CSCB63 Winter 2021

Week 6 Lecture 2 - Directed Graphs and Strongly Connected Components

Anna Bretscher

February 19, 2021

TODAY

Quick Review DFS

Strongly Connected Components

Kosaraju's Algorithm

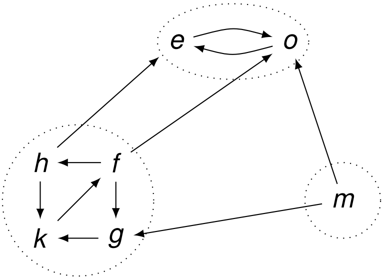
Recursive DFS

```
time = 0
start = any vertex
new stack order
dfs(start)

dfs(v)
   v.state = visited        # ie, label v green
   v.start = time
   time += 1
   for each neighbour w of v
      if w.state == visited        # ie, green
         # we have a cycle
      else if w.state == not_visited        # ie, black
         add edge vw to tree T
         dfs(w)
   v.finish = time
   v.state = finished
   order.push(v)          <== new
   time += 1
```

Strongly Connected Components

Strongly connected component (SCC): is *maximal subset* of *vertices* reachable from each other in a *directed* graph.



Three *strongly connected components*: $\{e, o\}$, $\{f, g, h, k\}$, $\{m\}$.

MOTIVATION

Q. What is an application that may want to find *strongly connected components*?

A.

-
-
-

Q. How do we find *strongly connected components* in a *directed graph*?

A. Two common algorithms, we will look at one - *Kosaraju's* algorithm.

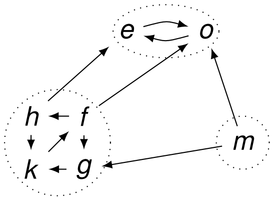
Makes an observation about G and it's *transpose*, G^T.

Transpose of G

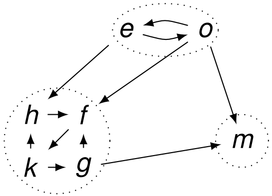
The *transpose of G* (notation G^{T}) means a graph with the same vertices as G and the edges are the *reverse* of G's.

Not to be confused with the *complement G^C of G*.

G:



G^{T}:



Observation: G^{T} has the *same* strongly connected components as G's.

Q. What is the *complexity* to compute adjacency lists of G^{T}:

A.

Kosaraju's SCC Algorithm

Overview.

- *DFS* on G, be sure to visit all vertices, note finish times, accumulate vertices in reverse finishing order (i.e., as we finish with them *push* them to a *stack*.)
- *Compute* adjacency lists of G^{T}.
- *DFS* on G^{T}, pop vertices from the *stack* to pick *start/restart vertices*.
- Each *tree* found has the vertices of *one strongly connected component*.

  Total $O(|V| + |E|)$ time.

  Let's see it in action...

Proof of Kosaraju's Algorithm

Notation

- We denote by $f(v)$ the time at which node v is finished (all outgoing neighbours have been visited).
- $f(u) < f(v)$ means "*node u was finished before node v.*"
- Note that every node is eventually *finished*, so this notation is *well-defined*.
- Let C be an *SCC*. Define $f(C)$ as the time at which *the last node* in C is finished.

$$f(C) = max_{v \in C} f(v)$$

Proof Overview

Lemma. If s is the first node in SCC C visited by DFS, then $f(C) = f(s)$.

Theorem Suppose we run DFS in G, restarting as needed. Let C_1 and C_2 be SCCs in G. If (u, v) is an edge in G where $u \in C_1$and $v \in C_2$, then $f(C_2) < f(C_1)$.

Corollary. Let C_1 and C_2 be distinct SCC's in $G = (V, E)$. Suppose there is an edge (u, v) in E^T where $u \in C_1$ and $v \in C_2$. Then $f(C_1) < f(C_2)$.

Corollary. Let C_1 and C_2 be distinct SCC's in $G = (V, E)$, and suppose that $f(C_1) > f(C_2)$. Then there cannot be an edge from C_1 to C_2 in G^T.

Corollary. Let C_1 and C_2 be distinct SCC's in $G = (V, E)$, and suppose that $f(C_1) > f(C_2)$. Then there cannot be an edge from C_1 to C_2 in G^T.

Q. What does this mean?

A.

-
-
-
-

Corollary. Let C_1 and C_2 be distinct SCC's in $G = (V, E)$, and suppose that $f(C_1) > f(C_2)$. Then there cannot be an edge from C_1 to C_2 in G^T.

Q. What does this mean?

A.

-
-
-
-
-

PROOFS

Lemma. If s is the first node in SCC C visited by DFS, then $f(C) = f(s)$.

