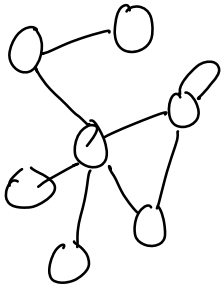
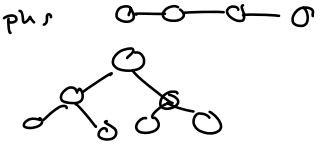
Graphs

Is nodes connected by edges

ex)



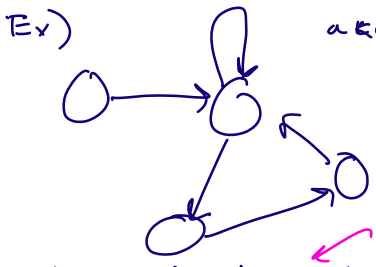
- linked lists are graphs
- trees are graphs



Def A undirected Graph $G = (V, E)$ where V is a set of nodes (vertices) and E is a set of unordered pairs of V (i.e. $E \subseteq V^2$). We say $\exists$ an edge between $v_1, v_2 \in V$ if $(v_1, v_2) \in E$ or $(v_2, v_1) \in E$.

Directed Graph $G = (V, E)$ where edges are ordered. So $(v_1, v_2) \in E \not\Rightarrow (v_2, v_1) \in E$

Ex)

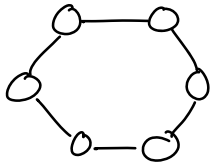


aka $v_1 \to v_2$ doesn't mean $v_2 \to v_1$.

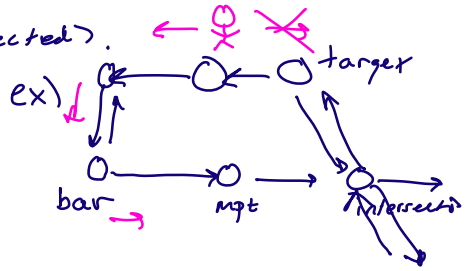
(not really a thm, but redefinition).

Thm A directed G such that for every edge $(v_i, v_j) \in E$, then $(v_j, v_i) \in E$ we say G is undirected (or can be seen as undirected).

ex)



benzene (undirected)



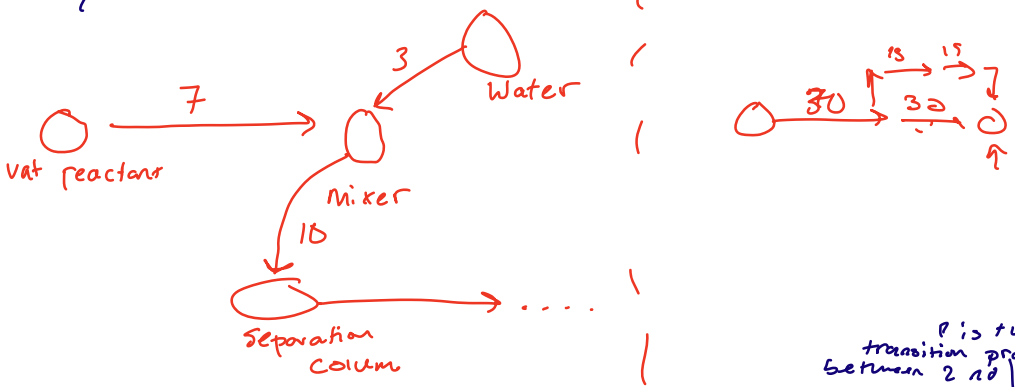
Examples

- → Molecules
- → compiler treat your program as graphs
- → Maps
- → Machine learning computing gradients of neural networks
- → Lots of things!

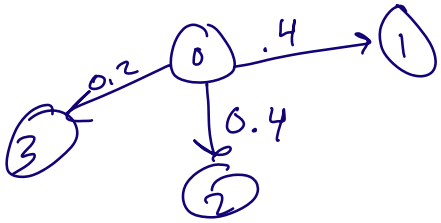
Def We say a Graph G has weighted edges if $\exists f: E \to \mathbb{R}$ so that every edge $e \in E$ is associated with some weight $f(e)$. We say $f(e)$ is the flow or weight of an edge.

Similar for nodes

ex)



Def A random walk on a probability-weighted graph $G(V, E, P)$ (P is the transition probability between 2 nodes that are connected) takes steps from $v_i \in V$ to $v_j \in V$ based on $P(v_i, v_j)$ iff $\exists e \in E$ with (v_i, v_j).



If a walker starts on node 0, there a 40% chance at $t = 2$ its on node 1.

Def A neighbor of a node $v \in V$ is another node $u \in V$ s.t. $\exists e$ from $v \to u$. The set of all neighbors is written $\mathcal{N}(v) = \{(v,u) : (v,u) \in E\}$.

Def We say adjacency matrix of a Graph is a matrix representation of a graph such that a_{ij} of a matrix is

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

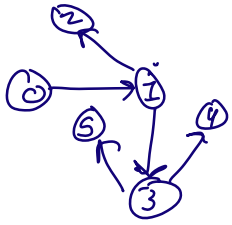
For a weighted graph

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

lemma An adjacency for a undirected graph is symmetric, Meaning $a_{ij} = a_{ji}$ or $M^T = M$.

Def A a graph has self loops if $a_{ii} > 0$.

ex)



$V = \{0, 1, 2, 3, 4, 5\}$

$E = \{(0,1), (1,2), (1,3), (3,5), (3,4)\}$.

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

ex) K_3

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

Symetric.

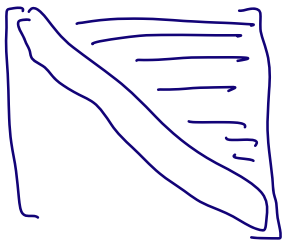
Def K_n is the undirected complete graph on n nodes.

Def We say a graph is complete if $\forall v \in V$ and $\forall u \in V$ s.t. $u \neq v$ $\exists (u,v) \in E$.

$K_2 \to$ 

$a_{ij} > 0$ if $i \neq j$.

Q. How many edges does a K_5 have?



$$\frac{5^2-5}{2} \rightarrow K_5$$

in gen $\frac{n^2-n}{2} \rightarrow \frac{n(n-1)}{2}$

Data Structure

→ vectors of edges
→ linked data structure
→ 2D matrix
→ Adjacency List

ex) Linked graph structure

```
struct graphNode {
    std::vector<graphNode*> neighbors;   // * : pointer
};
struct Graph {
    graphNode* root;
};
```

Advantages
→ really easy to add a node

Disadvantage
→ arbitrary of root to select
→ really impossible with directed graphs
→ how detect a cycle??

Matrix

advantage
→ O(1) look up for edge
→

disadv
→ $O(n^2)$ Memory
→ adding and removing would also be n^2.

Adjacency Form

```
struct Graph {
    std::map<int, std::list<int>> g;   // vertices (nodes) ← ; a list of its neighboors
};
```

adj listin

3	4
3	5
6	7
8	2

Verti

3 → 4, 5
4 → 3
5 →
6 → 1
8 → 2

Average case ore