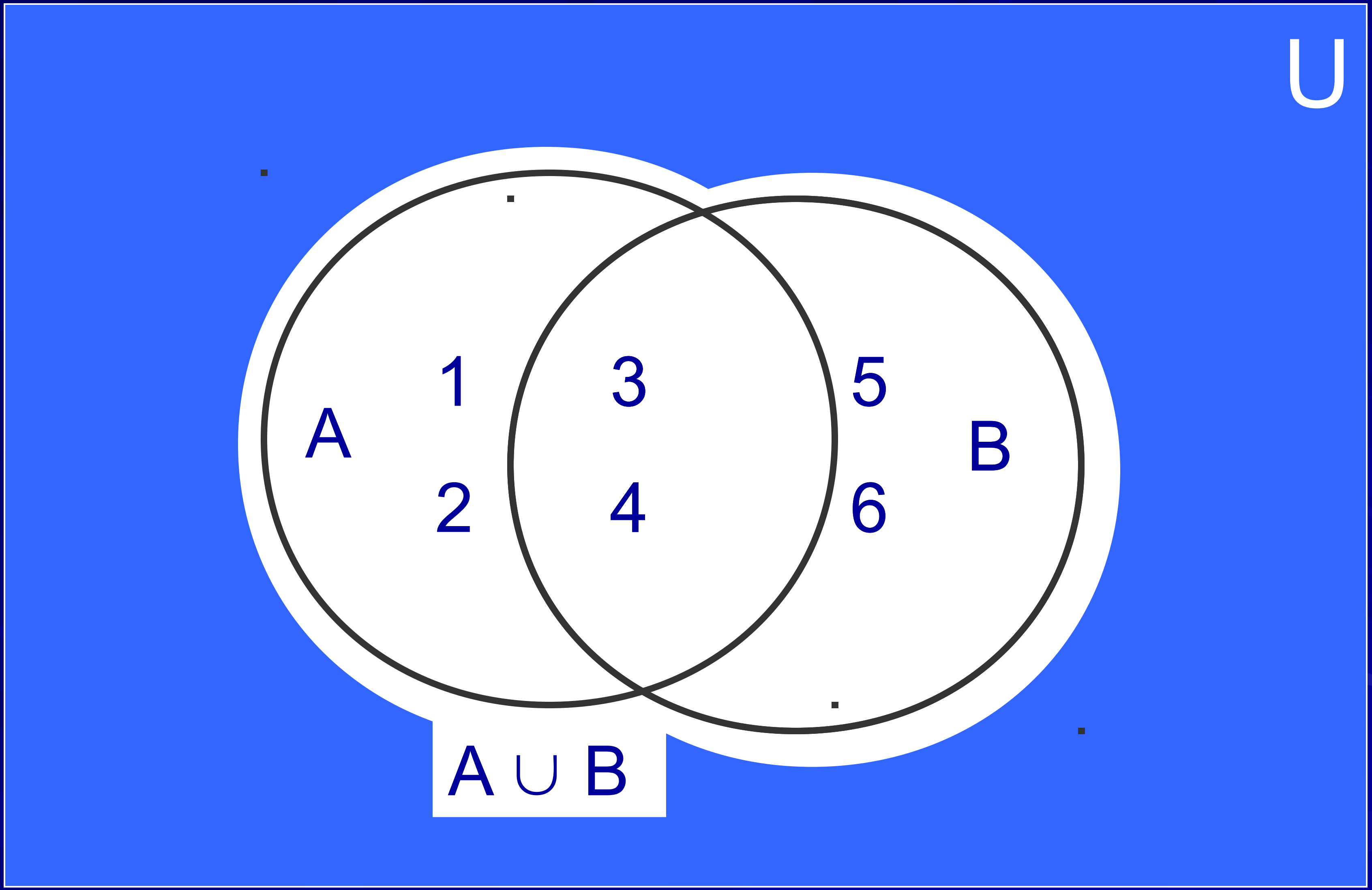
Minimal Spanning Trees

Union

- The ***union*** of A and B is the set containing all elements that belong to A or B, denote A U B.
  - A U B = { x | x Є A or x Є B}
- Show Venn Diagram with shading
- Ex: Let

