Lecture: More on Connectivity

Shuai Li

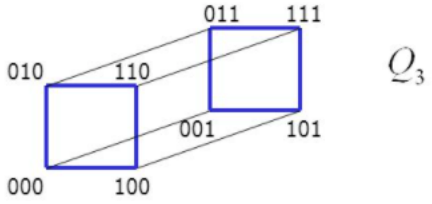
John Hopcroft Center, Shanghai Jiao Tong University

https://shuaili8.github.io

https://shuaili8.github.io/Teaching/CS445/index.html

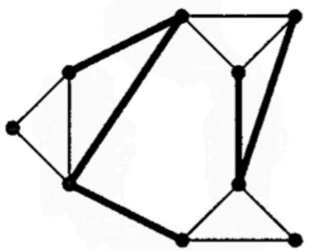
Vertex cut and connectivity

- A proper subset S of vertices is a vertex cut set if the graph $G - S$ is disconnected
- The connectivity, $\kappa(G)$, is the minimum size of a vertex set S of G such that $G - S$ is disconnected or has only one vertex
  - The graph is k-connected if $k \leq \kappa(G)$
- $\kappa(K^n) := n - 1$
- If G is disconnected, $\kappa(G) = 0$
  - $\Rightarrow$ A graph is connected $\Leftrightarrow \kappa(G) \geq 1$
- If G is connected, non-complete graph of order n, then
$$1 \leq \kappa(G) \leq n - 2$$

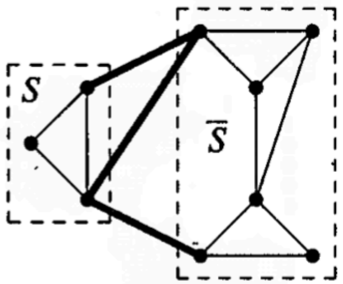


- For convention, $\kappa(K_1) = 0$
- Example (4.1.3, W) For k-dimensional cube $Q_k = \{0,1\}^k$, $\kappa(Q_k) = k$

Edge-connectivity



disconnecting set



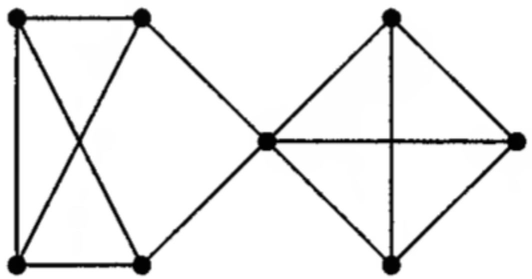
edge cut

- 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 $\lambda(G)$, is the minimum size of a disconnecting set
- 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 $[S, S^c]$ where S is a nonempty proper subset of $V(G)$
- Every edge cut is a disconnecting set, but not vice versa
- Every minimal disconnecting set of edges is an edge cut

Connectivity and edge-connectivity

Proposition (1.4.2, D) If G is non-trivial, then $\kappa(G) \leq \lambda(G) \leq \delta(G)$

- Example (4.1.10, W) Possibility of $\kappa(G) < \lambda(G) < \delta(G)$

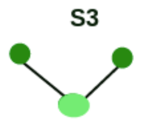


- Theorem (4.1.11, W) If G is a 3-regular graph, then $\kappa(G) = \lambda(G)$

Properties of edge cut

- When $\lambda(G) < \delta(G)$, a minimum edge cut cannot isolate a vertex
- Similarly for edge cut
- Proposition (4.1.12, W) If S is a set of vertices in a graph G, then
$$|[S, S^c]| = \sum_{v \in S} d(v) - 2e(G[S])$$
- Corollary (4.1.13, W) If G is a simple graph and $|[S, S^c]| < \delta(G)$ for some nonempty proper subset S of $V(G)$, then $|S| > \delta(G)$

