# 1 Connectivity in Digraphs

How do we achieve a clean description of reachability in a directed graph?

- ullet Note that reachability is no longer symmetric. That is, we can reach w from v but not the other way around.
- Can we maybe allow reachability in either direction?
- Maybe we can allow the ability to follow edges in either direction? However, this treats a digraph as an undirected graph.

# 1.1 Strongly Connected Components

# **Definition 1.1: Strongly Connected Components**

In a directed graph G, two vertices v and w are in the same **strongly connected component** if v is reachable from w and w is reachable from v.

# 1.2 Equivalence Relation

Let  $v \sim w$  if v is reachable from w and vice versa.

**Proposition.** This is an equivalence relation. Namely:

- $v \sim v$  (v is reachable from itself).
- $\bullet \ v \sim w \implies w \sim v \ (\textit{relation is symmetric}).$
- $u \sim v$  and  $v \sim w \implies u \sim w$ .

If we take any v, the set of all w so that  $v \sim w$  is the component of v. Everything connects to everything else in this equivalence class and does not connect (both ways) to anything outside.

#### 1.3 Connectivity

Do strongly connected components completely describe connectivity in G?

• No. In directed cases, we can have an edge between strongly connected components.

### 1.4 Metagraph

#### Definition 1.2: Metagraph

The **metagraph** of a directed graph G is a graph whose vertices are the strongly connected components of G.

## 1.4.1 Result

# Theorem 1.1

The metagraph of any directed graph is a DAG.

# 1.5 Computing SCCs

Given a directed graph G, compute the SCCs of G and its metagraph.

### 1.5.1 Easy Algorithm

- For each v, compute vertices reachable from v.
- Find pairs v, w so that v reachable from w and vice versa.
- For each v, the corresponding w's are in the SCC of v.

The runtime is O(|V|(|V| + |E|)).

### 1.5.2 Better Algorithm

Suppose that SCC(v) is a sink in the metagraph.

- G has no edges from SCC(v) to another SCC.
- We can run explore(v) to find all vertices reachable from v. This will contain all vertices in SCC(v). But, it contains no other vertices.
- If v is in the SCC, then explore(v) finds exactly v's component.

With this observation, we consider the following strategy:

- Find v in a sink SCC of G.
- Run explore(v) to find the component  $C_1$ .
- Repeat process on  $G \setminus C_1$ .

The problem is, how do we find v that is in a sink?

**Proposition.** Let  $C_1$  and  $C_2$  be SCCs of G with an edge from  $C_1$  to  $C_2$ . If we run DFS on G, the largest postorder number of any vertex in  $C_1$  will be larger than the largest postorder number in  $C_2$ .

The reason why we care is because if v is the vertex with the largest postorder number, then:

- There is no edge to SCC(V) from any other SCC.
- SCC is a source SCC.

However, we wanted a sink SCC. So, how do we relate these two?

• A sink is like a source, only with edges going in the opposite direction.

So, we define a reverse graph like so:

### Definition 1.3

Given a directed graph G, the **reverse graph** of G (denoted  $G^R$ ) is obtained by reversing the directions of all the edges of G.

Some properties of reverse graphs are:

- G and  $G^R$  have the same SCCs.
- The sink SCCs of G are the source SCCs of  $G^R$ .
- The source SCCs of G are the sink SCCs of  $G^R$ .