  A = { 1, 2, a, f, 5} and B = { 1,a,3,f,g}
  - Then A U B = {1,2,a,f,5,3,g}

Union



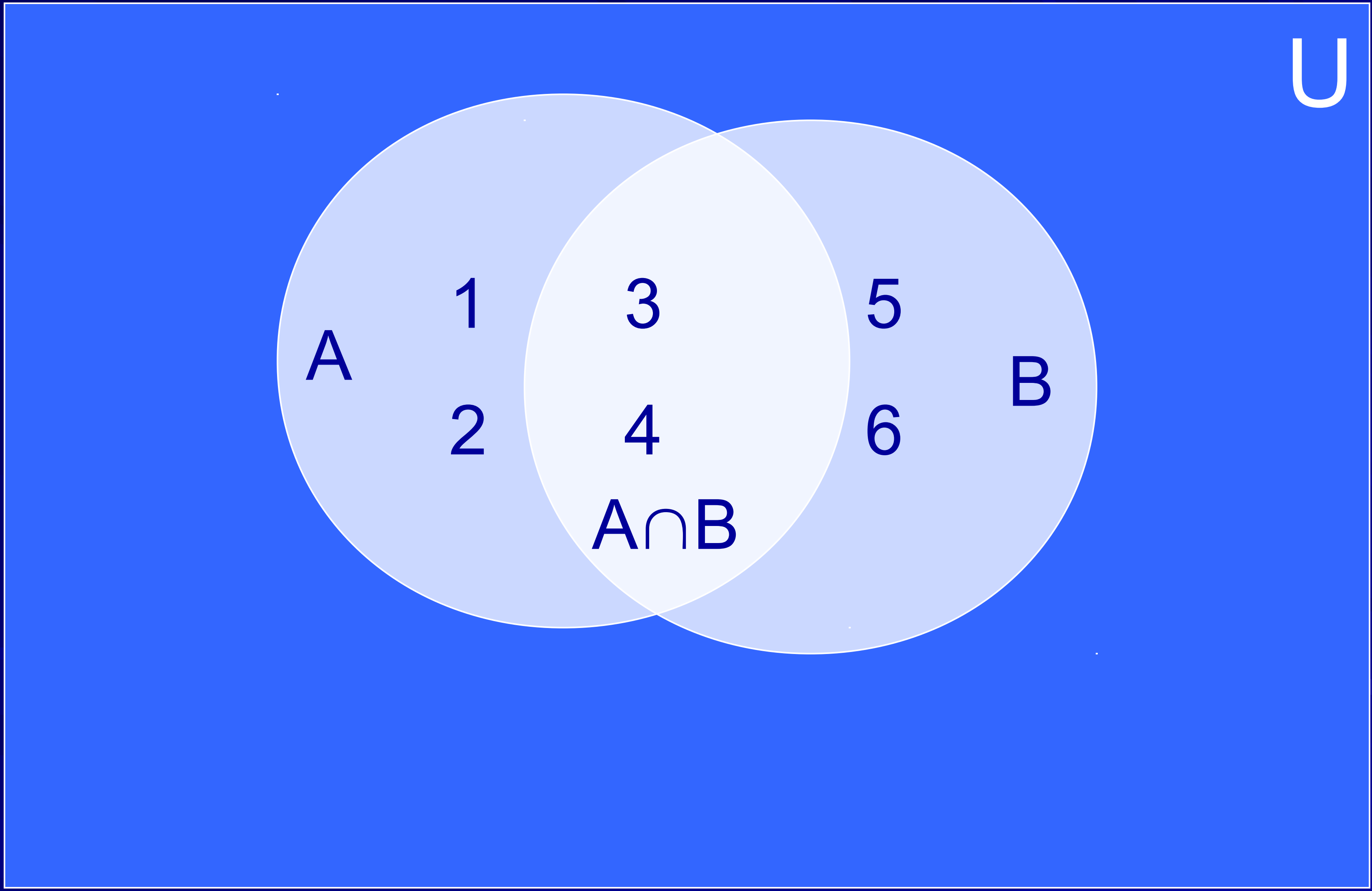
Intersection

- The ***intersection*** of A and B is the set containing all elements that belong to A and belong to B, denote $A \cap B$.
  - $A \cap B = \{ x \mid x \in A \text{ and } x \in B\}$
- Show Venn Diagram with shading
- Ex: Let

