Basic Graph Algorithms

- Exploring Graphs
- Depth First Search
- DFS Forest - Start and Finish Time
- DAG, Topological Sorting
- Strongly Connected Components
- Breadth First Search
- Bipartite Graphs

Imdad ullah Khan

Strongly Connected Components

A directed graph is strongly connected if for every pair of vertices u and v, there is a path from u to v and a path from v to u

A strongly connected component of a digraph G is a maximal strongly connected subgraph of G

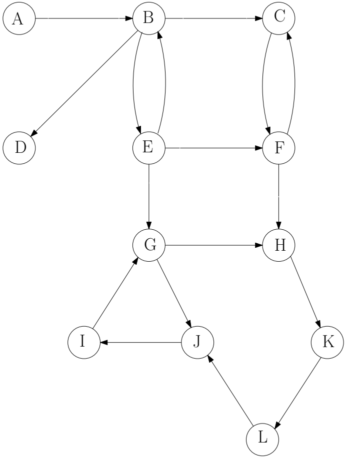
SCC apears in many applications areas in

- Social Network analysis
- Web Mining
- Markov Chain Theory
- Communication

SCC

Input: A digraph $G = (V, E)$

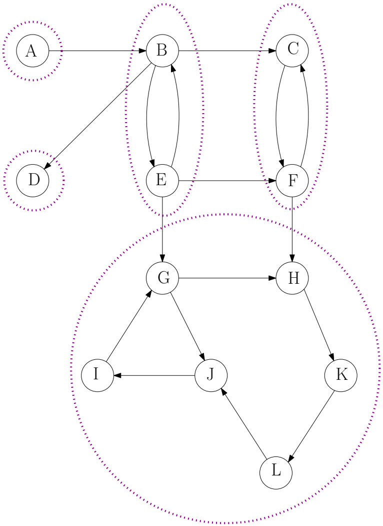
Ouput: SCC's of G



SCC

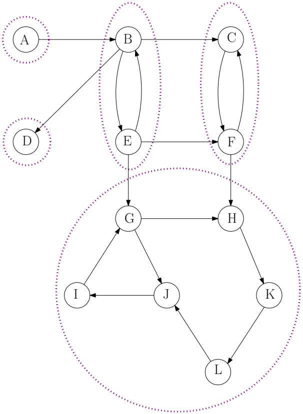
Input: A digraph $G = (V, E)$

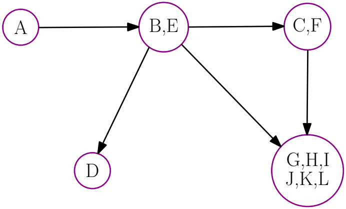
Ouput: SCC's of G



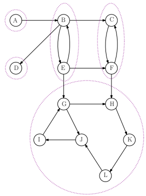
The SCC Graph

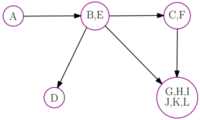
- Consider each SCC of G a (meta) vertex
- An edge from C_i to C_j if there is $(u, v) \in E$ with $u \in C_i$ and $v \in C_j$





The SCC Graph

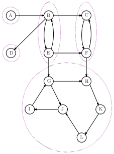


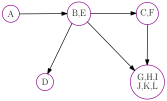


Lemma: The SCC graph of any digraph is a DAG

If there is a cycle, then meta vertices should be merged

Strongly Connected Components: Algorithm





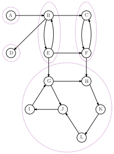
Lemma: Running DFS on G gives largest $f(v)$ to a vertex in a source component of the SCC graph

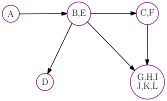
Lemma: Starting DFS on G from a vertex in a sink component, makes that sink component a separate tree in the DFS forest

Algorithm:

- Identify a vertex v in the sink component
- Run DFS-EXPLORE from v and remove the tree (when call finishes)
- Repeat

Strongly Connected Components: Algorithm





Lemma: Running DFS on G gives largest $f(v)$ to a vertex in a source component of the SCC graph

Lemma: Starting DFS on G from a vertex in a sink component, makes that sink component a separate tree in the DFS forest

Algorithm:

- Identify a vertex v in the sink component
- Run DFS-EXPLORE from v and remove the tree (when call finishes)
- Repeat

Source can be identified but not sink

Strongly Connected Components: Algorithm

Algorithm Find Strongly Connected Components in G

DFS(G) - RECORD fin-time array $f[1 \ldots n]$

Compute G^T ▷ Reverse all edges directions

DFS(G^T) -

in the main loop of DFS, process vertices in decreasing order of $f[\cdot]$

OUTPUT each DFS tree as an SCC

- Runtime is two runs of DFS plus graph transpose $O(n + m)$
- There are other ways to do it
- Complete your homeworks for detailed proofs