Components in Graphs

25 September 2024 20:13

Top down approach → focus on constraints.

Look @ whole structure, identify substructs.

– I have a big picture, zoom in where I think there's a weak spot.

Bingo.

I can see decomposition into smaller units.

Directed graph – components (weak and strong)

Components of a graph are sub-graphs that are connected within, but disconnected between themselves.

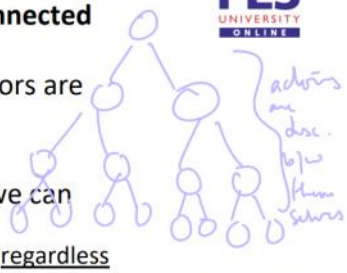
If a graph contains one or more "isolates," these actors are components (strictly by definition).

A disconnected graph may have components.

For directed graphs (in contrast to simple graphs), we can define two different kinds of components.

- A **weak component** is a set of nodes that are connected, regardless of the direction of ties. We can find such a component in undirected version of the directed graph but may not find in the original directed graph.
- A **strong component** requires that there be a directed path from A to B in order for the two to be in the same component

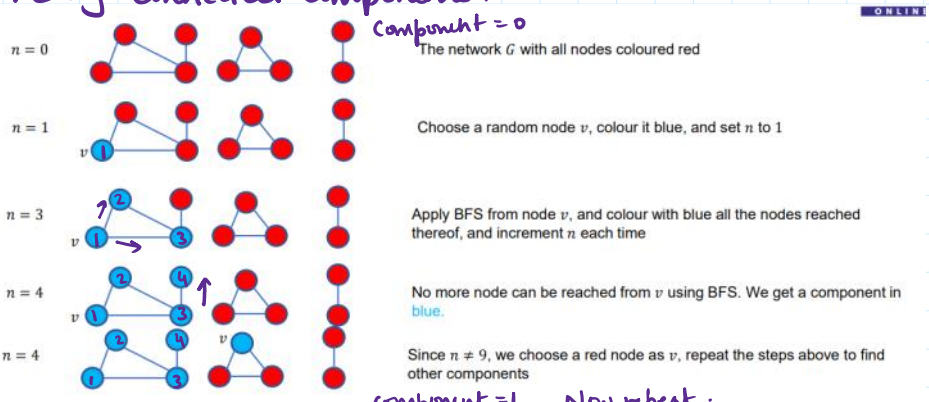
Always directed



weak component: connection exists in undirected graph, make it directed & there's no connection

Finding Connected Components:

component = 0



component = 1. Now repeat.

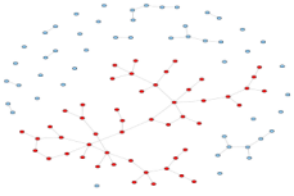
Directed graph

- out component (vertices reached from strong conn. component)
- in component (vertices that reach to the –11–)

↓

If node is both out & in component of scc, then it's part of the strongly connected comp.

In **real undirected networks**, we typically find that there is a large component (the **giant component**) that fills most of the network - **usually more than half** and **sometimes over 90%** - while the **rest of the network is divided into a large number of small components** disconnected from the rest.



One big group & then the others

Why? ① New node joins — connects itself to node w/ highest degree.

Namita Achyuthan

Why? ① New node joins — connects itself to node w/ highest degree.
(Preferential Attachment: Rich get richer)

Blocks + Cut Points.

"If node was removed, would struct. be divided into unconnected parts?"

→ Such nodes = CUT POINTS.
= act as brokers among otherwise disconnected groups.

* Component analysis = disconnected parts
* Bicomponent analysis = vulnerable parts

↓

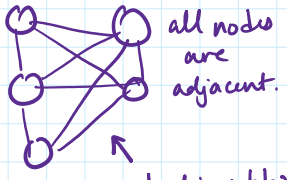
divisions into which they divide the parts = BLOCKS.

Bottom Up Approach.

find substructs → see how it builds up into bigger structures
dyad → triad → dense cluster.

Words you have to know:

① CLIQUE
- highly cohesive subgroup (induced subgraph that is a complete graph)
- may overlap + share nodes.
- every member is connected to everyone else



- MAXIMAL CLIQUE — can't add another node w/o disturbing property.
- MAXIMUM CLIQUE — clique w/ largest no. of vertices.

Problem is — super rigid definition — bleh.
Relax restrictions little bit.

② K-CLIQUE (focus on distance)
- clique where shortest path distance ≤ k.
  - 1-clique = everyone is one hop away
  - 2-clique = everyone @ max 2 hops away.

③ K-CLAN
- restrict k-clique even more — "all ties among actors occur thro' other members of group"

④ K-PLEX
- each vertex can have upto k-non neighbors
  "I can not be friends w/ this many people in my group"

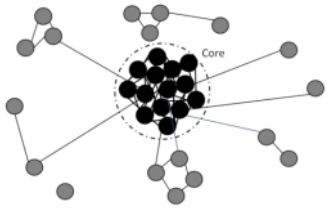
Degeneracy.

(≤k) k-degenerate = undirected graph
every subgraph has a vertex of degree at most k.

Namita Achyuthan

(≤k) k-degenerate = undirected graph
every subgraph has a vertex of degree at most k.
no matter how you look @ it, you can always find atleast one vertex that has k >, connections to other vertices.

degeneracy of a graph is the smallest value of k.



core periphery struct ↑

(≥k) k core = vertex has ≥k connections to others
= core no. is a measure of prestige.

A **k-core is a maximal subgraph of nodes**, such that each node in the subgraph has **at least a degree k** .

- 0-kore is full graph
- 1-kore is a graph with no isolate
- 2-kore (at least 2 neighbors); hence 2 core cannot have a pendulum structure as each leaf of the pendulum needs a degree 1 node

Any graph usually has **more than one core;** the core with the **largest possible k** is called the **main core of the graph.**

Two k-cores cannot overlap, since then they would just form another k-core of larger size

Finding k core: ——— remove k-core, left w/ k-const.
└ all nodes in core have k neighbors ⇒ k corona.

① Start w/ OG graph

② Remove all nodes w/ degree < k. — remaining nodes lose neighbors — degree of nodes ↓
Remove all incident edges.

③ Remove nodes w <k degree after prev. removal.

④ Remaining nodes form k-core.

Namita Achyuthan