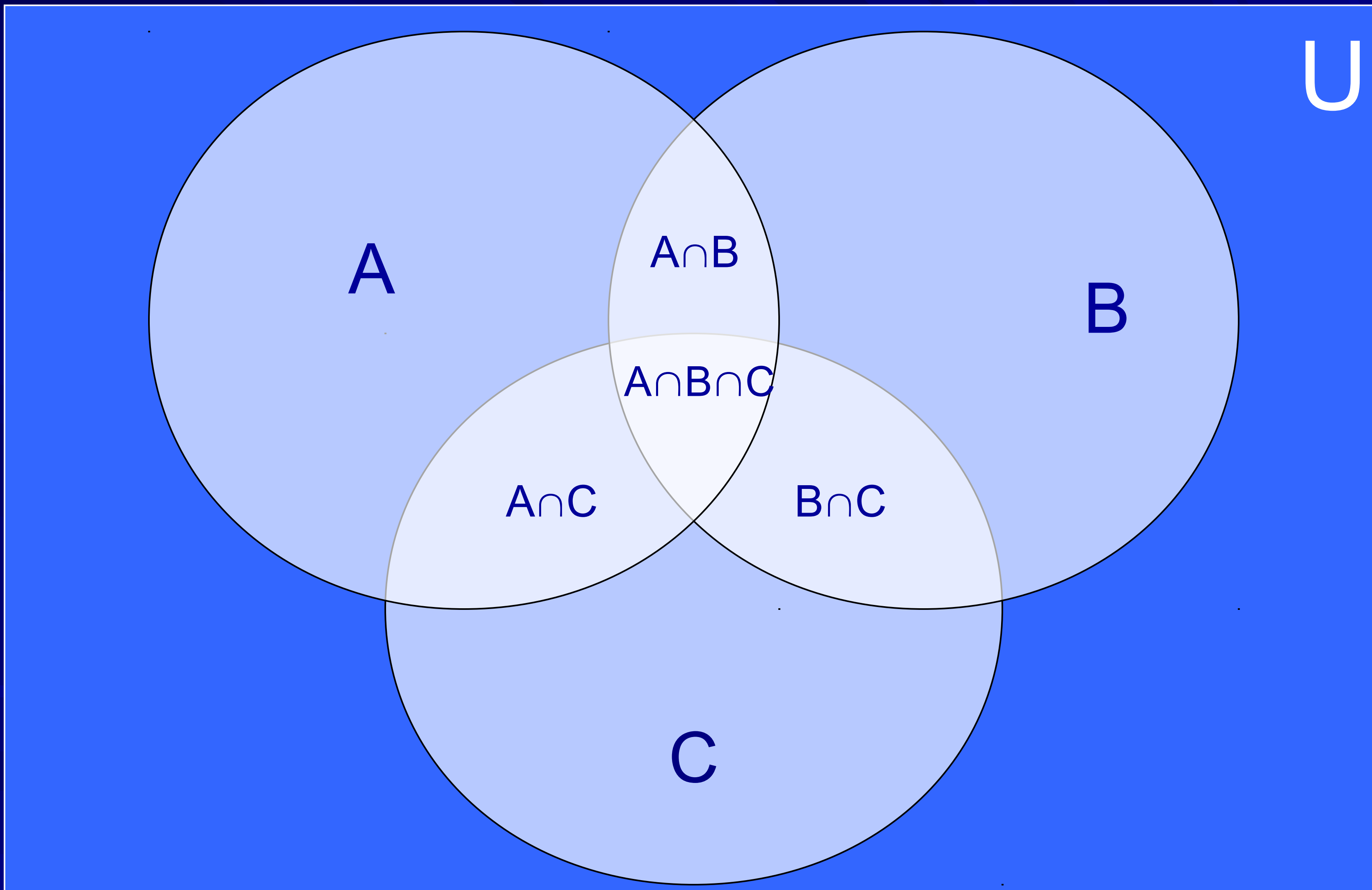
  $A = \{ 1, 2, a, f, 5\}$ and $B = \{ 1,a,3,f,g\}$

  and $C = \{2, 6\}$
  - Then $A \cap B = \{1,a,f,\}$ $A \cap C = \{2\}$ $C \cap B = \emptyset$
  - C and B are ***disjoint sets***.

Intersection 2 Sets



Intersection 3 Sets



Cartesian Product

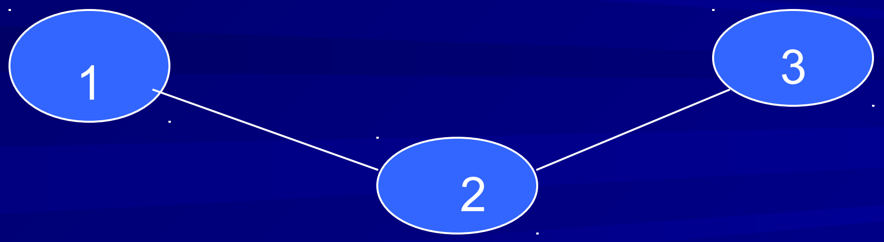
- If A and B are nonempty sets, the ***product set*** (or ***Cartesian product***), A x B is the set of all ordered pairs (a,b) with $a \in A$ and $b \in B$.
- (a,b) is a 2-tuple
- Example A={1,2,3} B={r,s}
  - AxB={(1,r), (2,r), (3,r),   (1,s), (2,s),   (3,s) }
- Example R x R, set of all points in the plane
  - $(1,2) \neq (2,1)$

Relational Database

- A={1,2,3} B={r,s}   C={t,y}
  - AxBxC={(1,r,t), (1,r,y), (1,s,t), (1,s,y),
    (2,r,t), (2,r,y), (2,s,t), (2,s,y),
    (3,r,t), (3,r,y), (3,s,t), (3,s,y)}
- Example
  - Employee ID, Last Name, Department, Years with Company
  - Database is a subset of ID x Name x Depart x YwC
- You need Union, Intersection, cross product for Database next semester

