Graph Representations

Instructor: Dr. Ahmed Zekri

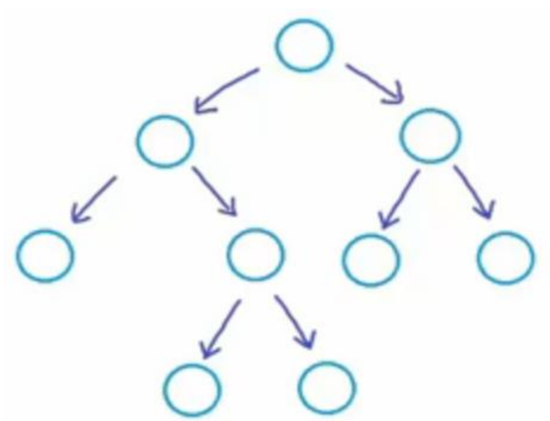
The slides are adapted from lectures of Prof. Charles Leiserson , MIT, CLRC textbook 2nd ed. & youtube lecture: Introduction to Graphs(mycodeschool)

Agenda

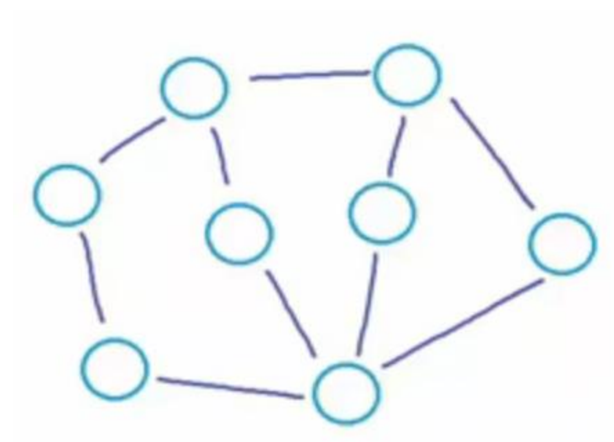
- Overview of Graphs
- Graph representation
  - adjacency list
  - adjacency matrix
  - Space complexity analysis

What is a graph?

