Chapter 5: Trees and Spanning Trees

Discrete Mathematics 2

Lecturer: Nguyen Kieu Linh
Posts and Telecommunications Institute of Technology

Hanoi, 2024
http://www.ptit.edu.vn

Contents

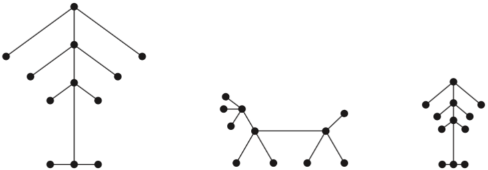
Trees

Definition

- A *tree* is a connected undirected graph with no simple circuits.
- A *forest* is an undirected graph with no simple circuits. A forest has the property that each of its connected components is a tree.

Example

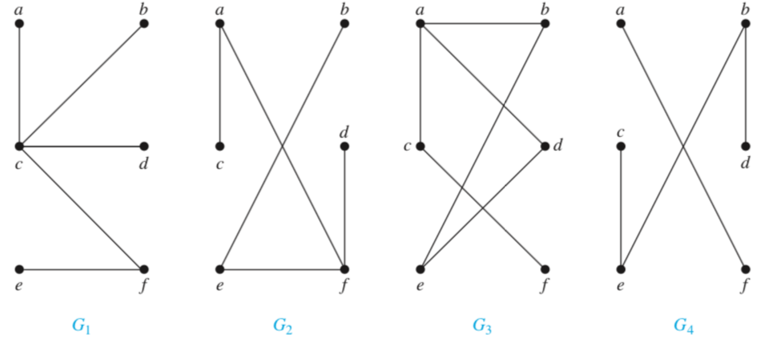
Example of a Forest.



Trees

Example

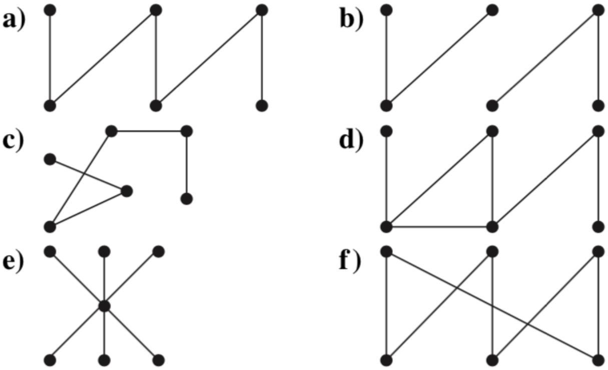
Which of the following graphs are trees?



Trees

Example

Which of the following graphs are trees?



Trees

Theorem

Suppose that $T = \langle V, E\rangle$ is an undirected graph with n vertices, the following statements are equivalent:
1) T is a tree
2) T has no simple circuits and has $n-1$ edges
3) T is connected and has $n-1$ edges
4) T is connected and each its edge is a cut edge
5) There is exactly one simple path connecting between two every vertices of T
6) T has no simple circuits but if we add a new edge we will have exactly one circuit

Proof: Strategy

$$(1) \Rightarrow (2) \Rightarrow (3) \Rightarrow (4) \Rightarrow (5) \Rightarrow (6) \Rightarrow (1)$$

Contents

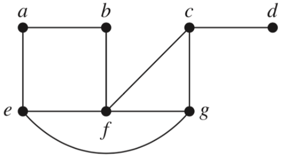
Spanning trees

Definition

* Suppose that G is a connected undirected graph. A subgraph T of G is called a spanning tree of G if T satisfies two following conditions:
  - T is a tree
  - The set of vertices of T equals to the set of vertices of G

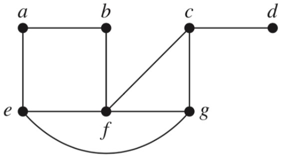
Example

Find a spanning tree of the following?

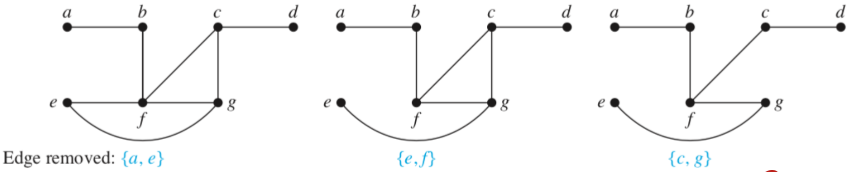


Example

Find a spanning tree of the following?