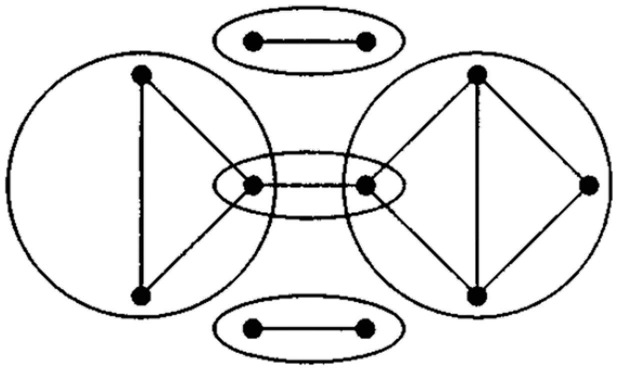
Bond



- An edge cut may contain another edge cut
- Example: $K_{1,2}$ or star graphs
- A bond is a minimal nonempty edge cut
- Proposition (4.1.15, W) If G is a connected graph, then an edge cut F is a bond $\Leftrightarrow$ $G - F$ has exactly two components

Blocks

- A block of a graph G is a maximal connected subgraph of G that has no cut-vertex. If G itself is connected and has no cut-vertex, then G is a block
- Example



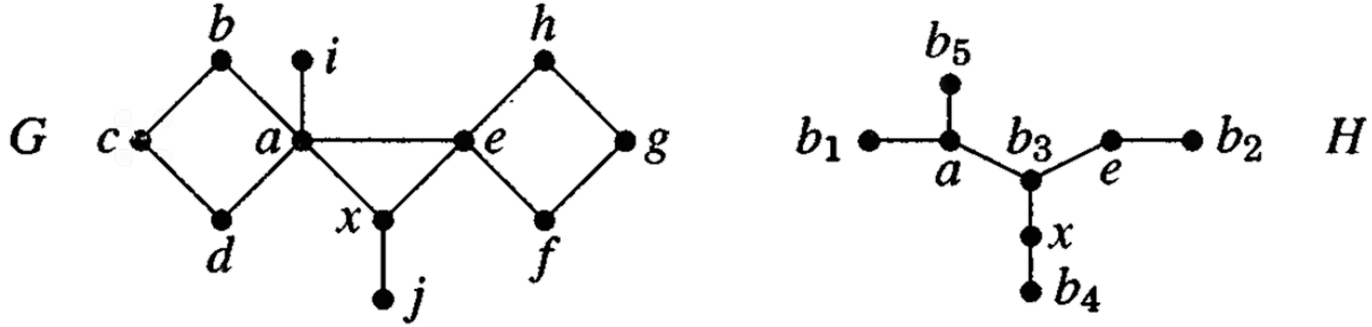
- An edge of a cycle cannot itself be a block
    - An edge is block $\Longleftrightarrow$ it is a bridge
    - The blocks of a tree are its edges
- If a block has more than two vertices, then it is 2-connected
    - The blocks of a loopless graph are its isolated vertices, bridges, and its maximal 2-connected subgraphs

Intersection of two blocks

- Proposition (4.1.19, W) Two blocks in a graph share at most one vertex
  - When two blocks share a vertex, it must be a cut-vertex
- Every edge is a subgraph with no cut-vertex and hence is in a block. Thus blocks in a graph decompose the edge set

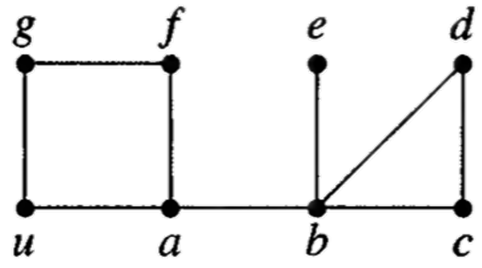
Block-cutpoint graph

- The block-cutpoint graph of a graph G is a bipartite graph H in which one partite set consists of the cut-vertices of G, and the other has a vertex b_i for each block B_i of G. We include vb_i as an edge of $H \Leftrightarrow v \in B_i$



