15-210
Parallel and Sequential Algorithms and Data Structures

Lecture 9

Graphs

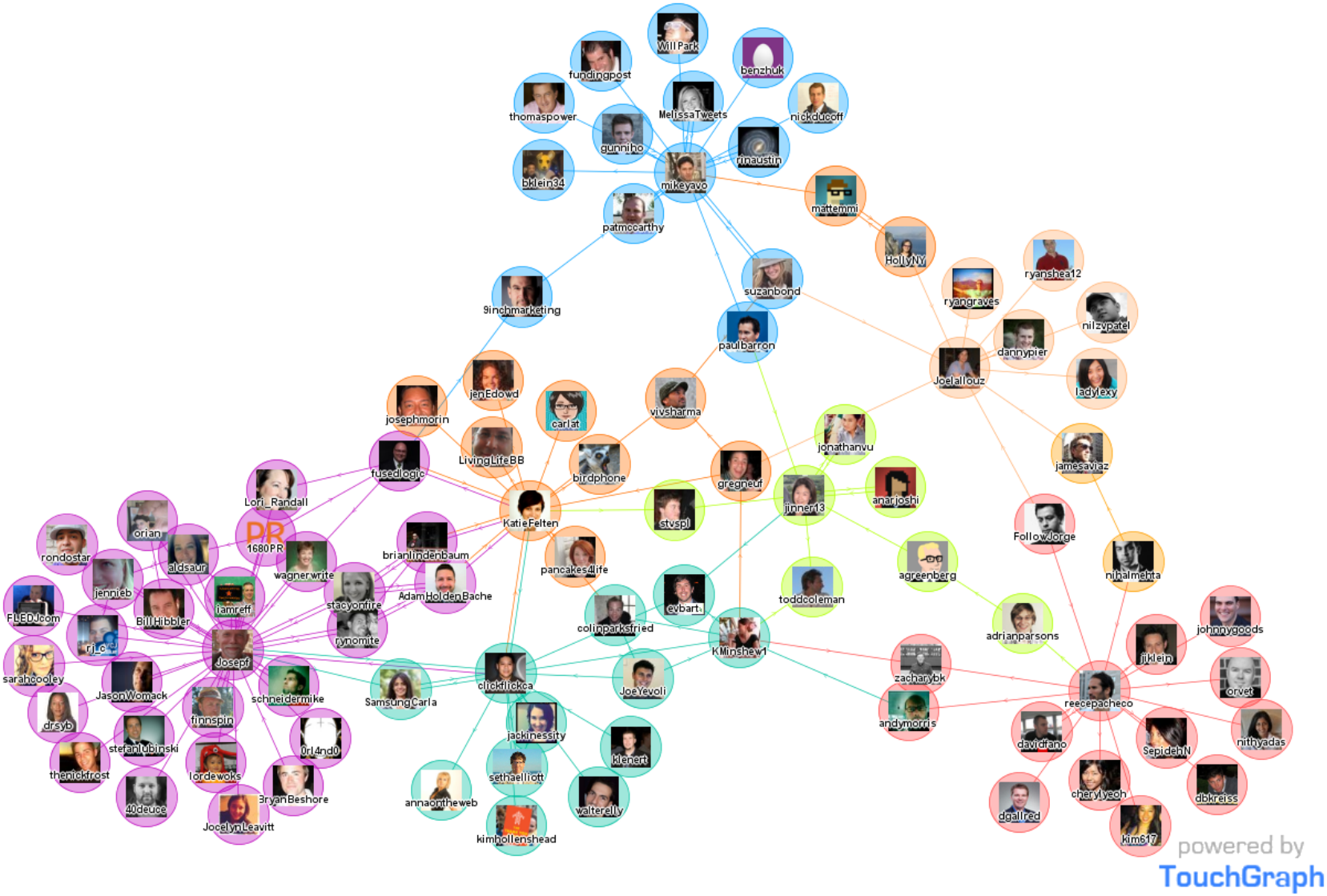
Synopsis

- Graphs
- Graph terminology/definitions
- Graph representations/costs.
- Graph search

Graphs

- Most versatile ADT in the study of algorithms
- Captures relationships between pairs of items
- A graph consists of
  - a set of V vertices/nodes
  - a set edges $E \subseteq V \times V$
- Edges represent relationships between nodes.
  - directed edges (asymmetric relationships)
  - undirected edges (symmetric relationships)
- Nodes or edges can have additional weights or values associated.

