Networks And Graphs: Final Exam Preparation

5.1: Trees

An important example where trees are used is transportation networks.
Connector problem: setting up a communication infrastructure between a collection of nodes but such that the total costs are minimized.
Communication network: Set up an overlay network so that the total cost from a source to all destinations is minimized.
Data structures: The parse tree of sentences (NLP), B+Trees, etc.
The tree contains two types of nodes: leaf nodes which are the ones having a degree 1 forming the "lowest level" nodes, contain the variables or constants. The other intermediate nodes represent operations.

5.2: Fundamentals

Theorem 5.1: *For any connected (simple) graph G with n vertices and m edges, $n \leq m + 1$.*
Since this theorem works for simple graphs it certainly also works for non-simple graphs hence…
Theorem 5.2: *For any tree T with n vertices and m edges, $n = m + 1$, is a tree.*
Theorem 5.3: *A connected graph G with n vertices and m edges for which $n = m + 1$, is a tree.*
Theorem 5.4: *A graph G is a tree if and only if there exists exactly one path between every two vertices u and v.*
Theorem 5.5: *An edge e of a graph G is a cut edge if and only if e is not part of any cycle of G. Therefore, a connected graph G is a tree if and only if every edge is a cut edge.*

For a graph with n vertices and m edges:
1. G is a tree, that is, it is connected and acyclic.
2. G is connected with $n = m + 1$.
3. G is acyclic with $n = m + 1$.
4. There exists a unique path between every two vertices.
5. G is connected and every edge is a cut edge.
6. G does not contain a cycle and adding a single edge creates a unique one in G.

5.3 Spanning Trees

A spanning tree of a connected graph G is an acyclic connected subgraph of G containing all of G's vertices.
Let *G* be a weighted graph. A spanning tree *T* is **minimal** (an **MST**) if the sum of the weights of its edges is minimal.
Application: designs of networks for e.g computers, telecommunication, transportation, water supply and electricity.
Let T be a spanning tree for a graph G, and $e \in E(G) \setminus E(T)$ .Then H = T U {e} contains a cycle.
Lemma 1, Joining Fragments: Let G be a weighted connected graph. Let V_1 and V_2 be two nonempty partitions of the vertices of G. Let *e* be an edge in G with minimum weight $w(e)$ among those linking V_1 and V_2. Then there is a minimum spanning tree T of G that contains edge e.

Kruskal's Algorithm: *Choose an edge$_1$ with minimal weight. Choose a next edge$_k$+1 such that the following two conditions are met*

The induced graph is acyclic

The weight (e$_k$+1) is minimal

Stop when there is no more edge to select in the previous step.

Theorem 5.7: *Consider a weighted graph G with n vertices. Any spanning tree T$_{Kruskal}$ of G constructed by Kruskal's algorithm has minimal weight.*

For an efficient implementation it is important to use the disjoint-set data structure, to
1. determine if an edge joins different subtrees, and
2. join two different subtrees

See the course Data structures & Algorithms.

Lemma 2 (Weighted Graph with Distinct Weights): Let G be a weighted graph in which all edge weights are distinct. Let T be a minimal spanning tree for G, and S a subtree of T.

Let e be the lowest-weight outgoing edge of S (i.e., e is incident to exactly one vertex in S).

Then $e \in E(T)$

Consequences: A different MST algorithm (Prim-Jarnik): Repeatedly extend a subgraph, starting from an arbitrary vertex, by adding the least-weight outgoing edge (which is unique!). The result is a minimal spanning tree by the lemma.

In a weighted graph G in which all edges are distinct, the minimal spanning tree is unique.

5.4 Routing In Communication Networks

Trees are very important for communication networks.

A node usually maintains a routing table where each row specifies to which interface a message should be forwarded.

Messages to destination u should follow a path along a spanning tree rooted at u.

With *u* being the destination node, such a rooted tree is also called a sink tree for *u*.