- It is a non-linear data structure
- A tree is a graph with no cycle



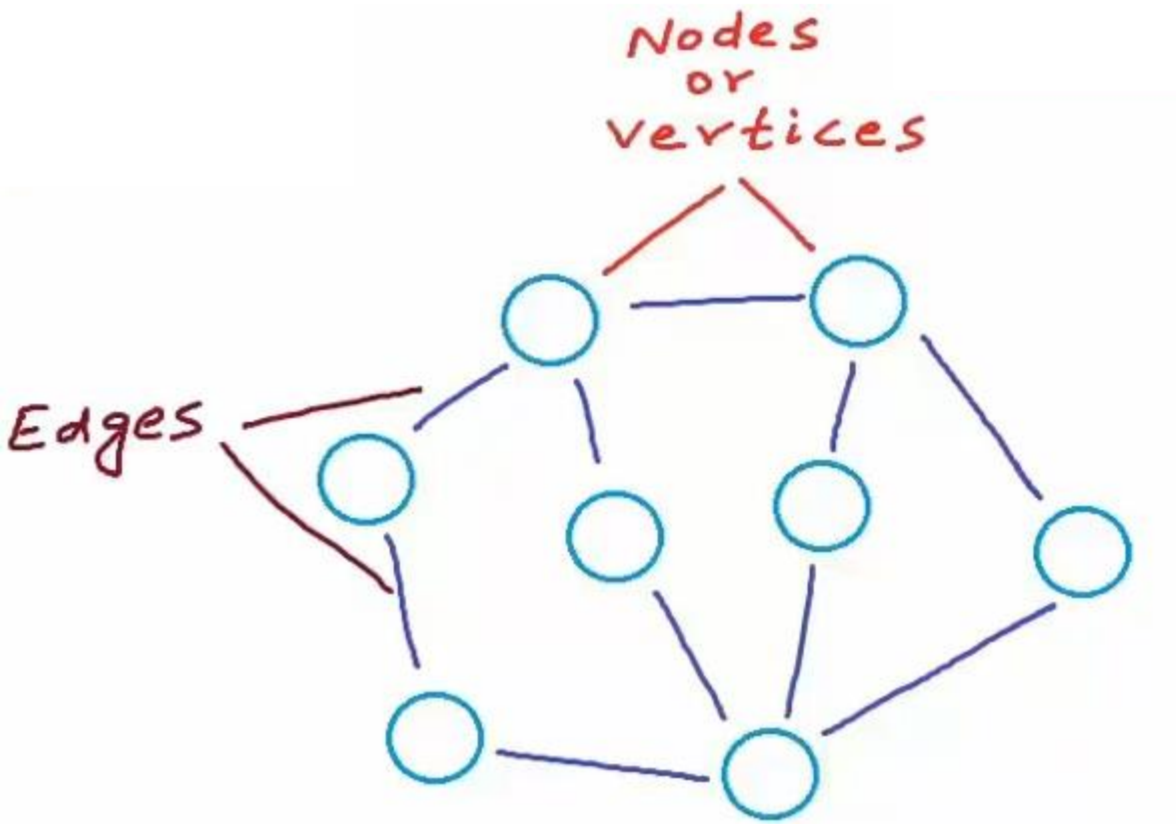
tree



graph

…

- G=(V, E)



Graph components

Graph:

A graph G is an ordered pair of a set V of vertices and a set E of edges.

$$G = (V, E)$$

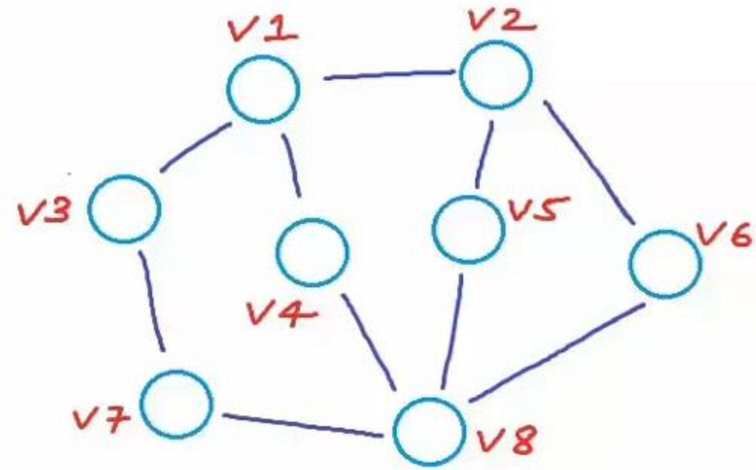
Edges:

directed

(u, v)