Social Networks



Social Networks - Questions

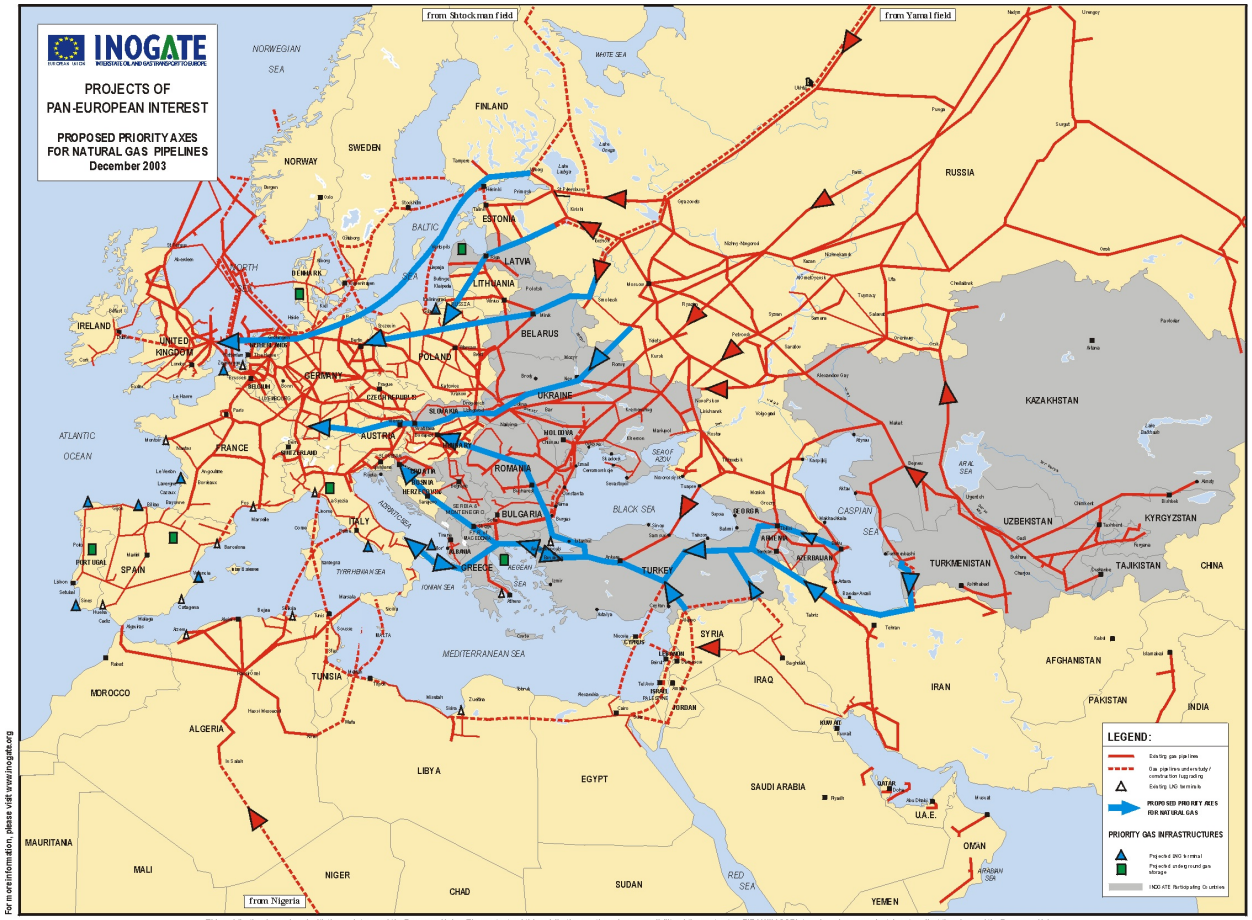
- Who is popular?
- What is the largest “clique”?
- Do I know somebody who knows X?
- What is the “diameter”?

Transportation Networks

Transportation Networks - Questions

- What is the shortest route from NYC to Los Angeles?
  - without Toll Roads?
  - without any state roads?
- What is the expected driving time from Boston to Atlanta?
  - considering traffic congestion?