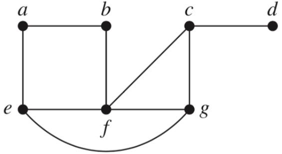


Solution.

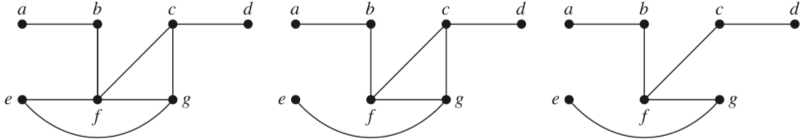


Example

Find a spanning tree of the following?

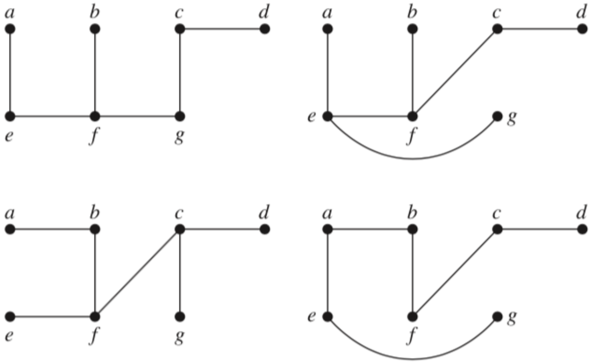


Solution.



Edge removed: $\{a, e\}$ $\{e, f\}$ $\{c, g\}$

Solution.



Producing a spanning tree from a simple graph

Problem: Given an undirected graph $G = \langle V, E \rangle$, produce a spanning tree of G starting from a vertex $v \in V$.

Method

- Using DFS or BFS
- When we reach vertex v from vertex u, edge (u, v) is added to the spanning tree.

Producing a spanning tree from a simple graph by DFS

Recursive algorithm starting from u

```
Tree-DFS(u){
        unChecked[u] = false;  //u has been visited
        for(v ∈ Adj(u)){
                if( unChecked[v]){ //v has not been visited
                        T = T ∪ {(u,v)}; //add (u, v) to spanning tree
                        Tree-DFS(v);      //recursive from v
                }
        }
}
```

Producing a spanning tree from a graph by DFS

```
Tree-Graph-DFS( ){
        //All vertices have not been visited
        for(u ∈ V)
                unChecked[u] = true;

        root = < a vertex of the graph >; //starting from any vertex
        T = ∅; //at the beginning the spanning tree is empty
        Tree-DFS(root); //call the recursive algorithm from root

        if(|T| < n − 1)
                <the graph is not connected>;
        else
                <we have the set of edges of spanning tree T>;
}
```

Producing a spanning tree from a graph by BFS

```
Tree-BFS(u){
        Step 1: Initialize
        T = ∅; queue = ∅; push(queue, u); chuaxet[u] = false;
        Step 2: Loop
        while(queue ≠ ∅){
                s = pop(queue);
                for(t ∈ Adj(s)){
                        if( unChecked[t]){
                                push(queue, t);
                                T = T ∪ {(s, t)};
                                unChecked[t] = false;
                        }
                }
        }
        Step 3: Return results
        if(|T| < n − 1) <graph is not connected>;
        else <we have the set of edges of spanning tree T>;
}
```

Example

Given an undirected graph represented as the adjacency matrix as below. Building a spanning tree of the graph using DFS starting from vertex $u = 1$.

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Verification

#	Vertices as the order of calling Tree-DFS(u)	T
0	1	$T = \emptyset$
1	1, 2	$T = T \cup \{(1,2)\}$
2	1, 2, 3	$T = T \cup \{(2,3)\}$
3	1, 2, 3, 4	$T = T \cup \{(3,4)\}$
4	1, 2, 3	
5	1, 2, 3, 5	$T = T \cup \{(3,5)\}$
6	1, 2, 3, 5, 6	$T = T \cup \{(5,6)\}$
7	1, 2, 3, 5, 6, 7	$T = T \cup \{(6,7)\}$
8	1, 2, 3, 5, 6, 7, 8	$T = T \cup \{(7,8)\}$
9	1, 2, 3, 5, 6, 7, 8, 9	$T = T \cup \{(8,9)\}$
10	1, 2, 3, 5, 6, 7, 8, 9, 10	$T = T \cup \{(9,10)\}$
11	1, 2, 3, 5, 6, 7, 8, 9, 10, 11	$T = T \cup \{(10,11)\}$
12	1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12	$T = T \cup \{(11,12)\}$
13	1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13	$T = T \cup \{(12,13)\}$
Cannot add edges to T		
$T = \{(1,2), (2,3), (3,4), (3,5), (5,6), (6,7), (7,8), (8,9), (9,10), (10,11), (11,12, (12,13))\}$		