undirected

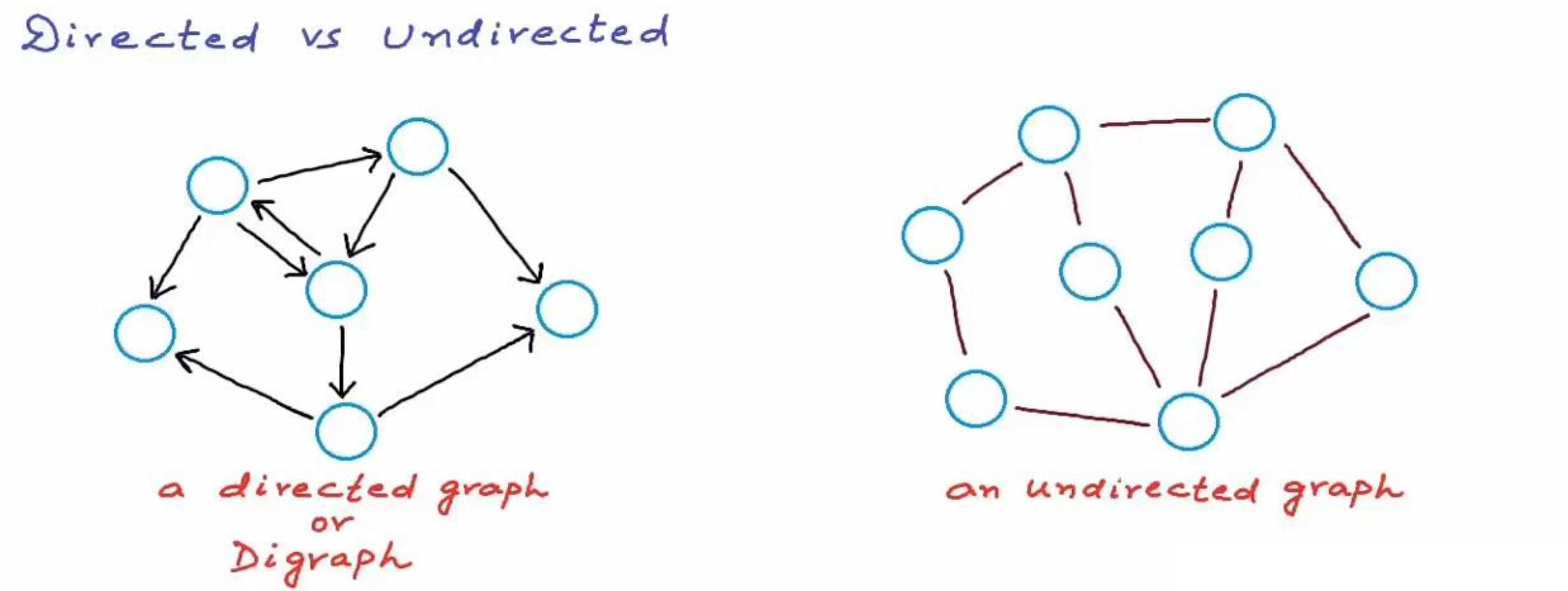
$\{u, v\}$



$V = \{v1, v2, v3, v4, v5, v6, v7, v8\}$

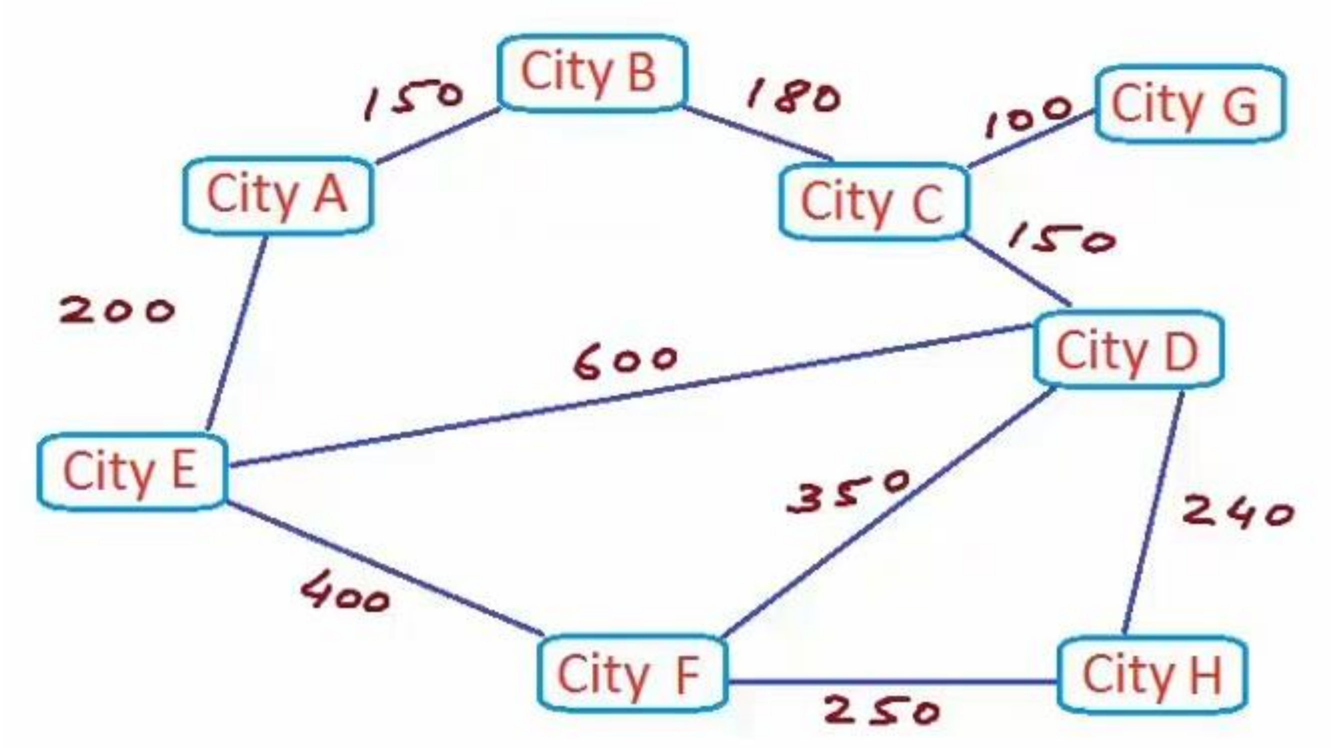
$E = \{\{v1, v2\}, \{v1, v3\}, \{v1, v4\}, \{v2, v5\}, \{v2, v6\}, \{v3, v7\}, \{v4, v8\}, \{v7, v8\}, \{v5, v8\}, \{v6, v8\}\}$