Flow Networks



Flow Networks - Questions

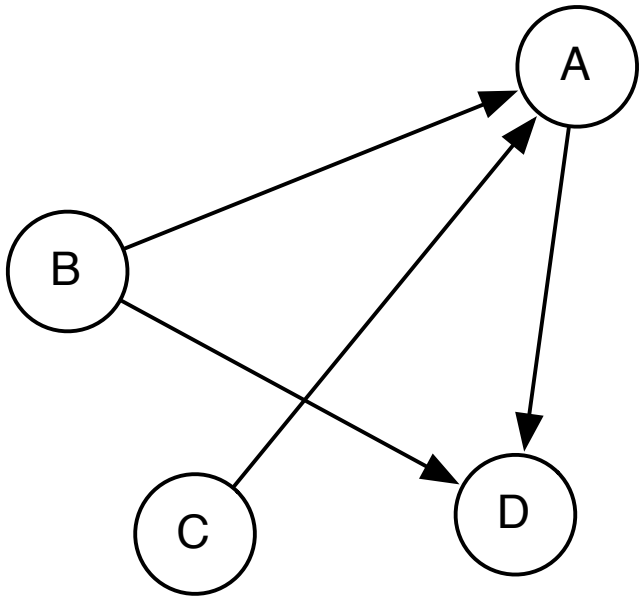
- Is it possible to send $1M$ cubic meters of gas to Paris daily?
- What is the maximum gas that can be pumped from Azerbaijan to Italy?

Other examples of Graphs

- Course prerequisite relation graphs (directed-acyclic)
- Web-page linkage graph
- Protein-protein interaction graph
- Neural networks
- Semantic networks

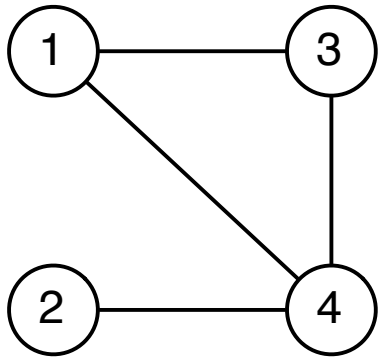
Directed Graphs

- A directed graph (digraph) is $G = (V, E)$
  - V is a set of vertices (or nodes), and
  - $E \subseteq V \times V$ is a set of directed edges (or arcs).
- Each arc is an ordered pair $e = (u, v)$
  - Arcs represent asymmetric relationships
  - A graph can have self loops (u, u)



Undirected Graphs

- An undirected graph is $G = (V, E)$
  - V is a set of vertices (or nodes), and
  - $E \subseteq V \times V$ is a set of edges
- Each edge is an unordered pair $e = \{u, v\}$
  - Edges represent symmetric relationships
  - Undirected graphs do not have self-loops.



Neighbors

- In an undirected graph, $G = (V, E)$, a vertex u is a neighbor of v if $\{u, v\} \in E$.
- In an undirected graph, $N_G(v) = \{u \mid \{u, v\} \in E\}$ is the neighborhood of v
- If U is a set of nodes,
  - $N_G(U) = \cup_{v \in U} N_G(v)$ is the neighborhood of U

Neighbors

- In a directed graph, $G = (V, E)$,
  - u is an in-neighbor of v if $(u, v) \in E$
  - u is an out-neighbor of v if $(v, u) \in E$
- In a directed graph
  - $N_G^-(u)$ is the set of in-neighbors of u.
  - $N_G^+(u)$ is the set of out-neighbors of u.
  - When we use $N_G(v)$, we mean out-neighbors.
- If U is a set of nodes,
  - $N_G^+(U) = \cup_{u \in U} N_G^+(u)$ is the out-neighborhood of U.

Node Degrees

- Undirected graphs: degree $d_G(v)$ of a vertex v is $|N_G(v)|$
- Directed graphs:
  - in-degree of a vertex v is $d_G^-(v) = |N_G^-(v)|$
  - out-degree of a vertex v is $d_G^+(v) = |N_G^+(v)|$
- We will remove subscript G if it is clear from context.

Paths

- A path is a sequence of adjacent vertices.
- For a graph $G = (V, E)$

$$Paths(G) = \left\{ P \in V^+ \mid 1 \leq i < |P|, (P_i, P_{i+1}) \in E \right\}$$

  - V^+ is denotes of sequence of length 1 or more.
  - Repeats are allowed.
- The length of a path is the number of edges.
- A path may have an infinite length.
- A simple path has no repeated vertices.
  - Often "simple" will be dropped.