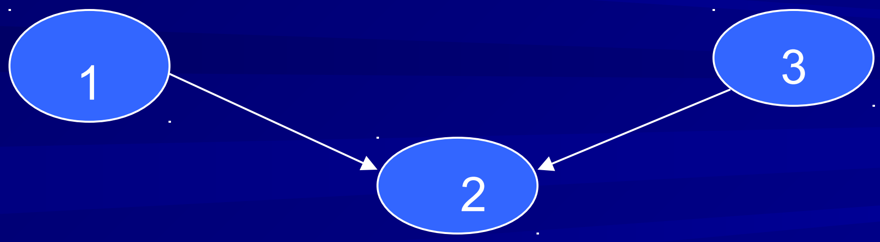
Graph

- A graph, ***G=(V,E)***, consists of a finite set V of objects called vertices, and a finite set E of objects called edged.
- Ex V={1,2,3} E= { {1,2}, {2,3} }



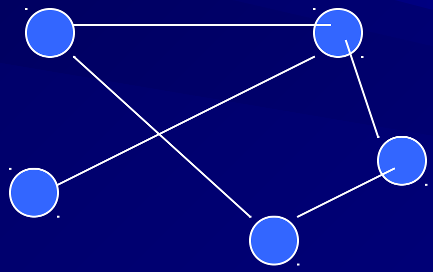
Digraph

- A Digraph, ***G=(V,A)***, consists of a finite set V of objects called vertices, and a finite set A of objects called arcs.
- Ex V={1,2,3} A= { (1,2), (3,2) }

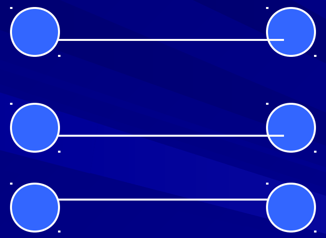


Connected Graph

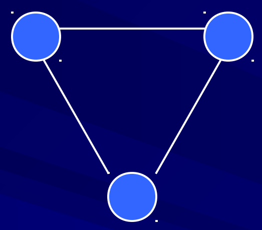
- A graph is connected if there is a path (of edges) between any two pairs of vertices.
- Connected



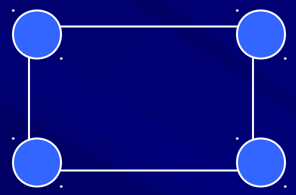
Disconnected



Cycle

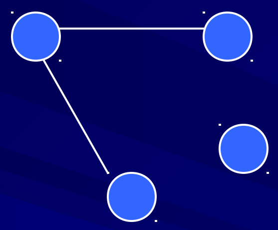


C_3

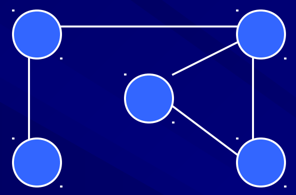


C_4

Acyclic: No Cycles

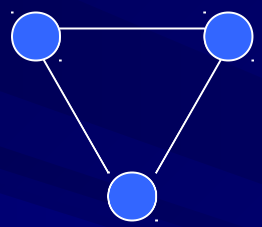


A disconnected
acyclic graph

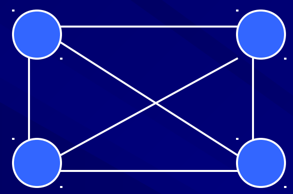


Not an acyclic graph

Complete Graph



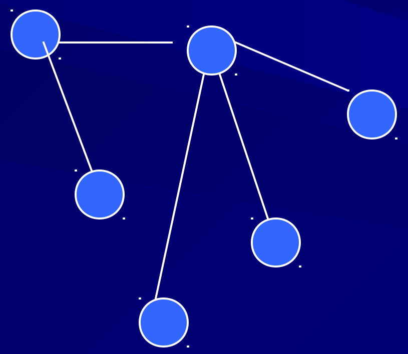
K_3



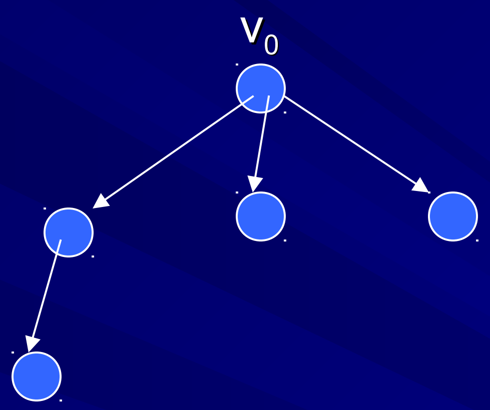
K_4

Tree

- ***Tree:*** a connected acyclic graph.