Types of graphs

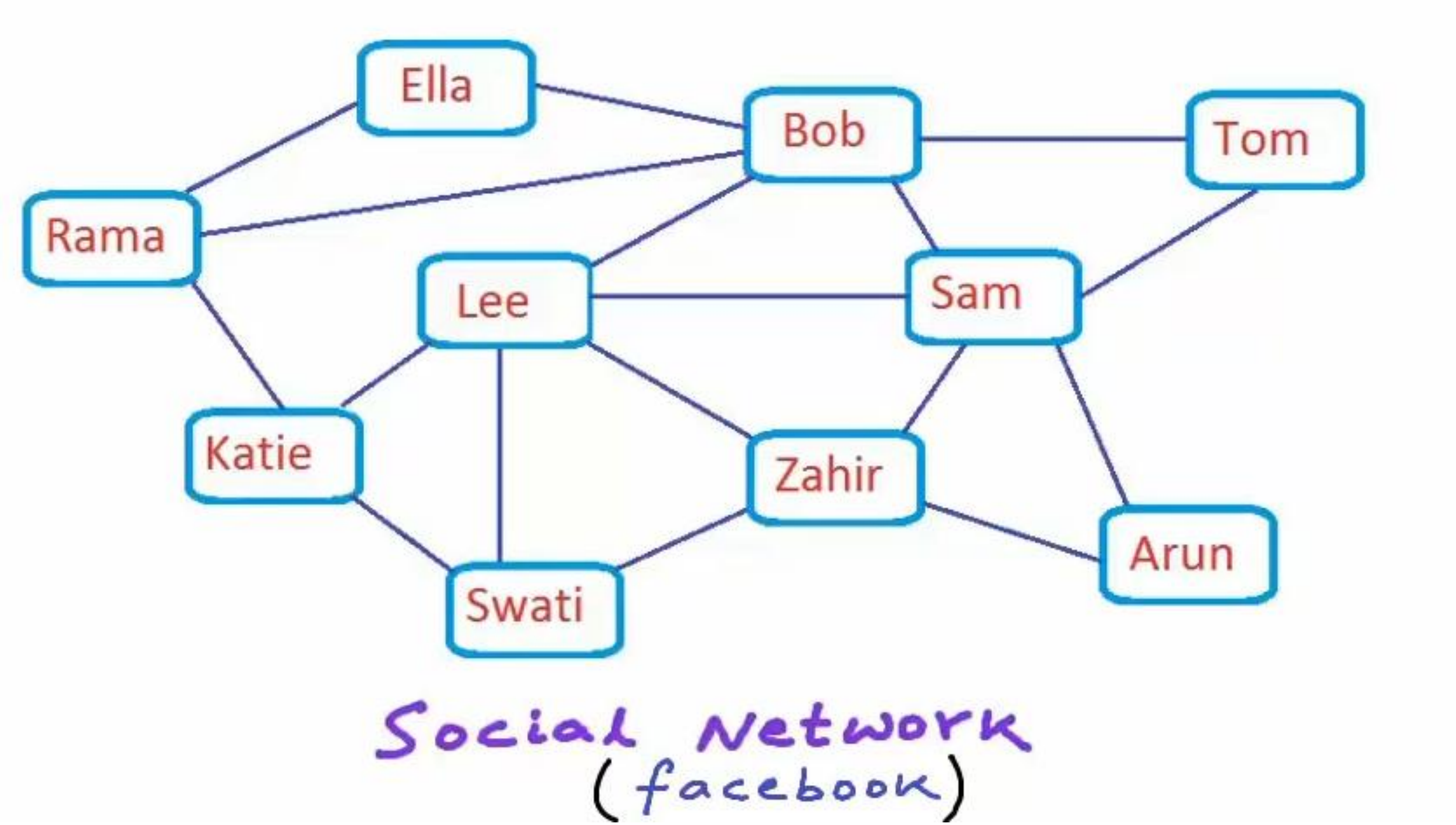


Weighted graphs

- Weight is a distance between two cities



The Facebook as a graph



Can you suggest some friends to Sam?