Example

Given an undirected graph represented as the adjacency matrix as below. Building a spanning tree of the graph using BFS starting from vertex $u = 1$.

$$\begin{bmatrix}
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0
\end{bmatrix}$$

Verification

#	Queue	T
0	1	$T = \emptyset$
1	2, 3, 4	$T = T \cup \{(1,2), (1,3), (1,4)\}$
2	3, 4	
3	4, 5	$T = T \cup \{(3,5)\}$
4	5	
5	6, 7, 8, 9	$T = T \cup \{(5,6), (5,7), (5,8), (5,9)\}$
6	7, 8, 9	
7	8, 9	
8	9	
9	10	$T = T \cup \{(9,10)\}$
10	11, 12, 13	$T = T \cup \{(10,11), (10,12), (10,13)\}$
11	12, 13	
12	13	
13	$\emptyset$	
$T = \{(1,2), (1,3), (1,4), (3,5), (5,6), (5,7), (5,8), (5,9),$ $(9,10), (10,11), (10,12, (10,13))\}$		

Contents

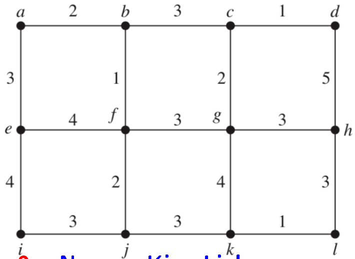
Minimum spanning tree problem

Definition

A *minimum spanning tree* in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.

Example

A Weighted Graph.



Minimum spanning tree problem

Problem Statement

- Given $G = [V, E\rangle$ is a connected, undirected graph with the set of vertices V and the set of edges E. Each edge e is assigned to a non-negative real number $c(e)$ called the length of the edge.
- Suppose that $H = \langle V, T >$ is a spanning tree of G. The length of the spanning tree H, denoted by $c(H)$, is the sum of the lengths of edges:

$$c(H) = \sum_{e \in T} c(e)$$

- Among spanning trees of the graph, find the minimum (length) spanning tree.

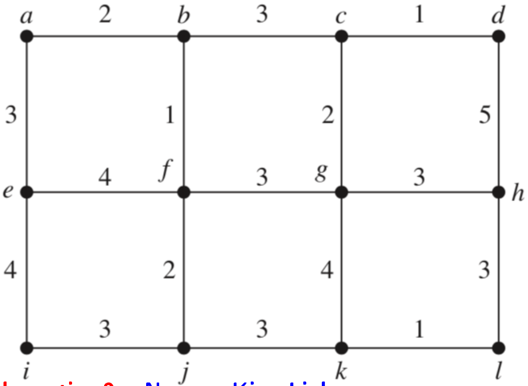
Prim's Algorithm

```
Prim( s){
    Step 1 (Initialize):
    V_H = {s};          //At the beginning V_H contains only s
    V = V\{s};          //Remove s from V
    T = ∅;              //Spanning tree is empty
    d(H) = 0;           //Length is 0
    Step 2 (Loop):
    while(V ≠ ∅ ){
        e = (u, v);         //Minimum length edge with u ∈ V, v ∈ V_H
        if(e does not exist)
            return <Not connected>;
        T = T ∪ {e};        //Add e to the spanning tree
        d(H) = d(H) + d(e); //Update length
        V_H = V_H ∪ {u};    //Add u to V_H
        V = V\{u};          //Remove u from V
    }
    Step 3 (Return results):
    return (T, d(H));
}
```

Prim's Algorithm

Example

Use Prim's algorithm to find a minimum spanning tree in the graph from the vertex *a*.