Dijkstra's Algorithm: *It is used to find a sink tree, one for every node in the network. The algorithm works for both directed and undirected graphs. One restriction of the algorithm is that the weight of the arc cannot be negative. It is widely deployed in communication networks where it is known as a* ***link-state routing protocol.*** *The algorithm uses edge relaxation.*

The root node is the destination.

WRITE THE ALGORITHM LATER

Implementing the algorithm efficiently requires the Fibonacci heap data structure, to determine which vertex in *H* has the smallest *d*-value. (This will be explained in the course *Data structures & Algorithms.*) Its worst-case time complexity then is $O(m + n \log n)$, where n and m are number of vertices and arcs respectively. Dijkstra's original description had worst-case time complexity $O(n^2)$, because he didn't use a heap...

We need to know which vertices are adjacent to each other and what the weight of their respective connecting edges are.

Bellman-Ford Algorithm: *We don't have to know the topology of the graph when we're applying this algorithm. It computes shortest paths from all vertices to a vertex u, in a weighted digraph. It allows edges to be negative. Again it uses relaxation, but now all arcs are considered n times (with n the number of vertices in the digraph). The idea is that each shortest path contains at most $n - 1$ arcs. Except when there is a cycle of negative weight!*

The time complexity of the algorithm is O (mn). At each step, each vertex needs to inspect the information collected at each of its neighbors. In total, the vertices needs to inspect roughly m other vertices, where m is the total number of edges. The total number of steps we need to perform is equal to the length of the longest shortest path and can be shown to increase proportional to the number of vertices.

On the Internet, data is routed with several routing protocols.

- Link-state routing uses Dijkstra. It consists of periodically send packets to all other nodes with the link-state between the nodes;
- Distance-based routing uses Bellman-Ford. In this protocol nodes only inform the neighbors of their state, and do not flood the network.

6.1: Vertex Degrees

If most vertex degrees are the same, we are dealing with a more or less regular network in which vertices have equal roles. On the other hand high-degree vertices play the role of **hubs**, of which the removal may actually partition a connected network into several components.

6.2: Distance Statistics

In real-world situations, when networks become very large, it is difficult to (visually) discover properties. Network analysis tools are needed.
Distribution of vertex degrees: How many vertices have a high respectively low degree?
Distance statistics: Where are vertices positioned in the network? Far away from each other, central in the network, etc.
Clustering: To which extent are my neighbors adjacent to each other?
Centrality: Are some vertices more important than others?

Let G be connected. $d\,(u,\,v)$ denotes the distance between vertices u and v, i.e., the length of a shortest path between u and v.
Eccentricity $\varepsilon(u)$: max $\{d\,(u,v)\ v \in V(G)\}$ maximum distance between vertex u and any other vertex
Radius $rad(G)$: min $\{\varepsilon\,(u)\ u \in V(G)\}$ min eccentricity. how disseparate the vertices in a network are.
Diameter $diam(G)$: max $\{d\,(u,\,v)\ u,\,v \in V(G)\}$ maximum distance between two vertices
$\bar{d}(u)$ is the *average* length of shortest paths from u to any other v
$\bar{d}\ G$ denotes the average path length
The characteristic path length is the median over all $\bar{d}\,(u)$. If number of values is even: (k /2 + k+1/2) / 2 is the median

6.3: Clustering Coefficient

In real-life networks, vertices often show a high degree of clustering: many neighbors of a vertex are each other's neighbors. In social networks the higher the degree of clustering, the slower the dissemination.

Maximum number of edges between neighbors of *v* is $\binom{\delta(v)}{2}$ = 1/2 $\delta(v)$ $(\delta(v)$ -1).

In a simple connected and undirected graph G with the set of vertices $V(G)$ where $v \in V(G)$, $\delta(v) > 1$ and has the neighbor set $N\,(v)$ the clustering coefficient of the node is calculated

as follows: (2 x m_v) / n_v (n_v −1) where m_v is the number of edges in a subgraph induced by $N(G)$ and n_v =|$N(G)$|. If it is a directed graph, the formula becomes m_v / n_v (n_v −1).
The clustering coefficient of CC (G) is (cc (u) + cc (v)…) / |V (G)*| *only the ones with a higher degree than 1.

