

Decomposition of Graphs: Connectivity

Daniel Kane

Department of Computer Science and Engineering
University of California, San Diego

Graph Algorithms
Data Structures and Algorithms

Learning Objectives

- Understand the importance of connected components of a graph.
- Compute the connected components of a graph.

Outline

① Connected Components

② Algorithm

Reachability

Want to understand which vertices in G are reachable from which others.

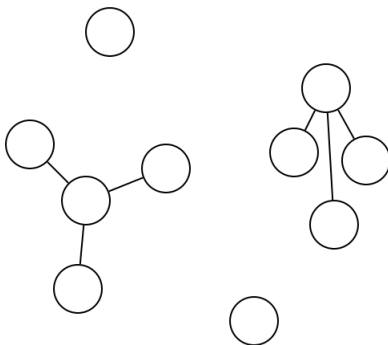
Connected Components

Theorem

The vertices of a graph G can be partitioned into **Connected Components** so that v is reachable from w if and only if they are in the same connected component.

Problem

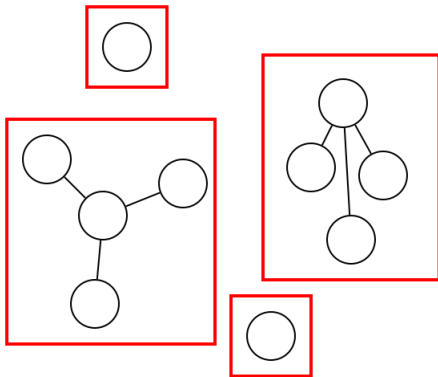
How many connected components does the graph below have?



Solution

How many connected components does the graph below have?

4.



Proof

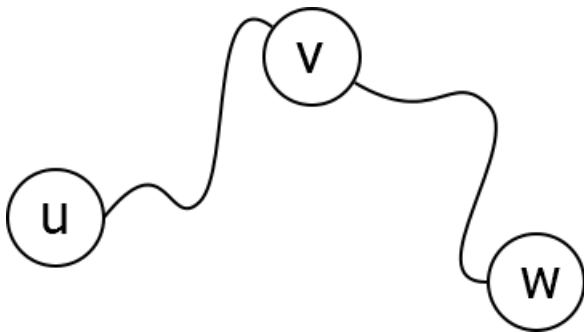
Proof.

Need to show reachability is an **equivalence relation**. Namely:

- v is reachable from v .
- If v reachable from w , w reachable from v .
- If v reachable from u , and w reachable from v , w reachable from u .



Proof (continued)



Problem

Connected Components

Input: Graph G

Output: The connected components of G

Outline

① Connected Components

② Algorithm

Idea

`Explore(v)` finds the connected component of v . Just need to repeat to find other components.

Idea

`Explore(v)` finds the connected component of v . Just need to repeat to find other components.

Modify DFS to do this.

Idea

Explore(v) finds the connected component of v . Just need to repeat to find other components.

Modify DFS to do this.

Modify goal to label connected components.

Modification of Explore

Explore(v)

visited(v) \leftarrow true

CCnum(v) \leftarrow cc

for (v, w) $\in E$:

 if not visited(w):

 Explore(w)

Modifications of DFS

DFS(G)

for all $v \in V$ mark v unvisited

$cc \leftarrow 1$

for $v \in V$:

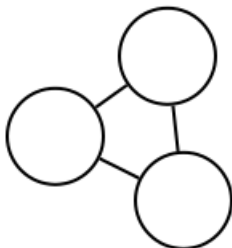
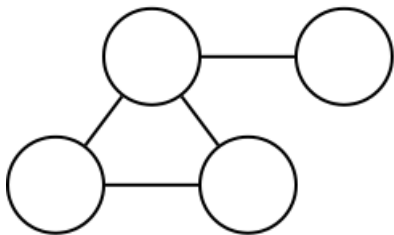
 if not visited(v):

 Explore(v)

$cc \leftarrow cc + 1$

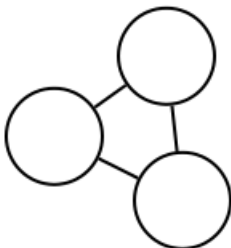
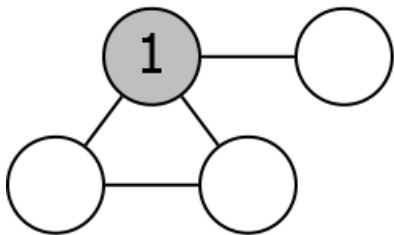
Example

