6.002

# Lecture 3: Graph-Theoretic Models

A Graph is a set of vertices and Edges where each edge is a connection between vertices.

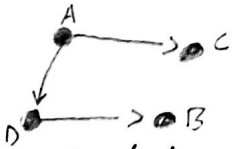
notation: $G = (V, E)$

Vertices V is a set of nodes that may have some property.

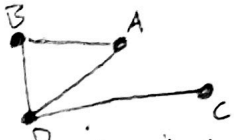
Edge E is a set of edges (arcs) each consist of pair of nodes $(s, d)$ → source, → destination

types of graph
- Directed graph
- undirected graph

types of Edges
- weighted edge
- unweighted edge



Directed graph



undirected graph

Min-flow or max-cut is trying to identify which cluster in the graph tend to have a lot of interactions within that cluster but not very many with other cluster

## Implementation of Graph

Baild Graph
- Nodes
- Edges
- Connecting nodes using edges to create Graph.

DI Graph
- Edges pass in one direction only

## Representation of Graph

### Adjacency matrix
- rows → source of nodes
- columns → destination nodes
- cell [s, d] = 1 if there is edge from s to d or 0 otherwise.

### Adjacency List
- Associate with each node a list of destination nodes

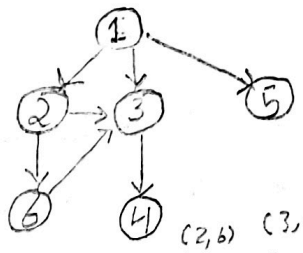
ex
$s = [d]$ means an edge from s to d exists.

# Finding Shortest Path

## Depth first Search

- Start at an initial node
- Consider all outgoing edges, in some order
- Follow the first, edge then check if goal is reached
- if not, repeat the process from new node
- Continue until either find goal node or run out of options.
  - when run out of option backtrack to the previous node and try next edge, repeating the process

S=1 D=5




(2,6) (3,4)

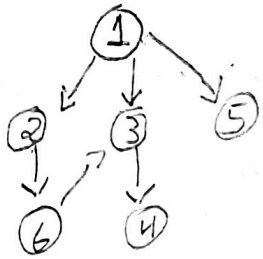
DFS = [1, 2, 6, 3, 4, 3, 5]

S (1,2) (6,3) (2,3) (1,5)

## Breadth First Search

- start an initial node
- Consider all outgoing edges in some order.
  - follow the first edge, and check see if goal node reached
- if not, try the next edge from current node
- Continue until either find goal node or run out of options.
  - when run out of options move to next node at same distance from start and repeat.
  - when run out of node option move to next level in the graph.

S=1 D=5




BFS = [1, 2, 3, 5]

S (1,2) (1,3) (1,5)

## Note

in a weighest shortest path we just need to minimize sum of weight in a path.

- DFS can be modified to this easily
- BFS can not hence we may have more than minimum number of hops.