A triangle in a graph is a complete subgraph of 3 vertices. A triple in a graph is a subgraph of 3 vertices and 2 edges.
A simple, connected graph G has $n_\Delta G$ distinct triangles and $n_\wedge(G)$ distinct triples. The network transitivity $\tau(G) = n_\Delta G / n_\wedge(G)$.

n_Δ(v) : Number of triangles in which v is a member

$n_\wedge$(v): Number of triples at v.

A triple at v means that v is incident to both edges ("in the middle").

Observation:

$$n_\Delta(G) = \frac{1}{3} \cdot \sum_{v \in V^*} n_\Delta(v)$$
$$n_\wedge(v) = \binom{\delta(v)}{2} = \frac{1}{2} \cdot \delta(v) \cdot (\delta(v) - 1)$$
$$cc(v) = \frac{n_\Delta(v)}{n_\wedge(v)}$$

6.4: Centrality

Consider a (strongly) connected graph G. The center $C(G)$ is the set of vertices with minimal eccentricity. $C(G) \overset{\text{def}}{} \{v \in V(G) \mid \varepsilon(v) = rad(G)\}$.
At the center means at minimal distance to the farthest vertex. The higher the centrality, the "closer" to the center of a graph.
$c_E(u)$ denotes the (eccentricity based) vertex centrality of u: 1/ ε(u)

In some cases, it is more important to know how close a node is to *all* other nodes. $c_C(u)$ denotes the closeness of u: 1/d (u, v)

Important vertices are those who lie on many shortest paths, as their removal may significantly increase the distance between other vertices.
$S(x, y)$ is set of shortest paths between x and y.
$S(x, u, y) \subseteq S(x, y)$ contains the shortest paths that pass through u.
$c_B(u)$ denotes the betweenness centrality of u:

$$c_B(u) = \sum_{x \neq y} \frac{|S(x,u,y)|}{|S(x,y)|}$$

7.1, 7.2: Introduction and Classical Random Networks

Many real world networks can be modeled as a random graph in which each edge appears with some probability.

An Erdős–Rényi $ER(n, p)$ graph is a simple, undirected graph which consists of n vertices and each pair of distinct vertices is adjacent with probability $p \in [0,1]$. Each vertex has $n - 1$ possible incident edges. So in an ER-graph, on average we can expect $p \cdot (n - 1)$ edges at each vertex.
Theorem: The expected clustering coefficient for any vertex in an ER-graph is p.
Theorem: $P[\delta(u) = k]$ is the probability that $\delta(u) = k$ (in an ER-graph).

$$\mathbb{P}[\delta(u) = k] = \binom{n-1}{k} \cdot p^k \cdot (1-p)^{n-1-k}$$

Theorem: The probability of a specific subset of k neighbors is $p^k.(1-p)^{n-1-k}$

$ER(n, p)$ represents a group of Erdős–Rényi graphs. Most $ER(n, p)$ graphs aren't isomorphic.
$ER(2000, 0.015)$

- expected $\bar{\delta} = 0.015 \times 1999 = 29.985$!!
- expected E $G2$ ="-n-δ="$\times 2000 \times 29.985 = 2998$.
- in $G2$: 29708 edges

The probability that the degree distribution of an ER-graph resembles the expected one increases with the size of the graph.
For large $G \in ER\ n, p$ the expected average shortest path length $\bar{d}(G)$ tends to $\frac{\ln(n) - \gamma}{\ln(\delta)} + 0.5$
Where ln is the natural logarithm and γ the Euler's constant (0.5772)
In an ER graph most vertices gather in a component and a few are not connected to the component. This component is called the giant component and as we increase p the component gets larger very quickly.

ER graphs experience a phase transition around ph = 1/n.
If $p < 1/n$, then the connected components are typically of size $O(\log n)$.
If ph $> 1/n$, then there is typically a connected component of size $O(n)$.

