Lecture 7: Strongly Connected Components

A Shortest Paths in a Graph

Last Time:
- DFS
  - connected components in undirected graphs.
  - Pre & Post numbers.  Back / forward / cross edges.
  - Topological Sort.

Today:
- Strongly Connected Components.
- BFS = Breadth First Search

(if time permits)  • Dijkstra's Algorithm
What about directed graphs? (Digraphs)

- What's the right analog of CC in the directed case?
- How can we compute these components?
**Definition:** We say that \( u, v \) are **strongly connected** if there is a path from \( u \) to \( v \), and a path from \( v \) to \( u \).

**Claim:** This relation is an equivalence relation.

- **Reflexive:** \( \forall v: v \sim v \)
- **Symmetric:** \( u \sim v \Rightarrow v \sim u \)
- **Transitive:** \( u \sim v \land v \sim w \Rightarrow u \sim w \)
Given a graph $G$, shrink every SCC to a meta-node and connect meta-nodes $C \rightarrow C'$ iff there is an edge $(u,v) \in E(G)$ with $u \in C$, $v \in C'$. This produces the SCC meta-graph $G'$.

**Example:**

$G_{\text{in}}$ and $G_{\text{out}}$ show an example.

**Claim:** A digraph $G$, the SCC metagraph $G'$ is a DAG.
Recall: pre & post numbers in DFS.

Figure 3.7 DFS on a directed graph.

\[
\begin{bmatrix}
A & B & E & F & G & H & B & C & D & D & C & A
\end{bmatrix}
\]

1 2 3 4 5 6

\( u, v \) is a back edge

if

\[
\begin{bmatrix}
C & C & J & J \\
C & J & D & D
\end{bmatrix}
\]
**Q:** Can we find a node $v \in V$ s.t. $v$ is in a sink SCC and a source SCC?

**Claim:** Run DFS on $G$. The node with highest post number, then this node is in a source SCC.

**Proof:** Assume not. Let $v$ be the node with highest post number.

Let $C$ be the SCC of $v$.

There exists another component $C'$ such that $C' \rightarrow C$.

Consider the first vertex $u \in C \cup C'$ that is explored by DFS($G$).

- if $u \in C$: we process all vertices in $C$ only after we would start exploring $C'$, a contradiction.
- if $u \in C'$: $\text{post}(u) > \text{post}(v)$ = contradiction. $\square$
We actually proved something stronger:

Claim: Run DFS on $G$. A SCCs $c' \rightarrow c$.
the highest post number in $c' >$
the highest post number in $c$. 

**Strongly Connected Components - the Algorithm**

**Explore (G, v):**
- `visited[v] = true`
- `comp[v] = scc`
- for all `(u, v) \in E`:
  - if not `visited[u]`:
    - `Explore(G, u)`

**SCC(G):**
- Run DFS on `G^R` \( \Rightarrow \) post numbers.
- for all `v \in V`:
  - `visited[v] = false`
  - `comp[v] = null`
- `scc = 1` in descending order ordered by post on `G^R`
- for all `v \in V`:
  - if not `visited[v]`:
    - `Explore(G, v)`
    - `SCC += 1`
Q: What are the distances from A to all other vertices?

- $A \rightarrow A$: 0
- $A \rightarrow B$: 1
- $A \rightarrow D$: 1
- $A \rightarrow C$: 2
- $A \rightarrow E$: 2

Assume: unit distances on edges for now.

New Topic: Distances in Graphs, Shortest Paths
BFS\( (G,s) \):

- dist\( [s] = 0 \).
- \( \forall v \in V \) dist\( [v \not= s] = \infty \).
- CartLayer = \{ s \}
- queue queue
- while CartLayer \( \neq \{ \} \):
  - \( v = \) queue. eject()
  - NextLayer = CartLayer.

\textbf{Claim:} In iteration \( i \) of the while loop we identify all vertices of distance exactly \( i \) from \( s \).

\textbf{Proof:} By induction on \( i \).

\( A \stackrel{B}{\leftarrow} C \stackrel{E}{\rightarrow} \)

\( \{A\} \)

\( \{B,C,D\} \)

\( \{E\} \)
Running Time of BFS

Very similar to DFS. Also in BFS:
- every vertex is inserted to the queue at most once,
- ejected from the queue,
- every edge is considered at most once (di-graph)
  or at most twice (undirected graph).

\( \Theta(n+m) \).

\( n = |V| \quad m = |E| \).
Shortest Paths with Weights on Edges.

\( G = (V, E) \) \[ \text{n bits} \]

\[ \text{A} \quad 100 \quad \text{B} \]

\[ \text{A} \quad 50 \quad \text{C} \]

\[ \text{2}^n \text{ intermediate vertices} \]

\[ \text{199} \text{ intermediate} \]

\[ \text{what to do with negative weights?} \]

Next time:

- Dijkstra's Algorithm.
- Proof of Correctness.
- Runtime Analysis.