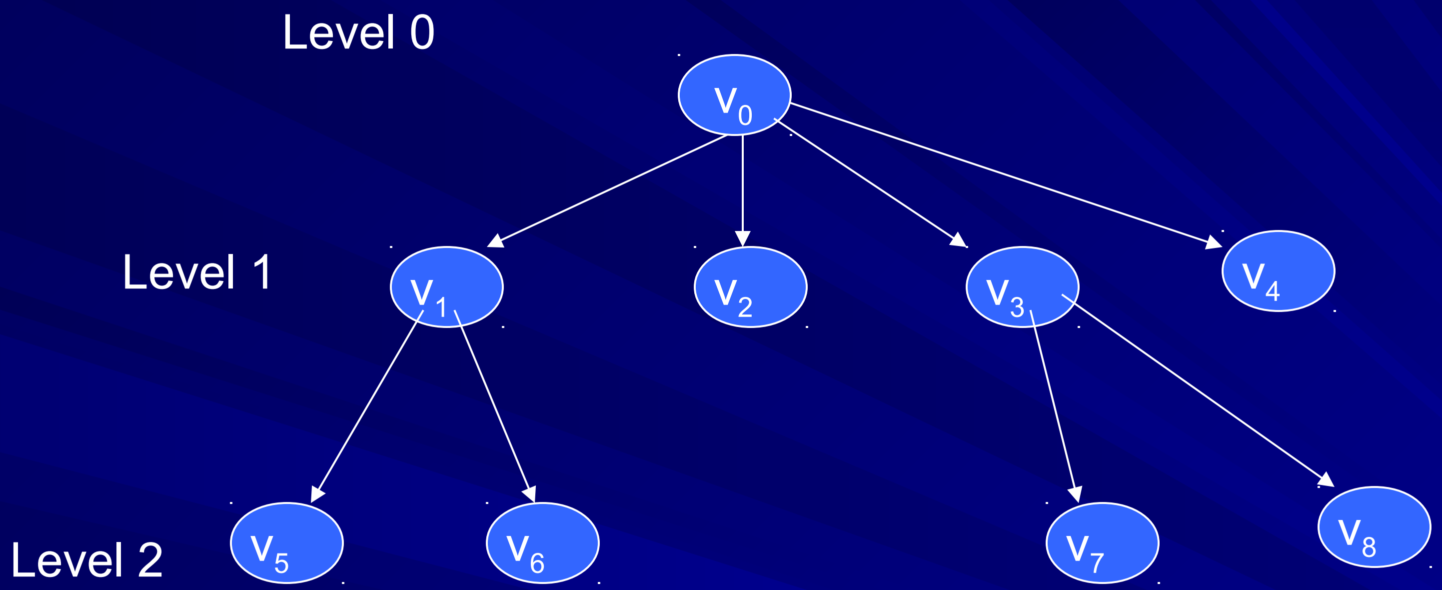


Rooted Tree



- (T,v_0) is a rooted tree
  - v_0 has in-degee 0 and all other vertices have in-degree 1

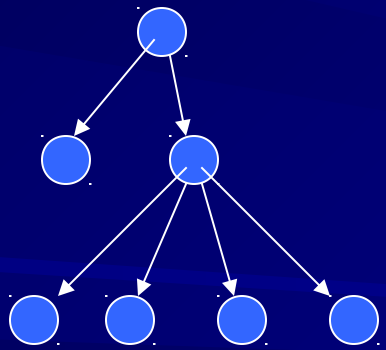
Tree Terms



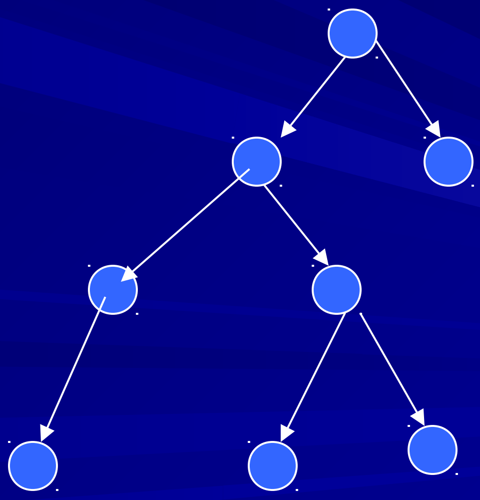
- v0 is the parent of v1 , v2 , v3 , and v4
- v1 , v2 , v3 , and v4 are children (or offspring) of v0
- v1 , v2 , v3 , and v4 are siblings
- v5 , v6 , v2 , v7 , v8 , and v4 are leaves (no offspring)
- The height of the tree is 2

n-tree

- We say T is a n-tree if every vertex has at most n offspring.
- N=2 is a 2-tree or binary tree.
- 4-tree