Reachability

- A vertex v is reachable from a vertex u in G if there is a path starting at u and ending at v in G.
- $R_G(u)$ is the set of vertices reachable from u.
- An undirected graph is connected if all vertices are reachable from all other vertices.
- A directed graph is strongly connected if all vertices are reachable from all other vertices.

Cycles

- A cycle is a path that starts and ends at the same vertex.
- In a directed graph a cycle can have length 1 (i.e. a self loop).
- In an undirected graph we require that a cycle must have length at least three.
  - Going from u to v and back to u does not count.
- A simple cycle is a cycle that has no repeated vertices other than the start vertex being the same as the end.

Trees, Forests and DAGs

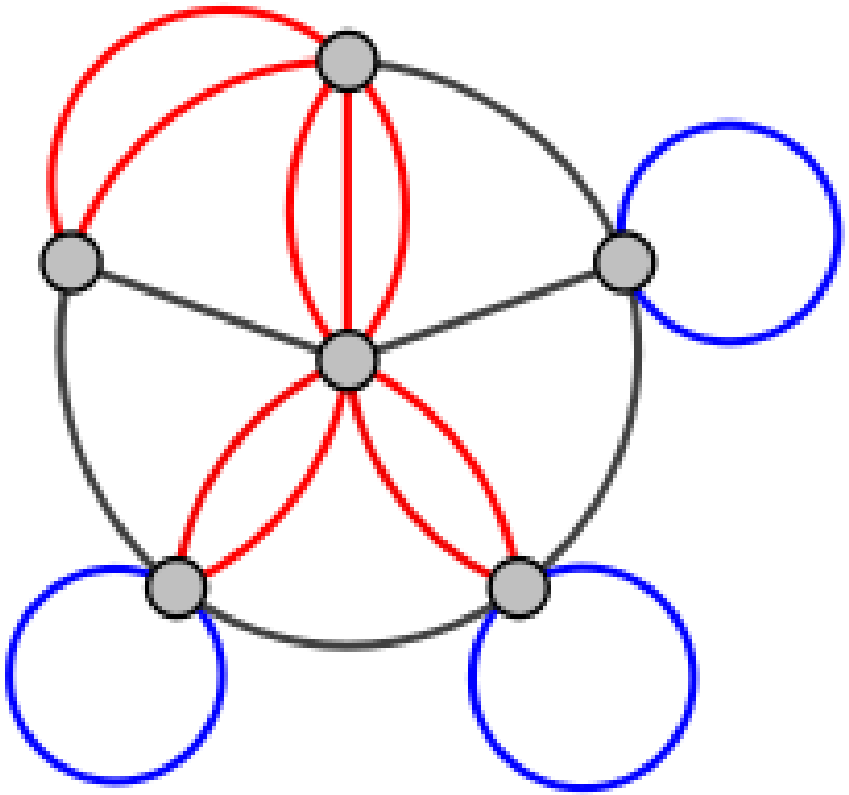
- An undirected graph with no cycles is a forest.
- If it is connected then it is a tree.
- A directed graph is a forest or tree, if it becomes a forest or tree, when all arcs are made undirected.
- In a rooted tree one node is the root.
- For a directed graph, all edges are either towards the root or away from the root.
- A directed graph with no cycles is a directed acyclic graph (DAG)

Distance and Diameter

- The distance $\delta_G(u, v)$ from a vertex u to a vertex v in a graph G is the *shortest* path (minimum number of edges) from u to v.
- The diameter of a graph is the *maximum shortest path length* over all pairs of vertices: $\text{diam}(G) = \max\{\delta_G(u, v) : u, v \in V\}$.

Multi-graphs

- Multi-graphs allow multiple edges between same pair of vertices.



Sparse and Dense Graphs

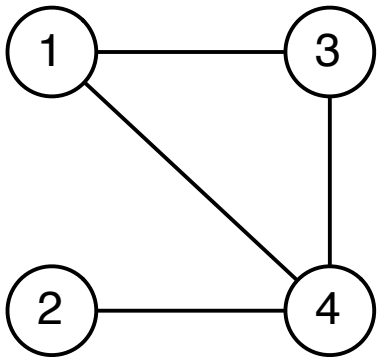
- Let $n = |V|$ and $m = |E|$.
- A directed graph can have at most n^2 edges.
- An undirected graph can have at most $\frac{n(n-1)}{2}$ edges.
- A graph is sparse if $m \ll n^2$. Otherwise it is called dense.
- In most applications, the graphs are sparse.
  - Nobody on Twitter has 10^9 followers
  - Though some have very large number– but still small when compared to n.