<u>7.3: Small Worlds</u>

Many real-world networks have a small average shortest path length, as well as a high clustering coefficient whereas ER graphs have a small average shortest path length but a low clustering coefficient.
Many real-world networks (collaboration networks, World Wide Web, Internet) contain very few high-degree vertices. $P[\delta(u) = k]$ decreases exponentially when k increases.

Watts-Strogatz graphs: The idea is to combine properties of classical random graphs with high clustering coefficients.
V= {v1, v2, v3,…, vn}. Choose $n \gg k \gg \ln(n) \gg 1$, with k even. ($\gg$ means much larger)

- Order the n vertices into a ring. Connect each vertex to its first k/2 right-hand neighbors, and to its first k/2 left-hand neighbors. This is equivalent to construct a Harary graph .
- For each vertex u, considers (only once) each edge $\langle u,v \rangle \in H_{k,n}$. With probability p, replace it by $\langle u,w \rangle$ where $w \neq v$ is randomly chosen from $V - N(u)$. The resulting graph is in $WS(n, k, p)$.

Observation: Many vertices in a WS-graph will be close to each other. Because if p isn't very close to 0, edges are created to other "groups" of vertices.
Weak links are the long links in a WS-graph that cross the ring.

Observation: WS-graphs have a high clustering coefficient because for p significantly smaller than 1, many edges aren't replaced.
Theorem: CC (G) ≈ 0.75 for any WS graph. For any WS(n, k, 0) graph CC(G) = 3 (k-2) / 4 (k-1).

Theorem: For all graphs in $WS(n, k, 0)$, the average shortest path length $\bar{d}_u$ from vertex u to any other vertex is roughly n/2k.
Theorem: The average shortest path length in $WS(n, k, 0)$ graphs is high whereas in small world graphs this is not the case. But if p increases, the average shortest path length drops rapidly but the clustering coefficient stays relatively high. Typically, $p = 0.05$ is a good value for both $CC(G)$ and $\bar{d}(G)$.

7.4: Scale-free Networks

The WS graph is used to represent the small-world phenomenon but it doesn't capture real-world networks such as biological networks. In a real-world network there are a few high degree nodes and number of nodes with a high degree decreases exponentially.

In a scale-free network, the degree distribution follows a power law: $P[\delta(u) = k] \propto k^{-\alpha}$ (usually $2 < \alpha < 3$)
If the degree distribution is scale-free then the shape is the same everywhere.
This becomes visible when we zoom in at vertices with a low degree.

We can only build *scale-free* networks by using a growth process combined with preferential attachment. Meaning that in order to understand real-world large networks, we mimic their creation by observing how new nodes attach themselves to existing nodes.

Barbasi-Albert graphs: Let $G \in ER(n_0, p)$ and $V = V(G)$. Let $n \gg n_0$. While $|V| < n$ do: :

1. Add a new vertex v_s to V_{s-1} (i.e., $V_s \leftarrow V_{s-1} \cup \{v_s\}$).
2. Add a new edge $<v,u>$ for $m \leq n_0$ distinct $u \in V - \{v\}$. Each u chosen with a probability

$$\mathbb{P}[select\ u] = \frac{\delta(u)}{\sum_{w \in V - \{v\}} \delta(w)}$$

3. Stop when n vertices are added.

Result: a Barbarasi-Albert graph BA (n, n0; m). The expected distribution is: $\mathbb{P}[\delta(u) = k] \approx \frac{2m^2}{k^3} \propto \frac{1}{k^3}$

Generalized Barbasi-Albert Graphs: Start with a set V of $n0$ vertices, and no edges. While $|V| < n$ do:

1. $V \leftarrow V \cup \{v\}$ for some new vertex v. / Add a new vertex v_s to V_{s-1}

2. Add edges $<v,u>$ for $m \leq n_0$ distinct $u \in V - \{v\}$. Each u is chosen with prob. proportional to $\delta(u)$.

3. For a constant $c \geq 0$, add $c \times m$ edges between vertices in $V - \{v\}$. The probability to add an edge $<x,y>$ is proportional to $\delta(x) \times \delta(y)$.