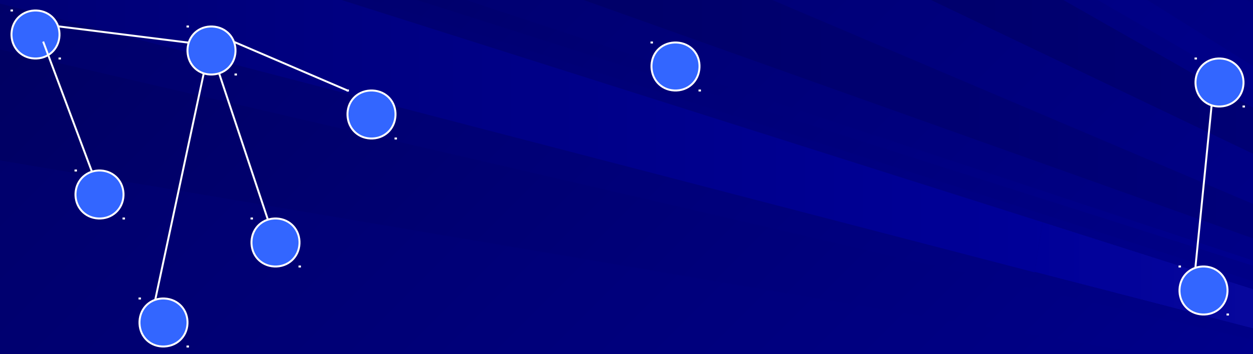


2-tree



Forest

- ***Forest:*** a collection of trees.



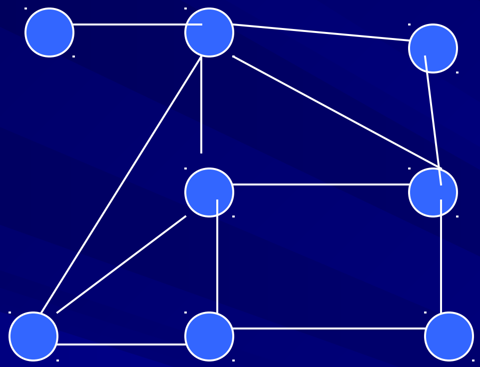
Disjoint Sets

- Example   a={1,4} b={2,3}   c={7}
- Make-Set(x)
  - Makes a new set whose member is x, and x is not in another set.
- Union(x,y)
  - Combines the sets that contain x and y
- Find-Set(x)
  - Finds the set which contains x

Disjoint Sets

- Example {1,4} {2,3} {7}
- Make-Set(x)
  - Make-Set(5)
    - {1,4} , {2,3}, {7}, {5}
- Union(x,y)
  - Union(2,7)
    - {1,4} {2,3,7} {5}

Spanning Tree

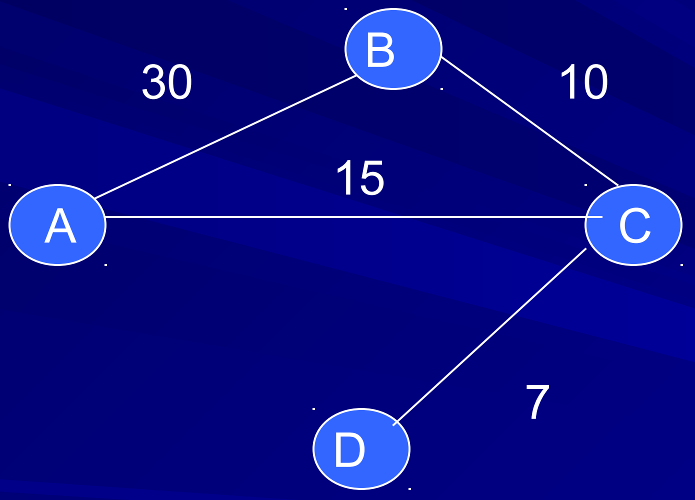


Spanning Tree

- T is a spanning tree of graph G, if T is a tree with the same vertices as G, and a subset of the edges of G.

Notation

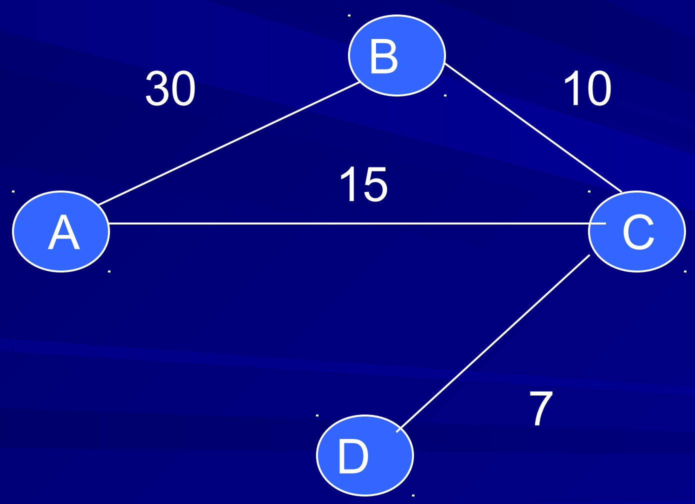
- A ***weighted graph*** is a graph where the edges have weights.
- Ex:



- The weight of {A,B} is called the distance between A and B.