Proof. At the time $DFS(s)$ is called,

- since s is the first node in C visited by the DFS, all nodes in C have *not been discovered* (colour black).
- Since C is a SCC, every node $v \in C$ is reachable from s.
- This means there is a path from s to v for every $v \in C$.
- Thus every node $v \in C$ will be *finished* when $DFS(s)$ returns.
- Since the last step of the $DFS(s)$ is to finish s, this means that s is finished only *after all other nodes* in C are finished.
- Therefore, $f(s) > f(v)$ for any $v \in C$. Since by definition $f(C) = max_{v \in C} f(v)$, this means $f(C) = f(s)$.

Theorem Suppose we run *DFS* starting at each node in G.

Let C_1 and C_2 be *SCC*s in G. If (u, v) is an edge in G where $u \in C_1$ and $v \in C_2$, then $f(C_2) < f(C_1)$.

Proof.

- Let x_1 and x_2 be the first nodes DFS visits in C_1 and C_2, respectively.
- By our lemma, $f(C_1) = f(x_1)$ and $f(C_2) = f(x_2)$. Therefore, we will show $f(x_2) < f(x_1)$.
- Note x_2 is reachable from x_1, since there is a path from x_1 to u in C_1, across (u, v), and there exists a path in C_2 from v to x_2.
- However, x_1 is not reachable from x_2, since then x_1 and x_2 would be *strongly connected*, contradicting that they belong to different *SCC*s.

TWO CASES...

Case 1. $DFS(x_2)$ is called before $DFS(x_1)$:

- Since x_1 is not reachable from x_2, x_1 will not be finished (white) when $DFS(x_2)$ returns.
- Thus x_1 is finished after x_2, so $f(x_2) < f(x_1)$.

Two cases...

Case 2 $DFS(x_1)$ was called before $DFS(x_2)$.

- When $DFS(x_1)$ is called, all nodes in C_1 and C_2 have not been visited, so there will be a *DFS path* from x_1 to x_2.
- Thus when $DFS(x_1)$ returns, x_2 will be white or finished.
- Since $DFS(x_1)$ finishes with x_1 just before it returns, this means that x_1 was finished after x_2, so $f(x_2) < f(x_1)$.

We proved...

Theorem. Suppose we run *DFS* starting at each node in G. Let C_1 and C_2 be *SCC*s in G. If (u, v) is an edge in G where $u \in C_1$ and $v \in C_2$, then $f(C_2) < f(C_1)$.

Now, we can show:

Corollary. Let C_1 and C_2 be distinct *SCC*'s in $G = (V, E)$. Suppose there is an edge (u, v) in E^T where $u \in C_1$ and $v \in C_2$. Then $f(C_1) < f(C_2)$.

Proof.

- Edge $(u, v) \in E^T$ implies $(v, u) \in E$.
- Since *SCC*'s of G and G^T are the same, $f(C_2) > f(C_1)$.
- This completes the proof.

Using the Corollary

Corollary. Let C_1 and C_2 be distinct *SCC*'s in $G = (V, E)$.

Exists $(u, v) \in E^T$ where $u \in C_1$ and $v \in C_2 \longrightarrow f(C_1) < f(C_2)$.

What does the *contrapositive* of this corollary tell us?

Contrapositive.

$f(C_1) > f(C_2) \longrightarrow$ not exists $(u, v) \in E^T$ where $u \in C_1$ and $v \in C_2$

Corollary.

Let C_1 and C_2 be distinct *SCC*'s in $G = (V, E)$, and suppose that $f(C_1) > f(C_2)$, then there cannot be an edge from C_1 to C_2 in G^T.

Tying it all Together

Corollary. Let C_1 and C_2 be distinct *SCC*'s in $G = (V, E)$, and suppose that $f(C_1) > f(C_2)$. Then there cannot be an edge from C_1 to C_2 in G^T.

This means...

- If we start the *DFS* on G^T at the *SCC* C_1 with *maximum* finish time $f(C_1)$ in the first *DFS* on G,
- then, since $f(C_1) > f(C_2)$ for all $C_2 \neq C_1$, there are no edges from C_1 to C_2 in G^T.
- therefore, *DFS* will only visit vertices from C_1.
- if the start vertex in C_1 is x, then $DFS(x)$ returns a *DFS tree* that contains only vertices from C_1.

Tying it all Together

- The next vertex chosen as *starting vertex* in the second *DFS* is in *SCC* C_2 such that $f(C_2)$ is *maximum* over all *SCC*'s other than C_1.
- The *DFS* visits all vertices in C_2. What do we know about but the *edges out* of C_2?
- They only go to C_1, which we've *already visited*.
- Therefore, the only *tree edges* will be to vertices in C_2.
- Continue for all remaining *SCC*s