Operations on Graphs

1. Map over the vertices $v \in V$.
2. Map over the edges $(u, v) \in E$.
3. Map over the neighbors of a vertex $v \in V$, or in a directed graph over the in-neighbors or **out-neighbors**.
4. Return the degree of a vertex $v \in V$.
5. Determine if an edge (u, v) is in E.
6. Insert or delete vertices.
7. Insert or delete edges.

Adjacency Matrix Representation

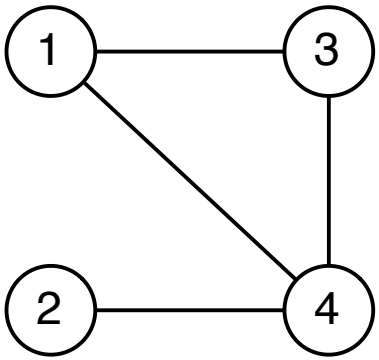
- Assume vertices are numbered $1, 2, \ldots, n$ (or $0, 1, \ldots, n-1$).
- Graph is represented by an $n \times n$ matrix of binary values in which location (i, j) is 1 if $(i, j) \in E$ and 0 otherwise.
  - For undirected graphs, matrix is symmetric and has 0's along the diagonal.

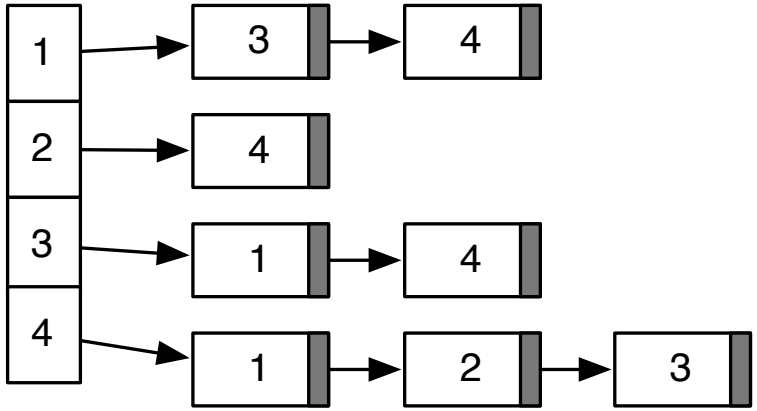


$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Adjacency List Representation

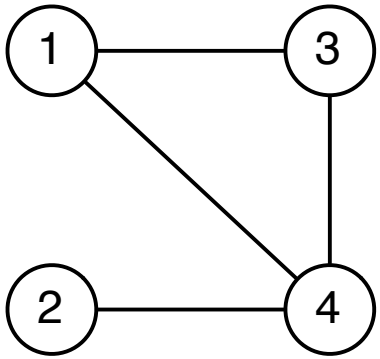
- Graph is represented by an array A of length n where each entry $A[i]$ contains a pointer to a linked list of all the out-neighbors of vertex i.
  - In an undirected graph edge $\{u, v\}$ will appear in the adjacency list for both u and v (not always necessary!)

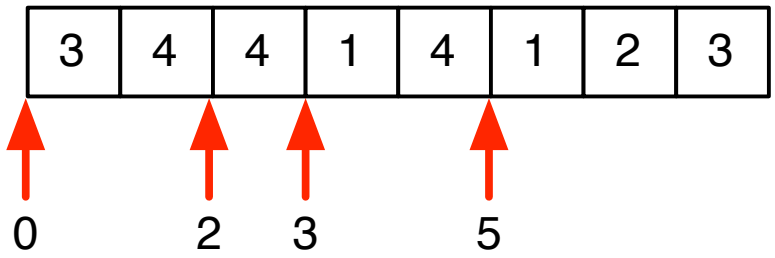




Other Representations

- Adjacency Array





- Edge List

$$((1,3),(1,4),(2,4),(3,1),(3,4),(4,1),(4,2),(4,3))$$

More Abstract Representations

- Edge Sets
  - Directed graphs: Set items are pairs (u, v) representing arcs.
  - Undirected graphs: Set items are sets $\{u, v\}$ representing edges.
- Edge Tables
  - Directed graphs: Table items are pairs $((u, v) \mapsto w_{u,v})$ representing arcs and associated values.
  - Undirected graphs: Set items are pairs $(\{u, v\} \mapsto w_{u,v})$ representing edges and associated values.