- (Ex34, S4.1, W) When G is connected, its block-cutpoint graph is a tree

Depth-first search (DFS)



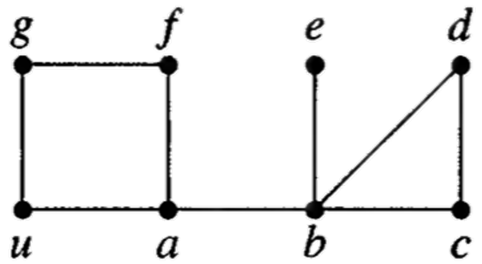
- Depth-first search

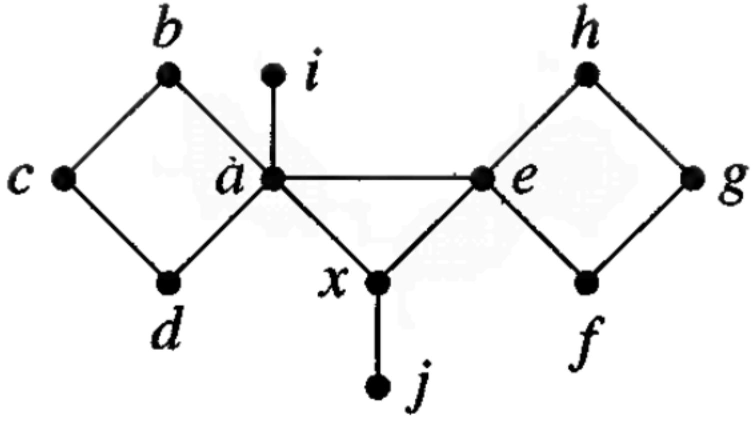
- Lemma (4.1.22, W) If T is a spanning tree of a connected graph grown by DFS from u, then every edge of G not in T consists of two vertices v, w such that v lies on the u, w-path in T

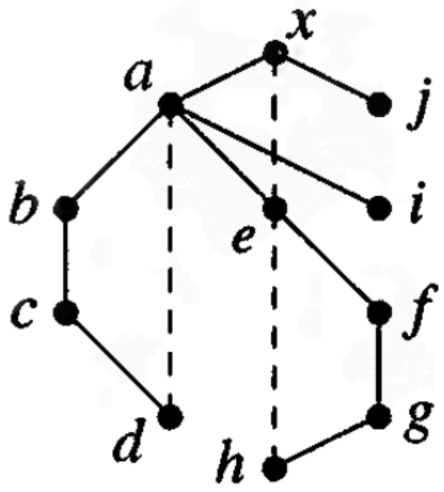
Finding blocks by DFS

- **Input**: A connected graph G
- **Idea**: Build a DFS tree T of G, discarding portions of T as blocks are identified. Maintain one vertex called ACTIVE
- **Initialization**: Pick a root $x \in V(H)$; make x ACTIVE; set $T = \{x\}$
- **Iteration**: Let v denote the current active vertex
  - If v has an unexplored incident edge vw, then
    - If $w \notin V(T)$, then add vw to T, mark vw explored, make w ACTIVE
    - If $w \in V(T)$, then w is an ancestor of v; mark vw explored
  - If v has no more unexplored incident edges, then
    - If $v \neq x$ and w is a parent of v, make w ACTIVE. If no vertex in the current subtree T' rooted at v has an explored edge to an ancestor above w, then $V(T') \cup \{w\}$ is the vertex set of a block; record this information and delete $V(T')$
    - if $v = x$, terminate

Example







Strong orientation

- Theorem (2.5, L) Let G be a finite connected graph without bridges. Then G admits a strong orientation, i.e. an orientation that is a strongly connected digraph
    - A directed graph is strongly connected if for every pair of vertices (v, w), there is a directed path from v to w

- The blocks of a loopless graph are its isolated vertices, bridges, and its maximal 2-connected subgraphs