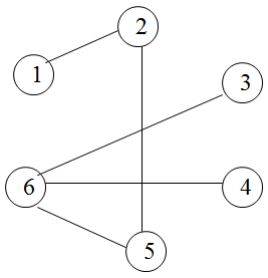
Graphs:

A graph G is denoted by G = (V, E) where
- V is the set of *vertices* or nodes of the graph
- E is the set of *edges* or arcs connecting the vertices in V

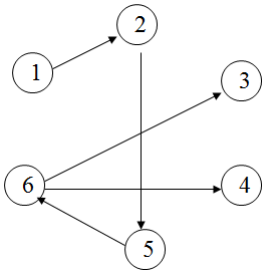
Each edge E is denoted as a pair (*v,w*) where *v,w* ϵ V

- Vertex *v* is *adjacent* to or neighbor of *w* iff (*v,w*) ϵ E
- Sometimes an edge has another component called a weight or cost. **(Weighted Graph)**.
- A path is a sequence of vertices $v_1, v_2, v_3, ... v_n$ such that (v_i, v_{i+1}) ϵ E, e.g. Path 1 to 6 is 1,2,5,6,3 where 1 is the starting point and 3 is the destination. This path has 5 vertices and 4 edges.
- Length of a path = Number of edges in the path.
- A loop is an edge from a vertex onto itself. It is denoted by (v, v).
- A simple path is a path where no vertices are repeated along the path.
- A cycle is a path with at least one edge such that the first and last vertices are the same, i.e. $v_1 = v_n$.



V = {1, 2, 3, 4, 5, 6}
E = {(1, 2),(2, 5),(3, 6),(4, 6),(5, 6)}

Edges – Undirected/Unordered



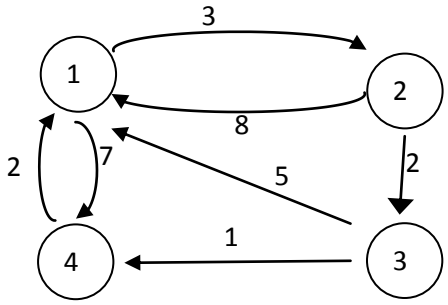
Directed Graph:

V = {1, 2, 3, 4, 5, 6}
E = {(1, 2),(2, 5),(5, 6),(6, 3),(6, 4)}

Applications of graphs:
- Driving Map
  - Edge = Road
  - Vertex = Intersection
  - Edge weight = Time and/or Distance
- Airline Traffic
  - Vertex = Cities
  - Edge = Flight between two cities
  - Edge weight = Time, Distance and/or Cost
- Computer networks
  - Vertex = Server nodes
  - Edge = Data link
  - Edge weight = Connection speed

Adjacency Matrix:

- Two dimensional matrix of size n x n where n is the number of vertices in the graph.
- a[i, j] = 0 if there is no edge between vertices i and j.
- a[i, j] = 1 if there is an edge between vertices i and j. a[i, j] = a[j, i] **(Undirected Graph)**.
- a[i, j] = weight if there is an edge between vertices i and j **(Directed Graph)**.
- Space Requirement: $O(n^2)$.
- **Problem:** The array is very sparsely populated, e.g. if a directed graph has 4 vertices and 3 edges, the adjacency matrix has 16 cells only 3 of which are 1. **(Sparse Matrix)**.

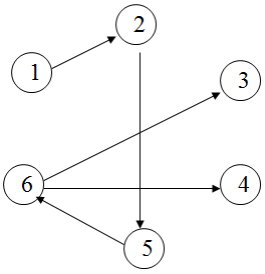


$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & 0 & 7 \\ 8 & 0 & 2 & 0 \\ 5 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

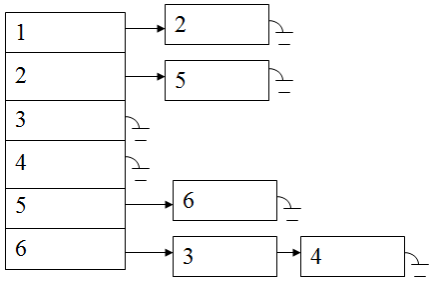
Adjacency List:

- Array of Arrays or List of Lists.
- Each vertex has an array/list entry.
- A vertex w is inserted in the list for vertex v if there is an outgoing edge from v to w.
- An adjacency list for a weighted graph should contain two elements in each nodes – one element for the vertex and the second element for the weight of that edge.
- Space requirement: O(E+V).
- Sometimes, a hash-table of lists is used to implement the adjacency list when the vertices are identified by a name (string) instead of a number.

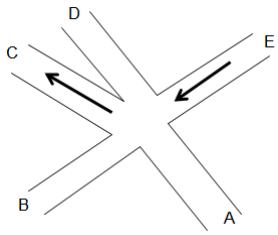
Graph:



Adjacency List (Array of Linked Lists):



Application/Example: Complicated Traffic Light Problem



An intersection

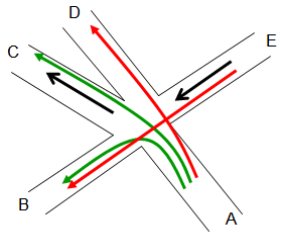
- Roads C and E are one way, others are two way.
- There are 13 permitted turns.
- Some turns such as AB and EC can be carried out simultaneously.
- Other like AD and EB cross each other and cannot be carried out simultaneously.

Solution:

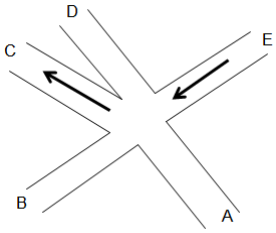
- Identify permitted turns; going straight is a “turn”.
- Make group of permitted turns.
- Make the smallest possible number of groups.
- Assign each phase of the traffic light to a group.

Turns:

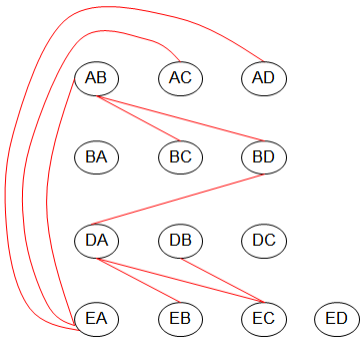
- A to B Permitted
- A to C Permitted
- A to D Permitted
- A to E
- B to A Permitted
- B to C Permitted
- B to D Permitted
- B to E
- C to A
- C to B
- C to D
- C to E
- D to A Permitted
- D to B Permitted
- D to C Permitted
- D to E
- E to A Permitted
- E to B Permitted
- E to C Permitted
- E to D Permitted



An intersection



An intersection



Partial graph of incompatible turns