Edge Sets and Tables

- Similar to edge lists but abstracts from underlying representation.
- Search for an edge needs $O(\log m)$ work.
- Searching for neighbors is not efficient: $O(m)$ work but $O(\log m)$ span. (Why?)

Adjacency Tables

- Table items are (*key*, *value*) pairs.
- Keys are vertex/node labels.
- Values are either sets or tables
  - Sets: All neighbors node labels or out-neighbor node labels.
  - Tables: All pairs of neighbors node labels and associated edge values.
- Accessing neighbors needs $O(\log n)$ work and span.
- (Constant work) Map over neighbors needs $O(d_G(u))$ work and $O(\log d_G(u))$ span.
- Looking up an edge needs $O(\log n)$ work and span.

Cost Summary

	edge set		adj table	
	work	*span*	*work*	*span*
`isEdge`$(G,(u,v))$	$O(\log m)$	$O(\log m)$	$O(\log n)$	$O(\log n)$
`map over all edges`	$O(m)$	$O(\log m)$	$O(m)$	$O(\log n)$
`map over neighbors of` v	$O(m)$	$O(\log m)$	$O(\log n + d_G(v))$	$O(\log n)$
$d_G(v)$	$O(m)$	$O(\log m)$	$O(\log n)$	$O(\log n)$

Graph Search

- Fundamental operation of graphs
  - Start at some (set of) node(s)
  - Systematically visit all reachable nodes (only once)
- Used for determining properties of graphs/nodes
  - Connected?
  - Bipartite?
  - Node v reachable from node u?
  - Shortest path from u to v?

Graph Search

For all graph search methods vertices can be partitioned into three sets at any time during the search:

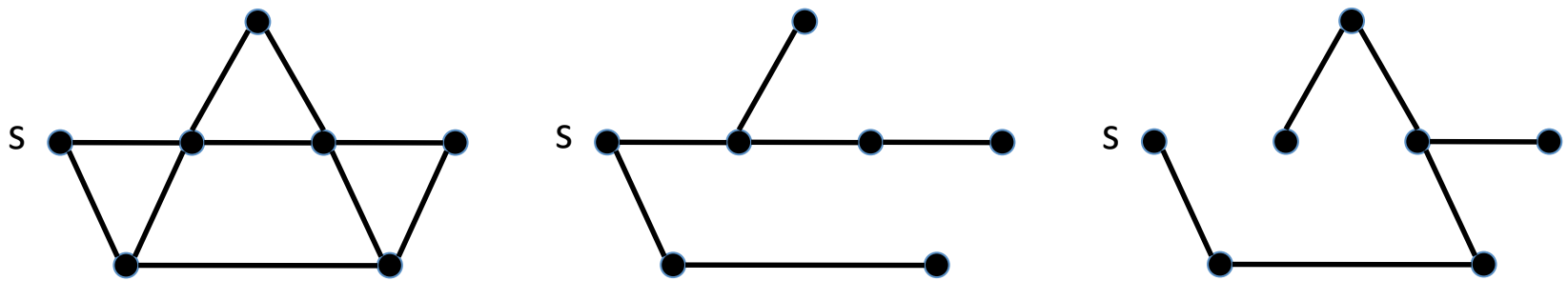
1. vertices already *visited* (X),
2. the unvisited neighbors of the visited vertices, called the *frontier* (F),
3. and the rest.

Graph Search Methods

- Breadth-first Search (BFS)
  - Parallelizable but for shallow graphs!
- Depth-first Search (DFS)
  - Inherently sequential!
- Priority-first Search (PFS)
- All reachable nodes from a source are visited, but in different orders.

Graph Search Trees

- Each search starting from a source node creates a search tree.
- We refer to the source node as the root.



- Which search schemes do these correspond to?

Summary

- Graphs
- Graph terminology/definitions
- Graph representations/costs.
- Graph search