3/16

Graphs and Trees:

A graph G with n vertices (nodes) and m edges (arcs) consists of a vertex set $V(G)=\{v_1,\dots v_n\}$ and an edge set $E(G)=\{e_1,\dots e_m\}$, where each edge e is an unordered pair of vertices. We write uv, or $\{u,v\}$ for an edge between u and v. If $uv\in E(G)$, then u and v are adjacent. The vertices contained in an edge e are its end points.

We can visualize a graph on paper by assigning a point to each vertex and drawing a curve for each edge between the points representing its endpoint. The vertices can be any objects like cities in a country, points in the plane, etc and edges are showing relation between the two objects. The graphs can model lots of problems (Graphs can also modeles people and their friendship relation)

A Graph $G=(V,E)$ will be considered often in this course.

A directed graph or digraph is graph in which its edges are ordered pairs. The edges are represented as uv or (u,v) which ~~if and only if~~ means there is an edge from u to v.

Some times, we consider more general models that allow repeated edges (multi edges) or edges with both endpoints the same (loops). such graphs are called multi-graphs.

In this cure, we usually consider simple graphs with at most $\binom{n}{2}=\Theta(n^2)$ edges.

let $G=(V,E)$ be an undirected graph. An induced subgraph of G is a graph $H=(V',E')$ such that $V'\subseteq V$ and E' includes all edges in E both of whose incident vertices are in V'. If just $E'\subseteq E$ then it is a subgraph of G.

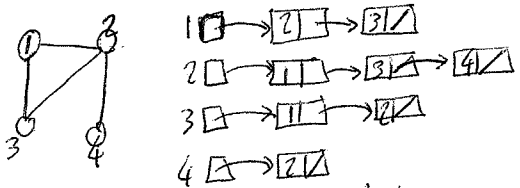
A degree of a vertex v is represented by $d_v(G)$ or $\deg_v(G)$ and is the number of vertices adjacent to that vertex. Thm: $\sum_{v\in V}\deg_v(G)=2|E|$. proof by double counting.

Representation of Graphs for Algorithms:

There are two ways to represent graphs: As a collection of adjacency lists or as an adjacency matrix. Adjacency list is preferred for sparse graphs, having small number of edges. Adjacency matrix is preferred for dense graphs, having large number of edges ($\Theta(n^2)$). ($O(n)$)

The adjacency-list representation of a graph $G=(V,E)$ consists of an array Adj of

of $|V|$ lists, one for each vertex of in V. For each $u \in V$, the adjacency list $Adj[u]$ contains pointeres to all vertices v such that there is an edge $(u,v) \in E$ (the elements of the list can be in an arbitrary order or a sorted order.)



Adjacency list

	1	2	3	4
1	0	1	1	0
2	1	0	1	1
3	1	1	0	0
4	0	1	0	0

Adjacency matrix

whether a graph is directed or not the adjacency-list representation has the memory amount $O(V+E)$. The disadvantage: determine if an edge (u,v) is in the graph can take $O(deg_V(G))$.

For the adjacency-matrix representation of a graph $G=(V,E)$, we assume that the vertices are numbered $1, 2, \ldots |V|$ in some arbitrary manner. The adjacency matrix then is a $|V| \times |V|$ array $A=(a_{ij})$ such that $a_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$

The adjacency matrix of a graph requires $\Theta(V^2)$ memory independent of the number of edge. However checking whether $(u,v) \in E$ is in $\Theta(1)$.

our graphs can be <u>weighted</u> also (for which each edge has an associated weight) and can be easily represented by both methods.

A <u>complete graph</u> or a <u>clique</u> is a simple graph in which every pair of vertices forms an edge and thus $|E| = \binom{n}{2}$.

An <u>independent set</u> is a graph is a vertex subset $S \subseteq V(G)$ such that the induced subgraph $G[S]$ has no edges.

The <u>complement</u> of a simple graph, written as $\bar{G}$, is a graph the same vertex set as G such that u,v are adjacent in $\bar{G}$ if and only if u and v are <u>not</u> adjacent in G.

Example: Job assignments and bipartite graphs: suppose we have m jobs and n people and each person can do some of the jobs. Can we make assignments to fill the jobs. we model the available assignments by a graph having a vertex for each job and each person, putting job j adjacent to person p if p can do j. (if each person can do only one job, the problem above is the matching problem).

A graph is bipartite if its vertex set can be partitioned into at most two independent set V_1, V_2 (like the person-job graph above). Then we represent it as $G(V_1, V_2, E)$.

A complete bipartite graph is a bipartite graph in which the edge set consists of all pairs having a vertex from V_1 and V_2.

For bipartite graphs, we can have a simpler adjacency matrix as follows: we have a matrix (array) of size $|V_1| \times |V_2|$, name/call it $a(i,j)$ such that $a_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$

Note that this array is not symmetric any more.

A walk of length k is a sequence $v_0, v_1, \ldots, v_k$ of vertices such that $e_i = \{v_{i-1}, v_i\} \in E$. A path is a walk with no repeated vertex. A cycle is a walk such that the first and the last vertex are the same, i.e., $v_0 = v_k$. A cycle is simple if there is no repeated vertex.

Thm: If there is a walk between u and v in G, then there is a path between them as well.

An undirected graph is connected if every pair of vertices is connected by a path.

The connected components of a graph are the equivalence classes of vertices under the "connected" relation.

A tree is a connected undirected graph with no cycle. A DAG (directed acyclic graph) is a directed graph with no cycles.

Properties of trees: Let $G=(V,E)$ be an undirected graph. The following statements are equivalent.

1) G is a tree
2) G is connected and $|E| = |V|-1$
3) G has no cycle and $|E| = |V|-1$

In particular:

Thm: If G is a tree then $|E| = |V|-1$.

Proof: First note that there is a vertex which is a leaf (a vertex of degree one) since there is no cycle. Then proof by induction on $|V|$.

A rooted tree is a tree in which one of the vertices is distinguished from the others, called the root. We often call vertices of a rooted tree (in general directed graphs) nodes.

If the last edge on the path from the root r of a tree T to a node x is (y,x), the y is the parent of x and x is the child of y. A node with no children is a leaf. A node which is not leaf is an internal node.

Consider a node x in a rooted tree T with root r. Any node on the unique path from r to x is called an ancestor of x. If y is an ancestor of x, then x is a descendant of y.

Rooted trees are often used to show hierarchical (data) structures. Because of this we can make the tree directed toward leaves or at least put the root on top of the tree. Then the root is connected to other nodes, which are at the level one of the hierarchy; they, in turn, are connected to other nodes at level 2 and so on.

The number of children of a node x in a rooted tree is called the out-degree of x.

The length of the path from the root r to a node x is the depth of x in T.

The largest depth of any node in T is the height of T.

Rooted trees of out-degree 2 (every vertex has out-degree ≤ 2) are called binary trees. In this case we identify children by left (for first) and right (for second).

The common representation of trees is the linked-list representation in which each node keeps pointer (as an array for binary trees; otherwise linked-list) to its children and also a pointer to its parent.

We see another implicit representation for binary trees in the next session.