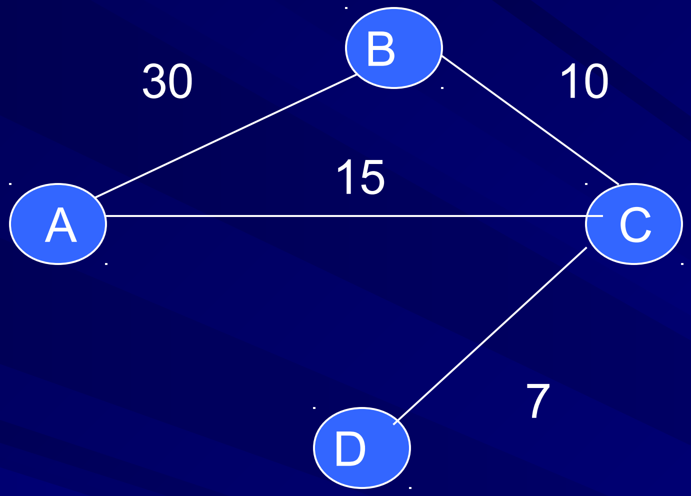
Notation

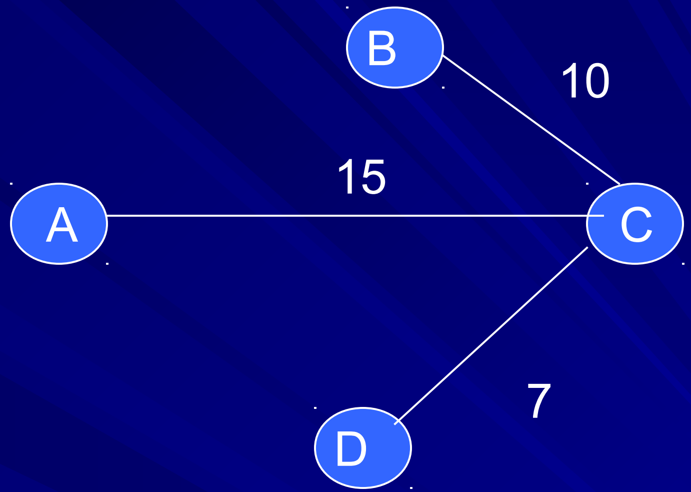
- Neighbors of C: B,A,D N(C
- Neighbors of A: B,C
- Nearest Neighbor of A: C

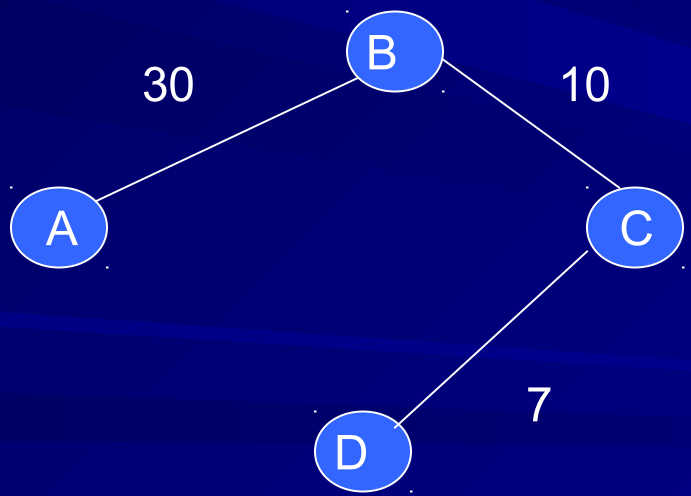


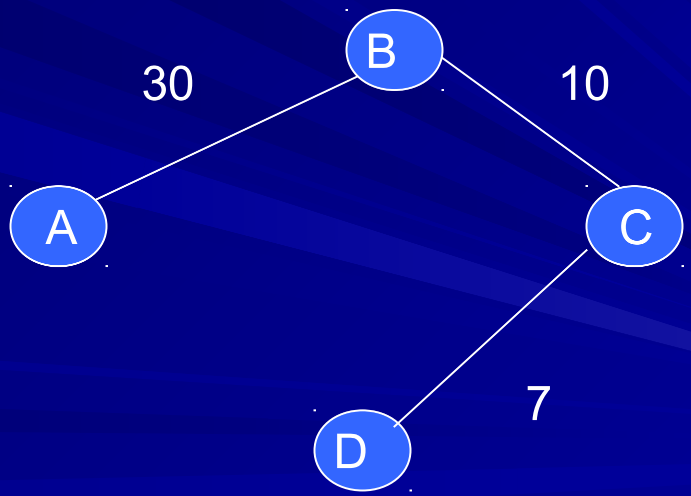
Minimal Spanning Tree

- A spanning tree for which the total weight of the edges is as small as possible