CC: 1



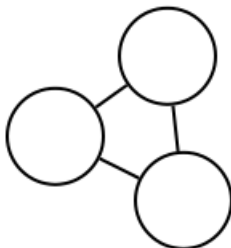
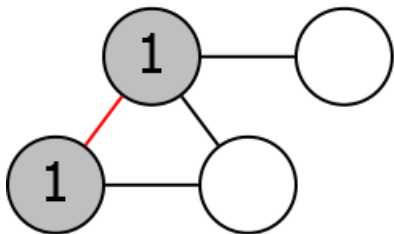
Example

CC: 1



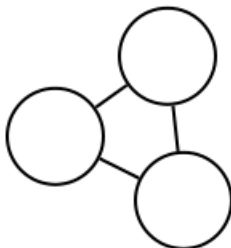
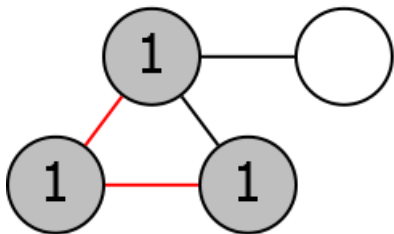
Example

CC: 1



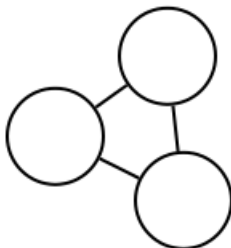
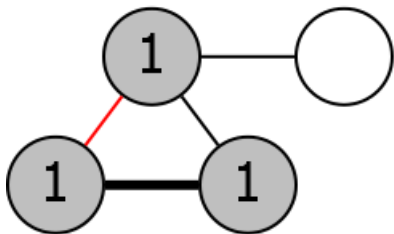
Example

CC: 1



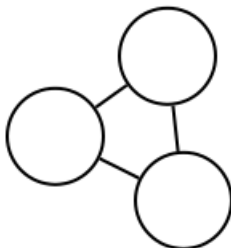
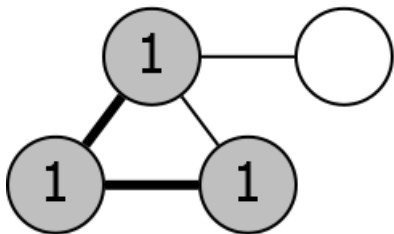
Example

CC: 1



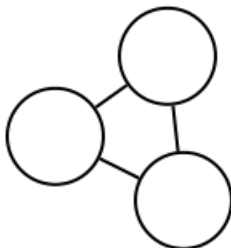
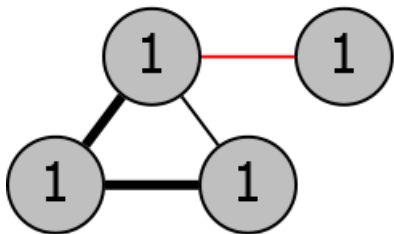
Example

CC: 1



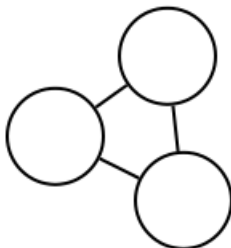
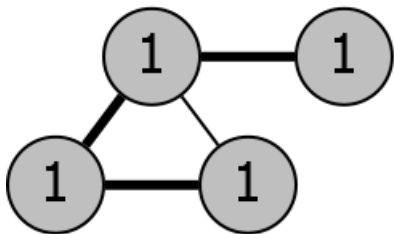
Example

CC: 1



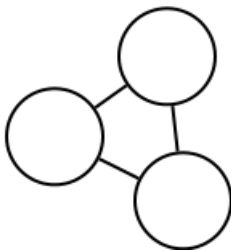
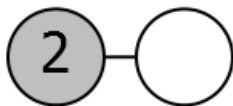
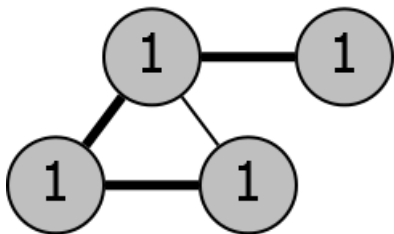
Example

CC: 1



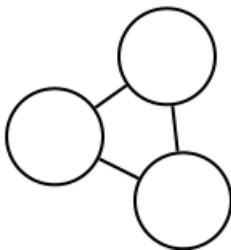
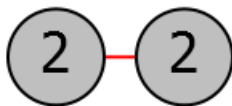
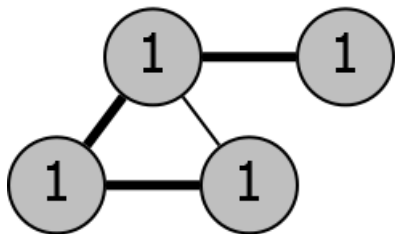
Example

