Graphs

Lecture 13
Graphs

What is “connected” to what

Many things we deal with in computer science are graphs

Networks: humans, communication, computation, transportation, knowledge
A simple graph $G = (V, E)$, where

$E \subseteq \{ \{a, b\} \mid a, b \in V, a \neq b \}$

$V$ is the set of nodes $E$ the set of edges

$V$ non-empty and finite (for us)

Note: the “drawing” is not part of the graph, only the connectivity is
Simple Graphs

Recall graphs for relations: directed graphs with self-loops

- Each element in the domain forms a node
- Each ordered pair \((a,b)\) in the relation forms an edge
  - Edges of the form \((a,a)\) are “self-loops”

A simple graph is a symmetric, irreflexive relation

- Symmetric: An undirected edge \(\{a,b\}\) can be modelled as two directed edges \((a,b)\) and \((b,a)\)
- Irreflexive: No self-loops

In a “non-simple” graph, can allow more than one edge between any pair (multigraphs), or more generally, allow weights on edges (weighted graphs)
Examples

Complete graph $K_n$: $n$ nodes, with all possible edges between them
- $E = \{ \{a, b\} \mid a, b \in V, a \neq b \}$
- # edges, $|E| = n(n-1)/2$

Cycle $C_n$: $V = \{v_1, ..., v_n\}$, $E = \{ \{v_i, v_j\} \mid j = i+1 \text{ or } (i=1 \text{ and } j=n) \}$

Bipartite graph: $V = V_1 \cup V_2$, where $V_1 \cap V_2 = \emptyset$ (i.e., a partition), and no edge between two nodes in the same “part”:
- $E \subseteq \{ \{a, b\} \mid a \in V_1, b \in V_2 \}$
- e.g., $C_n$ where $n$ is even

Complete bipartite graph $K_{n_1,n_2}$: Bipartite graph, with $|V_1| = n_1$, $|V_2| = n_2$ and $E = \{ \{a, b\} \mid a \in V_1, b \in V_2 \}$
- # edges, $|E| = n_1 \cdot n_2$

Later: Hypercube, Trees
Graph Isomorphism

$G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there is a bijection $f: V_1 \rightarrow V_2$ such that \{u,v\} $\in$ $E_1$ iff \{f(u),f(v)\} $\in$ $E_2$

Computational problem: check if two graphs (given as adjacency matrices) are isomorphic

- Can rule out if certain "invariants" are not preserved (e.g. $|V|, |E|$)
- In general, no "efficient" algorithm known, when graph is large
- Some believe no efficient algorithm exists!
Degree of a node

Given a simple graph $G = (V,E)$, for each node $v \in V$, the degree of $v$ is the number of edges incident on $v$.

Formally, $\deg(v) = | \{ u : \{u,v\} \in E \} |$.

Counting edges in two different ways: $2 \cdot |E| = \sum_{v \in V} \deg(v)$.

Degree sequence: sorted list of degrees. (e.g.: 0,1,2,2,3)

Degree sequence invariant under isomorphism.
A graph is said to be d-regular if all nodes have degree d. How many edges are there in a 3-regular graph with 6 nodes?

A. Such a graph doesn’t exist
B. 18
C. 12
D. 9
E. It depends on the graph

\[ e.g., K_{3,3} \]
\[ n \cdot d/2 \text{ (unless } n \text{ & } d \text{ odd) } \]
A subgraph of $G = (V,E)$ is a graph $G' = (V',E')$ such that $V' \subseteq V$ and $E' \subseteq E$.

To get a subgraph: Remove zero or more vertices, remove all edges incident on them, and further remove zero or more edges.

Induced subgraph: omit the last step.
Walks, Paths & Cycles

A **walk** (of length \( k, k \geq 0 \)) from node \( a \) to node \( b \) is a sequence of nodes \( (v_0, v_1, \ldots, v_k) \) such that

- \( v_0 = a, v_k = b \)
- for all \( i \in \{0, \ldots, k-1\} \), the edge \( \{v_i, v_{i+1}\} \in E \)

Length is the number of edges in a walk. Could be 0.

If a walk has no node repeating, then it is called a **path**.

If a walk of length \( k \geq 3 \) has \( v_0 = v_k \), but no other two nodes are equal, then it is called a **cycle**.

**Note:** we require a cycle to be of length at least 3.

A graph is **acyclic** if it has no cycles (i.e., no \( C_k \) is a subgraph of \( G \)).
Connectivity

Given a graph G, whether there is a path between two nodes u and v is an important question regarding G.

u is said to be connected to v if there is such a path.

Prove: u connected to v iff there is a walk from u to v.

Relation Connected(u, v) is an equivalence relation.

Reflexive, Symmetric and Transitive.

Equivalence classes of this relation are called the connected components of G.

Walks can be spliced together to get walks.
Distance

Shortest walk between nodes $u$ and $v$ is always a path.

Shortest path is of great interest in many applications.

- e.g., nodes correspond to locations on a map and edges are roads, optic fibers etc.
- Also, graph can be used to model probabilistic processes, with shortest path indicating the most likely outcome.

Length of the shortest path between $u$ and $v$ is called the distance between $u$ and $v$ ($\infty$ if no path).

$$\min_{W: u-v \text{ walk}} \text{Length}(W)$$

Diameter is the largest distance in a graph (can be $\infty$).

$$\max_{u,v} \text{Distance}(u,v) = \max_{u,v} \min_{W: u-v \text{ walk}} \text{Length}(W)$$