In a generalized $BA(n, n0, m)$-graph, the expected degree distribution is $\mathbb{P}[\delta(u) = k] \propto k^{-(2+\frac{1}{1+2c})}$

For c = 0, this is a (standard) BA graph, but for increasing values of c, the exponent =2.

Observation: For c = 0 $\mathbb{P}[\delta(u) = k] \propto \frac{1}{k^3}$ and $\lim_{c\to\infty} \mathbb{P}[\delta(u) = k] \propto \frac{1}{k^2}$

Graphs in $BA(n, n_0, m)$ have expected average shortest path length: (with γ the Euler constant (0.5772))

$$\frac{\ln(n) - \ln\left(\frac{m}{2}\right) - 1 - \gamma}{\ln(\ln(n)) + \ln\left(\frac{m}{2}\right)} + 1.5$$

BA-graphs have shorter average shortest path lengths than ER graphs, due to the presence of "hubs", i.e., vertices with a very high degree.

Hubs in scale-free networks make them vulnerable to targeted attacks. A scale-free network quickly becomes disconnected if hubs are removed. But it is at least as robust as an ER- graph under a random attack.

The expected clustering coefficient of a vertex v_s that was added to a $BA(n, n_0, m)$ graph in step s:

$$cc(v_s) = \frac{m-1}{8(\sqrt{n} + \sqrt{s}/m)^2}\left(ln^2(n) + \frac{4m}{(m-1)^2} ln^2(s)\right)$$

Among the vertices present at the start of constructing a BA-graph, only few are lucky to get connections early on, and are likely to get a very high degree. Vertices that don't get connections early on most likely end up with a small clustering coefficient. By contrast, vertices v with say $s \geq$ 20000 are mostly linked to vertices with a very high degree.

BA-graphs have higher clustering coefficients than ER-graphs, but these values are still relatively small compared to the real-world networks.

9.1: Social Network Analysis, Introduction

Google uses hyperlinks to determine the importance of a page.
Google page rank algorithm: p = page out of n pages d ∊ [0,1) is a constant (probably 0.85 in the case of Google)
Let V = { v1, v2, v3, …, vn}, t = 0, d ≈ 0.85

1. Set rank (v_i,t) = 1/n for all $v_i \in V$
2. For all $v_i \in V$, calculate

$$rank\ (v_i, t+1) = \frac{1-d}{n} + \text{d} \sum_{\langle q,p\rangle \in E} \frac{rank\ (q,t)}{\delta_{\ out}\ (q)}$$

3. Increment t for one unit
4. Go to step 2 unless we reach a maximum number of iterations or Σ $rank\ (v,t) - rank\ (v,t-1)$ is small enough.

PageRank is clearly based on indegrees, yet the rank of a page and its indegree turn out to be only weakly correlated. When we rank pages according to PageRank: P [rank = k] is proportional to 1 / $k^{2.1}$

Graphs can analyse social structures such as formation of groups, influence of relationships, ties of families and friends, (dis)liking in groups of people.

Jacob Moreno, 1934: A sociogram is a graph-like representation of a social structure.
Ex: To get an impression of how a class operates, teachers ask their pupils to list the three classmates they like or dislike the most.

Classroom: min ε to see which child is important then closeness but closeness is not always be a good indicator of importance. Thus, we look at the betweenness centrality which is considered the best indicator of importance. A popular child may not be the best for spreading info.

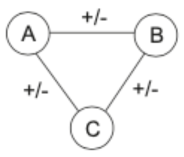
9.2: Some Basic Concepts

Proximity prestige: Let D be a digraph (with n vertices).The **influence domain** $R^{-}(v)$ of vertex v is the set of vertices from which v can be reached: $\{u \in V(D) - v \mid \textit{there exists a } (u,v) - path\}$
The proximity prestige $p_{prox}(v)$ is :

$$\frac{|R^{-}(v)|^2}{(n-1)\cdot\sum_{u\in R^{-}(v)} d(u,v)}$$

Intuition: the fraction of vertices that can reach v, divided by the average distance of these vertices to v.