Graph representation

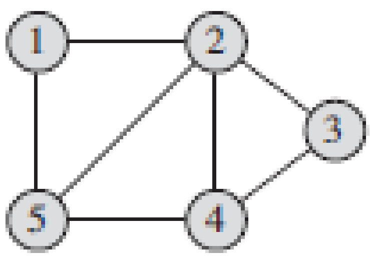
- Two explicit data structures can be used to represent a graph (directed or undirected)
  - Adjacency matrix
  - Adjacency list

The adjacency matrix

For the ***adjacency-matrix representation*** of a graph $G = (V, E)$, we assume that the vertices are numbered $1, 2, \ldots, |V|$ in some arbitrary manner. Then the adjacency-matrix representation of a graph G consists of a $|V| \times |V|$ matrix $A = (a_{ij})$ such that

$$a_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E , \\ 0 & \text{otherwise} . \end{cases}$$

Adjacency matrix



(a)

$$\begin{array}{c|ccccc|}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 0 & 1 & 0 & 0 & 1 \\
2 & 1 & 0 & 1 & 1 & 1 \\
3 & 0 & 1 & 0 & 1 & 0 \\
4 & 0 & 1 & 1 & 0 & 1 \\
5 & 1 & 1 & 0 & 1 & 0 \\
\hline
\end{array}$$

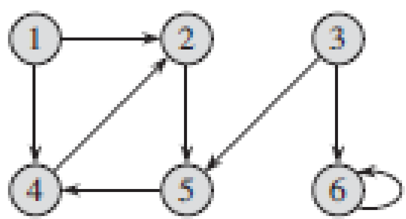
(c)

- This is an undirected graph (matrix is symmetric)
- Similar matrix for directed graph but not symmetric.

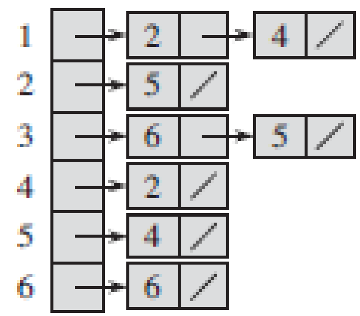
Adjacency list

- The ***adjacency-list representation*** of a graph G =(V,E) consists of an array *Adj* of |V| lists, one for each vertex in V .
- For each u ϵ V , the adjacency list *Adj[*u] contains all the vertices such that there is an edge (u,v) ϵ E.
- That is, *Adj[*u] consists of all the vertices adjacent to u in G.
- Since the adjacency lists represent the edges of a graph, in pseudocode we treat the array *Adj* as an attribute of the graph, just as we treat the edge set E. In pseudocode, therefore, we will see notation such as G.*Adj[*u].

Adjacency List



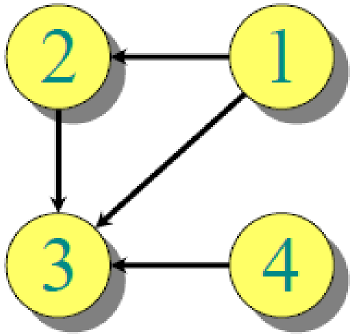
(a)



(b)

Adjacency list example

An ***adjacency list*** of a vertex $v \in V$ is the list $Adj[v]$ of vertices adjacent to v.



$Adj[1] = \{2, 3\}$
$Adj[2] = \{3\}$
$Adj[3] = \{\}$
$Adj[4] = \{3\}$

For undirected graphs, $|Adj[v]| = degree(v)$.
For digraphs, $|Adj[v]| = out\text{-}degree(v)$.

Space complexity

- The adjacency matrix:
- requires $\theta(V^2)$ memory, independent of the number of edges in the graph.

- The adjacency list:
- If G is a directed graph, the sum of the lengths of all s is |E|.
- If G is an undirected graph, the sum of the lengths of all the adjacency lists is 2|E|
- For both directed and undirected graphs, the adjacency-list representation the amount of memory it requires is $\theta(V + E)$