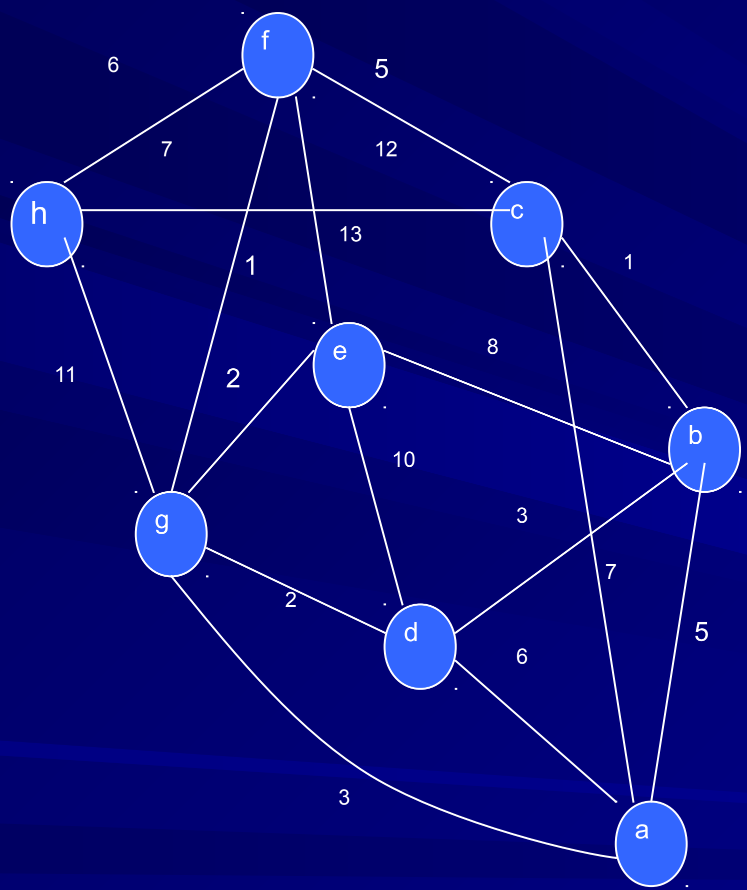

B
30
10
15
A
C
7
D

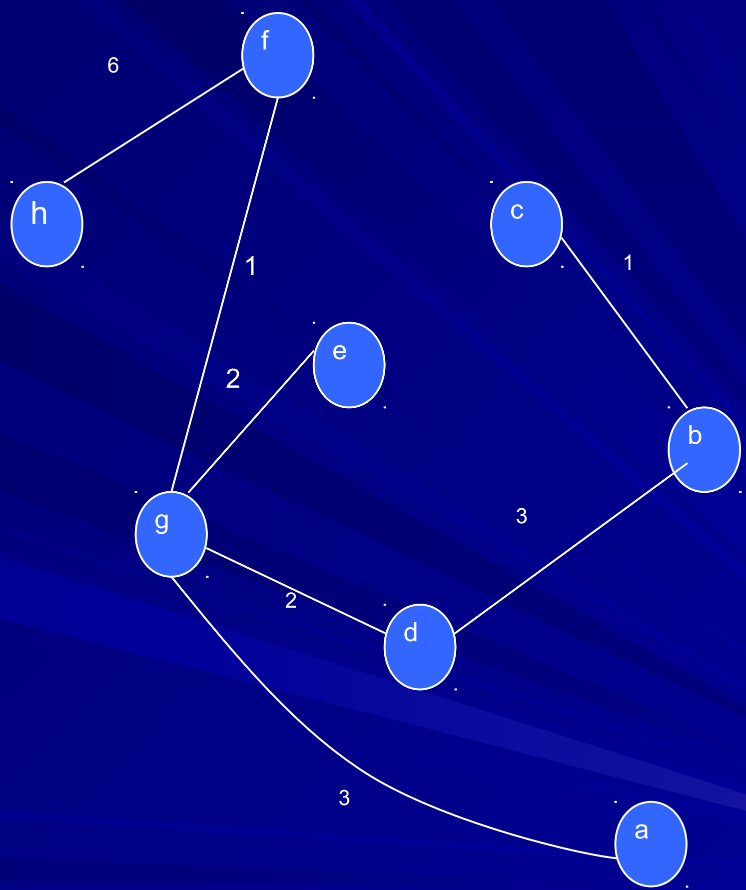

B
10
15
A
C
7
D


B
30
10
A
C
7
D


B
30
10
A
C
7
D

Prim's Algorithm





Kruskal's Algorithm