CC: 2



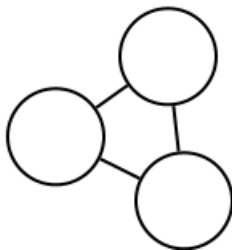
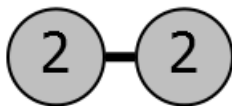
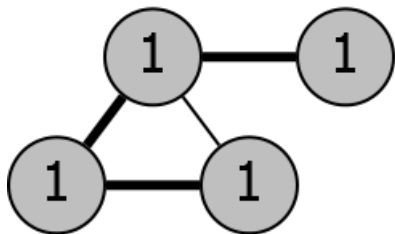
Example

CC: 2



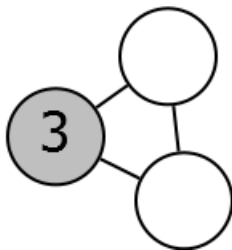
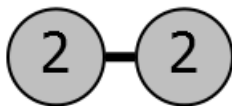
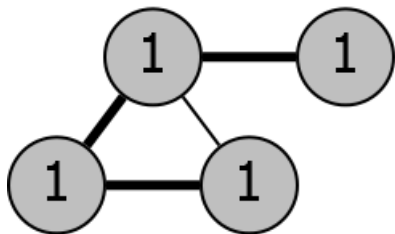
Example

CC: 2



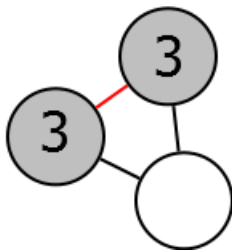
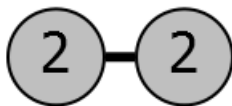
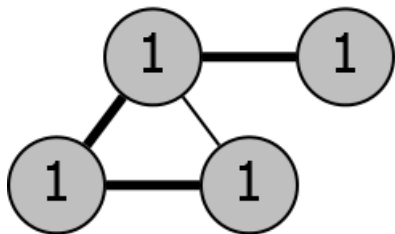
Example

CC: 3



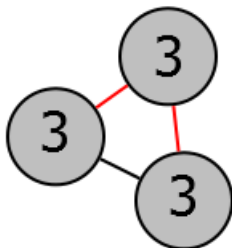
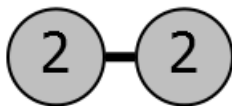
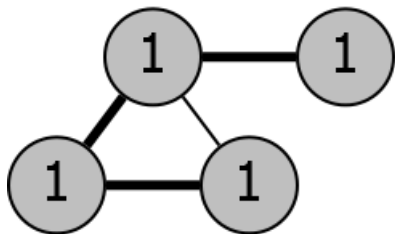
Example

CC: 3



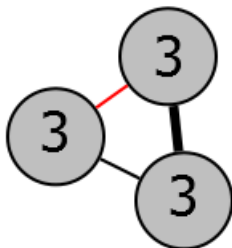
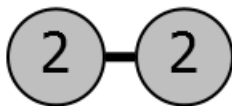
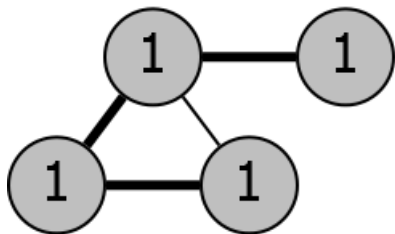
Example

CC: 3



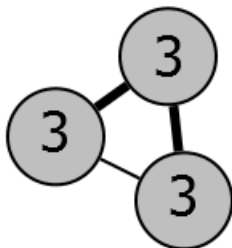
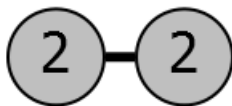
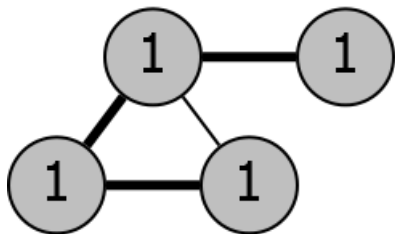
Example

CC: 3



Example

CC: 3



Correctness

- Each new explore finds new connected component.
- Eventually find every vertex.
- Runtime still $O(|V| + |E|)$.

Next Time

Other applications of DFS.