# Cuts, Connectivity, and Flow

# Vertex Cut and Connectivity

- A *separating set* or *vertex cut* of a graph G is a set $S \subseteq V(G)$ such that G–S has more than one component
  - A graph G is *k-connected* if every vertex cut has at least *k* vertices.
  - The *connectivity* of G, written as $\kappa(G)$, is the minimum size of a vertex cut.
  - A graph is *separable* if its vertex connectivity is 1

In all further discussions regarding connectivity, we leave out $K_n$

# Edge-connectivity

- A *disconnecting set* of edges is a set $F \subseteq E(G)$ such that G – F has more than one component.
  - A graph is *k-edge-connected* if every disconnecting set has at least *k* edges.
  - The *edge connectivity* of G, written as $\kappa'(G)$, is the minimum size of a disconnecting set.

# Edge Cut

- Given $S,T \subseteq V(G)$, we write [S,T] for the set of edges having one endpoint in S and the other in T.
  - An *edge cut* is an edge set of the form [S,S′], where S is a nonempty proper subset of V(G).
- Other definition of edge-cut:
  - A set of edges that disconnects a graph
  - *Minimal edge cut* – an edge cut that does not contain any other edge cut as its proper subset
- Do the two definitions give the same set of edge cuts for a given graph?

# Results

- Theorem [Whitney, 1932]

  If G is a simple graph, then $\kappa(G) \leq \kappa'(G) \leq \delta(G)$

- If S is a subset of the vertices of a graph G, then:

  $|[S,S']| = [\Sigma_{v \in S} d(v)] - 2e(G[S])$

- If G is a simple graph and $|[S,S']| < \delta(G)$ for some nonempty proper subset S of V(G), then $|S| > \delta(G)$.

# More results…

- A graph G having at least three vertices is 2-connected (biconnected) if and only if each pair $u,v \in V(G)$ is connected by a pair of internally disjoint $u,v$-paths in G.

- If G is a k-connected graph, and G′ is obtained from G by adding a new vertex $y$ adjacent to at least $k$ vertices in G, then G′ is k-connected.

# And more …

- If $n(G) \geq 3$, then the following conditions are equivalent (and characterize 2-connected graphs)
  - G is connected and has no cut vertex.
  - For all $x,y \in V(G)$, there are internally disjoint $x,y$-paths
  - For all $x,y \in V(G)$, there is a cycle through $x$ and $y$.
  - $\delta(G) \geq 1$, and every pair of edges in G lies on a common cycle

# *x,y*-separator

- Given $x,y \in V(G)$, a set $S \subseteq V(G) - \{x,y\}$ is an *x,y-separator* or a *x,y-cut* if $G-S$ has no *x,y*-path.
  - Let $\kappa(x,y)$ be the minimum size of an *x,y*-cut.

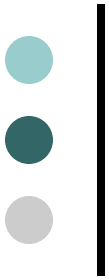
- Let $\lambda(x,y)$ be the maximum size of a set of pair-wise internally disjoint *x,y*-paths.
  - Let $\lambda(G)$ be the largest *k* such that $\lambda(x,y) \geq k$ for all $x,y \in V(G)$.
  - For $X,Y \subseteq V(G)$, an X,Y-*path* is a path having first vertex in *X*, last vertex in *Y*, and no other vertex in $X \cup Y$.

# Menger's Theorem

- If *x, y* are vertices of a graph G and $x,y \notin E(G)$, then the minimum size of an *x,y*-cut equals the maximum number of pair-wise internally disjoint *x,y*-paths.

- [Edge-version] If *x, y* are vertices of a graph G and $x,y \notin E(G)$, then the minimum size of an *x,y-edge*-cut equals the maximum number of pair-wise internally edge-disjoint *x,y*-paths.

- [Corollary] The connectivity of G equals the maximum *k* such that $\lambda(x,y) \geq k$ for all $x,y \in V(G)$. The edge connectivity of G equals the maximum *k* such that $\lambda'(x,y) \geq k$ for all $x,y \in V(G)$.



How do we compute the connectivity of graphs?