Consider a digraph with adjacency matrix A where, $0 \leq A[v,u] \leq 1$ and $\Sigma\ A[v,u] = 1$ *for each vertex u.*
Intuition: $A\ [v,\ u]$ expresses how much v is appreciated by u.
The ranked prestige of vertices v satisfies: $p_{rank}(v) = \Sigma\ A[v,u] \times p_{rank}(u)$
$\Sigma\ p_{rank}(v)^2 = 1$
Intuition: The importance of $A\ [v,\ u]$ depends on $p_{rank}(u)$
A **triad** is a (potential) relationship between a triple of social entities; the relationship between each pair is labeled as positive or negative. We consider **balanced** triads.



In a signed graph, each edge ***e*** is labeled with either a positive ("+") or a negative ("−") sign, denoted by ***sign(e)***. The graph can be directed or undirected.
The ***sign(T)*** of a walk T is the product of the signs of its edges: $\Pi_{e \in E(T)}\ sign\ (e)$

Theorem: A sign graph is balanced if all its cycles are positive. A signed graph G is balanced if and only if V(G) can be partitioned into two disjoint subsets v0 and v1 such that:
each negative edge is incident to a vertex from v0 and from v1: $E^{-}(G) = \{ \langle x,y\rangle \mid x \in V_0,\ y \in V_1\}$
each positive edge is incident to vertices from the same set. $E^{+}(G) = \{ \langle x,y\rangle \mid x,y \in V_0 \text{ or } x,y \in V_1\}$

In an **affiliation network**, people are tied together through **membership relations** (e.g., a sports club or management team).

Social structures are assumed to consist of **actors** and **events**. Actors are tied to each other through joint participation in an event.
Discover correlations between events by actors that participate in both.

Affiliation networks are naturally represented as bipartite graphs. Vertices in V_A represent actors and in V_E events.
Edge <a, e> represents actor a participates in event e.
Number off events in which actors a and b both participate: NE [a, b] = AE[a, e] x AE [b, e]
Number of actors that participate in events e and f: NA [e, f] = AE[a, e] x AE[a, f]

Jan Mertens (union leader, 1968): “Holland is governed by 200 people.”
Since 2006, De Volkskrant determines the 200 most influential people by means of an affiliation network.

Observation: Barabasi networks rely on two assumptions:
(A1) the number of edges grows linearly in the number of nodes
(A2) Diameter grows slowly (small world network)

Empirical evaluation has been done on **static** networks. Do realistic graphs indeed grow with these assumptions?

Densification

If we look at real-world graphs (esp. social networks) for a long time, then we observe that:

1. The average node degree increases. This phenomenon is called densification. More specifically: e(t) is proportional to n(t) $^{\alpha}$ where $e\ t$ and $n(t)$ are the number of edges and nodes resp. at time t and $\alpha > 1$.
2. The **diameter decreases** at the network grows.

What causes the densification and related diameter shrinking? Leskovec et al. (2005) propose two models: Community Guided Attachment (CGA) and Forest Fire Model (not covered in this course)

Community Guided Attachment (CGA): It has been observed that power-law occurs in *self- similar* datasets.
An object is *self-similar* it is similar to a part of itself.One example of a self-similar graph is communities.
Definition: Let T be a tree with N leaves, height H, and constant fanout b. Also, let $h_t(v, w)$ be the tree distance between leaves *v* and *w,* i.e., the height of the smallest subtree of T which contains both v and w .
Algorithm: We construct a *random graph* with N nodes where the probability that there is an edge between v and w is f(h" v,w).The function *f* is called *Difficulty Function.*

To obtain densification, *f* needs to be scale-free.
Theorem: If *f(h) =* c^{-h} *and* $1 \leq c < b$*,then the graph produced by the previous algorithm will have e(t) is proportional to n(t)* $^{\alpha}$ *where* $\alpha = \log_b c$*.*
Our algorithm does not let the graph “grow” as it is with Barabasi. However, a simple modification is possible to include the growth process in the construction of *G.*