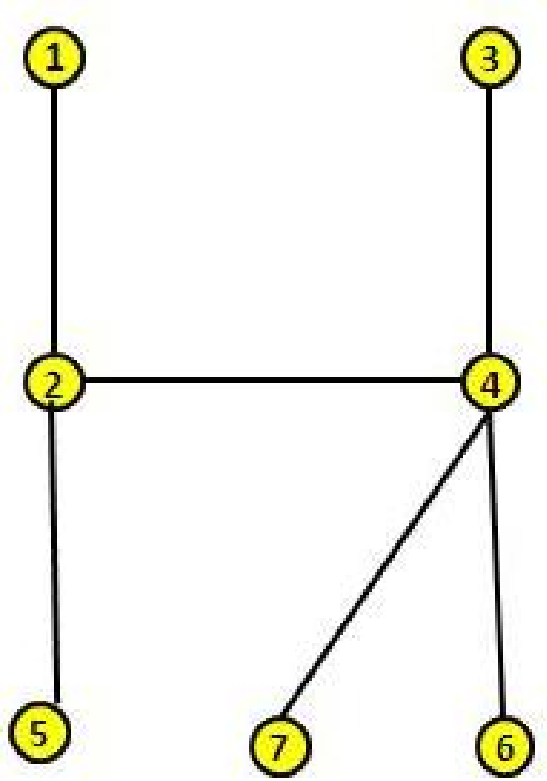


Step 6:



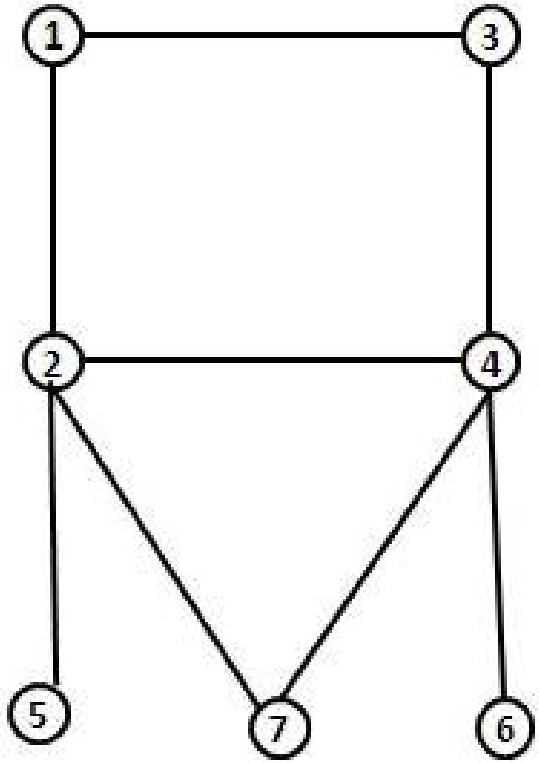
Step 7:

Spanning tree of G.



Breadth first search Algorithm (BFS)

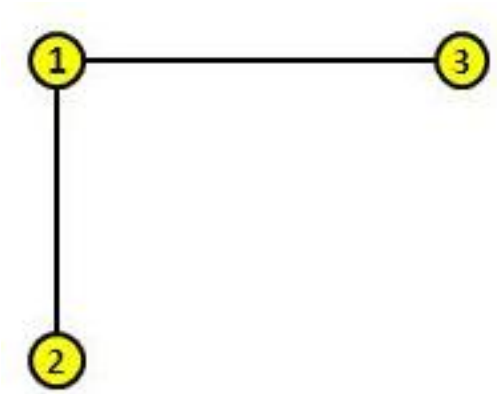
In breadth first search, when are first encounter a vertex, we do not proceed to search further from that vertex immediately. Example:



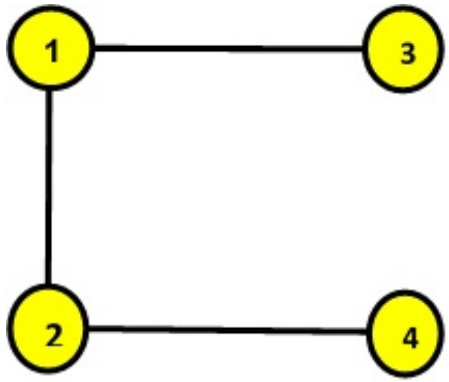
Step 1:

Step 2:

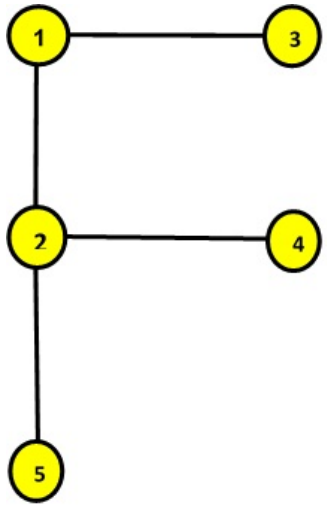
Step 3:



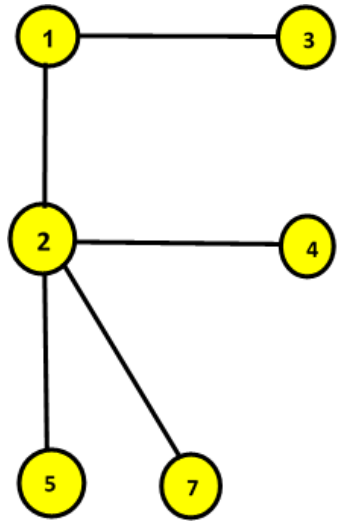
Step 4:



Step 5:

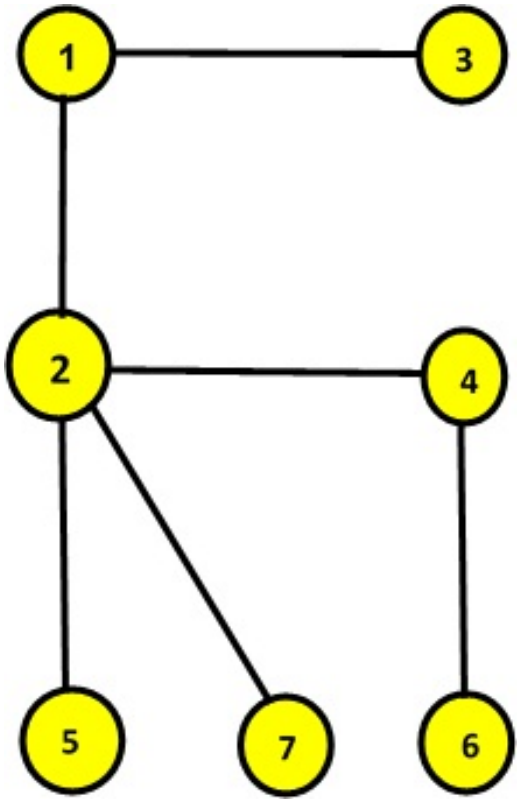


Step 6:



Step 7:

Spanning tree of G.



Outline

1. DFS and BFS are used to find spanning trees of a connected graphs.
2. Usually connected graph with n vertices has $n - 1$ edges.
3. If we consider connected weighted graph, we can look at the weight of spanning tree, that is sum of the weights of its edges.
4. The weight of different spanning trees may be quite different.
5. Some time it is very useful to find spanning trees with minimum weight such a tree is called minimum spanning tree.

Kruskal Algorithm

Remove from a connected graph as many edges as possible while remaining connected; this should yield a tree with $n-1$ edges. This is the minimal spanning tree found by the following algorithm.

Algorithm KruskalMST(G)

$E_{rest} := sort(E); E' := \phi;$

while $\mid E' \mid < n-1$ **do**

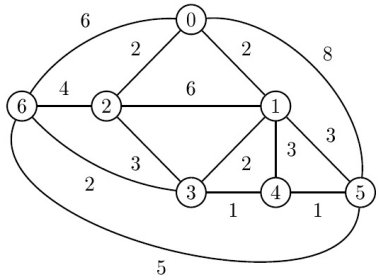
$\alpha := first(E_{rest}); E_{rest} := E_{rest} \setminus \{\alpha\};$

if $(V, E' \cup \{\alpha\})$ is acyclic **then**

$E' := E' \cup \{\alpha\};$

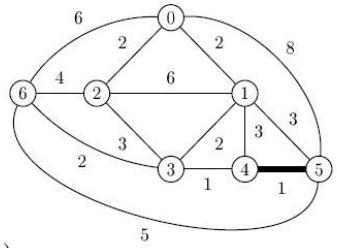
end if
end while

Let us look at an example

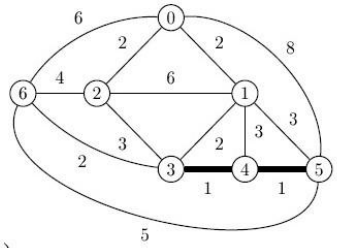


The different steps of the algorithm are

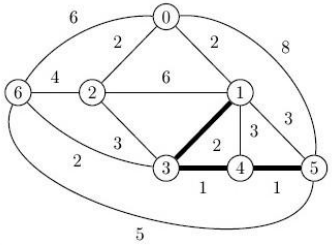
(1)



