Lecture 15

Readings Sections 22.1, 22.2

Self test Exercises 22.1-1, 22.1-2, 22.2-1

Graphs

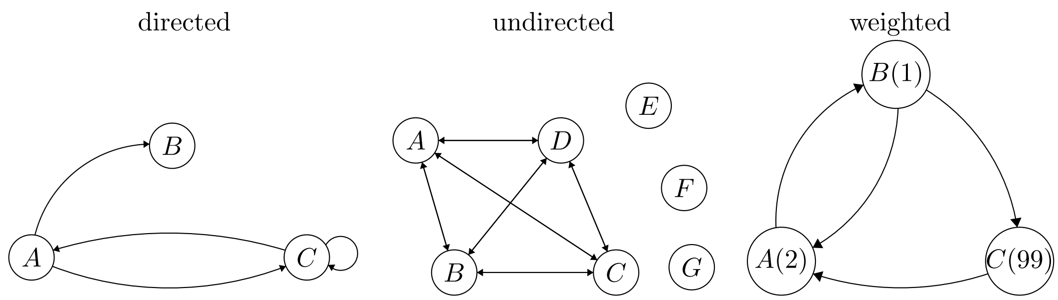
A graph $G = (V, E)$ consists of a set of "vertices" (or "nodes") V and a set of "edges" (or "arcs") E.

In a **directed** graph, each edge is a pair of two vertices (u, v) is considered different from the pair (v, u), self-loops of the form (u, u) are allowed.

In an **undirected** graph each set of two vertices $\{u, v\}$ (so $\{u, v\}$ and $\{v, u\}$ are the same) and self loops are disallowed.

A **weighted** graph is either directed or undirected. Each edge $e \in E$ is assigned a real number $w(e)$ called its weight.

A story to solve with a graph:
There is a party. 4 couples attend the party. Hand shaking occurs at the party. Hand shaking is reciprocal. Nobody shakes hands with their date. No duplicate answers when a person P asks everyone else how many hands did they shake. How many hands did P shake?



Standard operations on graphs

- add a vertex; remove a vertex; add an edge; remove and edge
- edge query
  - given two vertices u, v find out if a directed edge (u, v) or undirected edge $\{u, v\}$ is in the graph
- neighbourhood
  - given a vertex u in an undirected graph, find the set of vertices v such that $\{u, v\}$ is an edge
- in-neighbourhood, out-neighbourhood
  - apply to directed graph
  - in-neighbourhood $\rightarrow$ given a vertex u in a directed graph, set of vertices v such that $(v, u) \in G$
  - out-neighbourhood $\rightarrow$ given a vertex u in a directed graph, set of vertices v such that $(u, v) \in G$
- degree, in-degree, out-degree
  - degree $\rightarrow$ size of neighbourhood
- traversal
  - visit each vertex of the graph to perform some task

Definitions

path:= a set of edges to get from one vertex to another: $(v, u_1), (u_1, u_2), \ldots, (u_{k-1}, w)$
length of path:= number of edges
simple path:= no repeated edge or vertex
cycle:= a path with the end vertex equal to start vertex
simple cycle:= a cycle with no repeated edge or vertex
tree:= undirected graph that is acylic and connected
forest:= acyclic graph (collection of disjoint trees, each one a "connected component")

Adjacency Matrix

Let $V = v_1, v_2, \ldots, v_n$. Store edges in $[n \cdot n]$ array:

	A	B	C	D	E	F	G
A	0	1	1	1	0	0	0
B	1	0	1	1	0	0	0
C	1	1	0	1	0	1	0
D	1	1	1	0	1	0	1
E	0	0	0	0	0	0	0
F	0	0	1	1	0	0	0
G	0	0	0	1	0	0	0

Undirected graphs: matrix is symmetric $(A[i,j] = A[j.i])$
Space $\in Theta(n^2)$, edge queries take time $\Theta(1)$.
Weighted graph: store $w(v_i, v_j)$ in $A[i,j]$ if (v_i, v_j) in E; special value (-1/0/∞) otherwise (depending on application)

Adjacency List

Let V $= v_1, v_2, \ldots, v_n$. Store edges in a list of lists: "main" list has positions $1, \ldots, n$ (one for each vertex); sub-list$[i]$ contains $j_1, \ldots, j_{k_i}$ such that $(v_i, v_{j_1}), \ldots, (v_i, v_{i_{k_i}})$ are all edges from v_i.
Undirected graph: edge $\{u, v\}$ stored twice (u in sub-list$[v]$ and v in sub-list$[u]$).

A	b,c
B	
C	a,b,c
D	e
E	

Space $\in \Theta(n + m)$ (where $n = |V|$ and $m = |E|$).
Edge queries in time $\Theta(\log n)$ (actually, $\Theta(\log(\text{max degree}))$) if sub-lists stored in balanced trees.