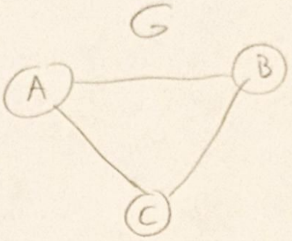
A <u>graph</u> is an abstraction of connections between objects.

Formally, it's a tuple of a finite set of vertices and edges $G = (V, E)$, where $E \subseteq \{\{u,v\} \mid (u,v) \in V^2 \wedge u \neq v\}$.

Ex)



$V = \{A, B, C)$

$E = \{\{A,B\}, \{B,C\}, \{A,C\}\}$

For edge $e = \{A, B\}$, vertices A and B are the <u>endpoints</u> of e. We also say e <u>joins</u> A and B. Moreover, e is <u>incident</u> upon both vertices.

Usually, we require that an edge connects distinct vertices, but occasionally we allow <u>loops</u>, where an edge connects a vertex to itself.

The <u>degree</u> of a vertex is the number of edges incident upon it.

Thm) The number of vertices with odd degree is even.

Pf)

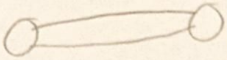
$$2|E| = \sum_{v \in V} \deg(v) = \sum_{\substack{v \in V \\ \deg(v) \text{ even}}} \deg(v) + \sum_{\substack{v \in V \\ \deg(v) \text{ odd}}} \deg(v)$$

Since $\sum_{\substack{v \in V \\ \deg(v) \text{ even}}} \deg(v)$ is clearly even, it follows that $\sum_{\substack{v \in V \\ \deg(v) \text{ odd}}} \deg(v)$ must be as well. $\square$

The empty graph is $(\emptyset, \emptyset)$.

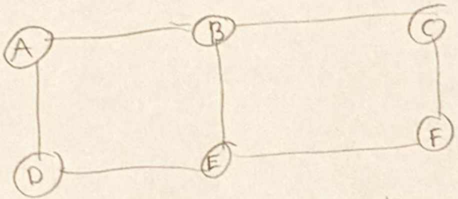
Sometimes we allow duplicate edges. When we do, we call it a multigraph.

Ex)



A path is a sequence of distinct vertices $v_1 \dots v_n \in V$ such that $\forall\, 1 \leq i < n$, $\{v_i, v_{i+1}\} \in E$. We write $v_1 \to v_n$ to denote a path from v_1 to v_n. A walk is a path without the uniqueness requirement.
A cycle is a path that ends where it begins (of length >1).
A circuit is a walk that begins where it ends (of length >1).

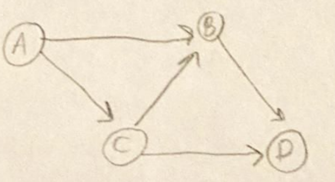
Ex)



- ABCF and ADEF are both paths from A → F.
- ABEDA is a cycle.

A directed graph (digraph) is a graph where edges go from one vertex to another. Formally, we have $G = (V, E)$, where $E \subseteq \{(u,v) \mid (u,v) \in V^2 \wedge u \neq v\}$. Notice that edges here are ordered pairs, emphasis on ordered.

Ex)



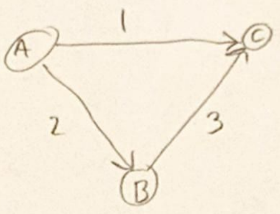
- ABD and ACBD are paths A → D but ABCD is not.
- There are no cycles.

A <u>weighting</u> on a graph $G=(V,E)$ is a function $w: E \to \mathbb{R}$.

Often we restrict weights to nonnegative, positive, or integral values.

A weighting together with a graph is a <u>weighted graph</u>.

Ex)



- Path AC has total weight 1
- Path ABC has total weight 5

Two vertices u,v in a graph are <u>connected</u> if there is a path $u \to v$.

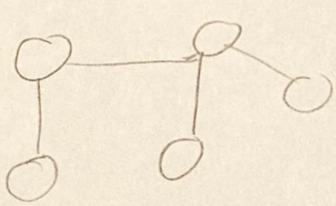
In a directed graph, we say u and v are <u>weakly connected</u> if a path $u \to v$ or $v \to u$ exists. They are <u>strongly connected</u> if both paths exist.

We say a graph is connected if every $u,v \in V$ are connected.

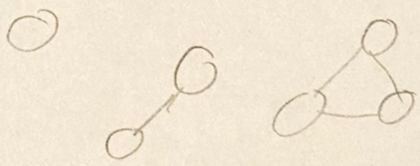
Note that a subgraph may be connected, which we call a <u>component</u>.

Ex)

Connected



3 Connected Components



Similarly, a digraph is <u>weakly connected</u> if replacing its edges with undirected ones leaves it connected.

It is <u>strongly connected</u> if $\forall u,v \in V$, $\exists$ a (directed) path $u \to v$.

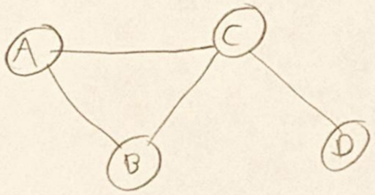
The <u>indegree</u> of $v \in V$ in a digraph is the number of incoming edges incident on v. The <u>outdegree</u> is the number of outgoing edges incident on v.

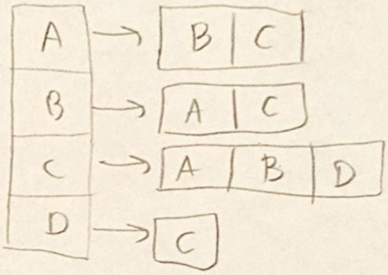
Now that we have basic vocab, how do we store the general graph. Certain types of graphs are more efficiently stored in other ways, but we have 2(and $\frac{1}{2}$) primary ways to store any graph.

Adjacency Lists

For each $v \in V$, we can store a list of vertices adjacent ot v. Typically we keep these lists all in one place, but we can also store them with their vertex.

Ex)

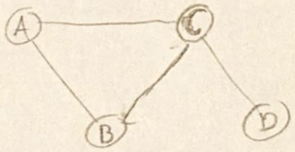




Adjacency Matrix

For each $u,v \in V$, we store a 1 (or weight) iff v is adjacent to u at $A_{u,v}$ in a $|V| \times |V|$ matrix. Note that if the graph is undirected, the matrix is symmetric.

Ex)



$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & 1 & 0 \\ B & 1 & 0 & 1 & 0 \\ C & 1 & 1 & 0 & 1 \\ D & 0 & 0 & 1 & 0 \end{array}$$

Trade Offs

- ALs are more memory efficient, especially in sparse graphs.
- AMs are more efficient for lookups and modifications, even if we keep an AL sorted.