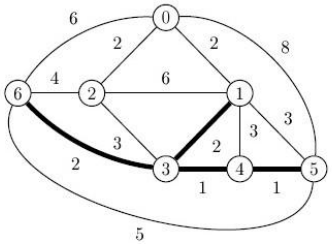
(2)



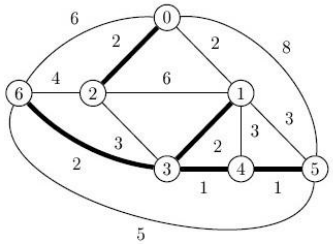
(3)



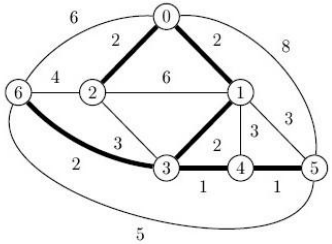
(4)



(5)



(6)



This constructs a tree which is a subgraph with $n - 1$ edges.

Prim's Algorithm

Now we look at an alternative algorithm with different time complexity.

The idea is to pick a random node and then grow a minimal tree from there,

Algorithm PrimMST(G)

Choose $u \in V$; $V' := \{u\}$; $E' := \phi$

for $i = 1 : n - 1$ **do**

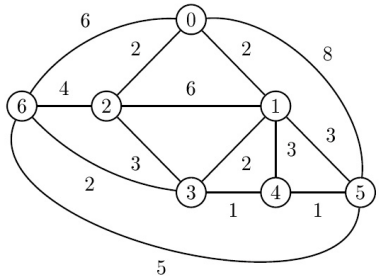
$E'' :=$ edges linking V to V'

choose $e = (u, v) \in E''$ of minimal weight and such that

$(V' \cup \{v\}), E' \cup \{e\})$ is acyclic

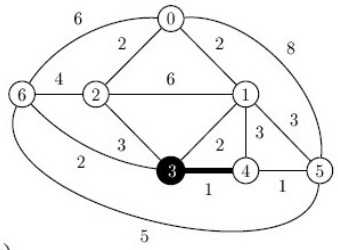
end for

Let us look at an example

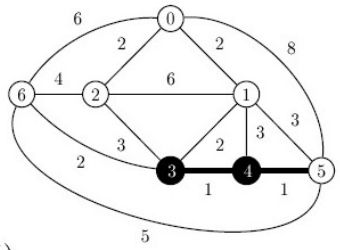


The different steps of the algorithm are

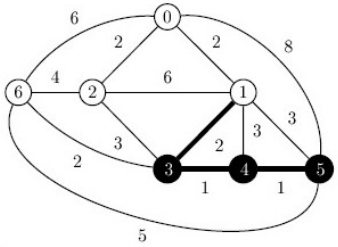
(1)



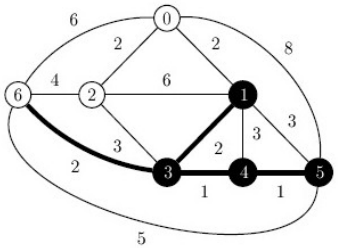
(2)



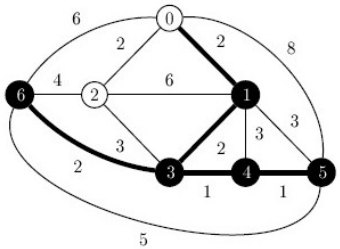
(3)



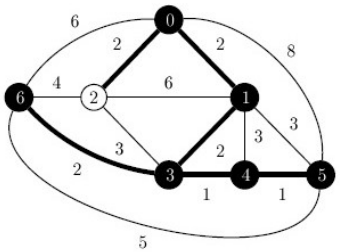
(4)



(5)



(6)



The graph (V, E') is a minimal spanning tree with $n - 1$ edges.