Example

Using Prim's algorithm to find the minimum spanning tree of the graph represented as the weighted matrix below starting from the vertex 1?

$$
\begin{bmatrix}
\infty & 2 & 1 & 3 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
2 & \infty & 2 & \infty & \infty & 5 & 5 & \infty & \infty & \infty & \infty & \infty & \infty \\
1 & 2 & \infty & 4 & \infty & 5 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
3 & \infty & 4 & \infty & 5 & 5 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & 5 & \infty & 6 & \infty & \infty & \infty & 6 & \infty & \infty & \infty \\
\infty & 5 & 5 & 5 & 6 & \infty & 6 & 6 & 6 & 6 & \infty & \infty & \infty \\
\infty & 5 & \infty & \infty & \infty & 6 & \infty & 6 & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & 6 & 6 & \infty & 7 & \infty & \infty & 7 & 7 \\
\infty & \infty & \infty & \infty & \infty & 6 & \infty & 7 & \infty & 7 & 7 & \infty & \infty \\
\infty & \infty & \infty & \infty & 6 & 6 & \infty & \infty & 7 & \infty & 7 & 7 & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 7 & 7 & \infty & 8 & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & 7 & \infty & 7 & 8 & \infty & 8 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & 7 & \infty & \infty & \infty & 8 & \infty
\end{bmatrix}
$$

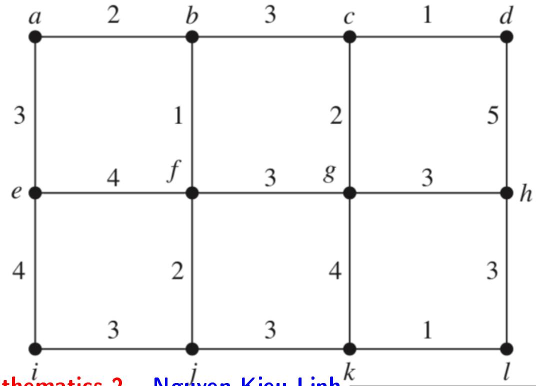
Kruskal's Algorithm

```
Kruskal( ){
    Step 1 (Initialize):
    T = ∅; //At the beginning the set of edges is empty
    d(H) = 0; //Length equals to 0
    Step 2 (Sort):
    <Sort edges of the graph in the ascending order of length>;
    Step 3 (Loop):
    while(|T| < n − 1 && E ≠ ∅ ){
        e = <The minimum length edge>;
        E = E\{e}; //Remove e
        if (T ∪ {e} dose not produce a circuit ){
            T = T ∪ {e}; //Adds e to the spanning tree
            d(H) = d(H) + d(e); //Update the length
        }
    }
    Step 4 (Return results):
    if(|T| < n − 1) <Not connected>;
    else return (T, d(H));
}
```

Kruskal's Algorithm

Example

Use Kruskal's algorithm to find a minimum spanning tree in the graph



Example

Using Kruskal's Algorithm to find the minimum spanning tree of the graph represented as the weighted matrix below?

$$\begin{bmatrix}
\infty & 2 & 1 & 3 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
2 & \infty & 2 & \infty & \infty & 5 & 5 & \infty & \infty & \infty & \infty & \infty & \infty \\
1 & 2 & \infty & 4 & \infty & 5 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
3 & \infty & 4 & \infty & 5 & 5 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & 5 & \infty & 6 & \infty & \infty & \infty & 6 & \infty & \infty & \infty \\
\infty & 5 & 5 & 5 & 6 & \infty & 6 & 6 & 6 & 6 & \infty & \infty & \infty \\
\infty & 5 & \infty & \infty & \infty & 6 & \infty & 6 & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & 6 & 6 & \infty & 7 & \infty & \infty & 7 & 7 \\
\infty & \infty & \infty & \infty & \infty & 6 & \infty & 7 & \infty & 7 & 7 & \infty & \infty \\
\infty & \infty & \infty & \infty & 6 & 6 & \infty & \infty & 7 & \infty & 7 & 7 & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 7 & 7 & \infty & 8 & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & 7 & \infty & 7 & 8 & \infty & 8 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & 7 & \infty & \infty & \infty & 8 & \infty
\end{bmatrix}$$

Summary

- Definitions and properties of trees
- Spanning tree
  - Every connected undirected graph has at least one spanning tree
  - Producing a spanning tree using BFS and DFS algorithms
- Minimum spanning tree problem
  - Prim's Algorithm
  - Kruskal's Algorithm