We will first look at ***Network flows***, and then come back to the above question

# Network Flow

Given

- A directed graph G = (V, E)
- Capacity C(e) ≥ 0 for all e in E
- Two distinguished vertices – s (*source*) and t (*sink*)

Flow f

- Assigns a value f(e) to each e in E

O(v) = total flow on edges leaving vertex v

I(v) = total flow on edges entering vertex v

Feasible flow – flow satisfying

- Capacity constraint : $0 \leq f(e) \leq C(e)$ for all e
- Conservation constraint: $I(v) = O(v)$ for all v except s and t
- Value of a flow $f = I(t) – O(t) = O(s) – I(s)$

- Maximum flow problem
  - Find flow with maximum value

# Augmenting Path

Residual graph $R_f$ corresponding to a feasible flow f

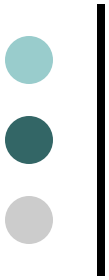
- For each pair <u, v> of vertices
  - if e1=(u, v) in E and e2=(v, u) in E
    - r(u, v) = C(e1) – f(e1) + f(e2)
    - r(v, u)=C(e2) – f(e2) + f(e1)
  - If e1 = (u,v) in E but (v,u) is not in E
    - r(u, v) = C(e1) – f(e1),  r(v, u) = f(e1)
  - If e2 = (v,u) in E but (u, v) is not in E
    - r(v, u) = C(e2) – f(e2),  r(u, v) = f(e2)
- Residual graph $R_f = (V, E_f)$, where
  - $E_f$ ={(u,v) | r(u,v) > 0} with weight w(e) = r(e) for e in $E_f$
- Augmenting path = path from s to t in $R_f$

# Ford-Fulkerson Algorithm

Input: a feasible flow f

*compute* $R_f$
*while (some augmenting path in* $R_f$*)*
*find one augmenting path P in* $R_f$
*find the edge with minimum weight W in P*
*augment (increase) the flow along all edges in P by W*
*compute new* $R_f$

When the loop terminates, the flow is a max-flow



- What is the worst case complexity of Ford-Fulkerson algorithm?
  - Note that F-F algorithm does not specify which augmenting path to choose
- There are many maximum flow algorithms with better complexity
  - primarily differs in the order in which augmenting paths are chosen, except for a class of algorithms called preflow-push algorithms
  - Example: Edmond-Karp, Dinic,…many more

# Max-flow Min-cut Theorem

- The maximum value of a feasible flow in a network equals the minimum capacity of a source/sink cut [Ford and Fulkerson, 1956]

# Finding Edge-connectivity

Input: Connected graph G = (V, E)

*choose any vertex p in V*
*min_size = |E|*
*for all vertices q ≠ p do*
    *find maxflow M in directed graph G' = (V, E')*
        *where E' = { (u,v), (v,u) | (u,v) in E }*
        *s = p, t = q, and all capacities = 1*
    *min_size = min (min_size, M)*
*edge connectivity of G = min_size*

Why is it sufficient to just find edge-connnectivity between a fixed *p* and all other vertices (and not between all pairs of vertices)?

Complexity?

# Finding Edge-cut

- The algorithm earlier gives the connectivity. What about finding an actual min-edge-cut?
- Easy
  - For each s-t pair in the flow problems solved,
    - Find set S of all vertices reachable from s in the final residual graph
    - Find [S, S']
  - Take the minimum s-t edge-cut found among all s-t pairs

# Finding Vertex-connectivity

Idea: convert a vertex to an edge to stop its reuse

- To find size of min-cut between vertices u and v
  - Convert the undirected graph G to a directed graph G' as before
  - Replace each vertex $x \neq u, v$, by two vertices x1, x2
  - Add a directed edge (x1,x2) with capacity 1
  - Replace each directed edge (w,x) in G' by a directed edge (w,x1), with capacity $\infty$
  - Replace each directed edge (x,w) in G' by a directed edge (x2,w), with capacity $\infty$
  - Solve the maximum flow problem with s = u and t = v
- To find vertex connectivity, repeat for all u,v pairs (why not just for a fixed u as before?), and take minimum