SE284: Introduction to Graph Algorithms

Katerina Taškova

Email: katerina.taskova@auckland.ac.nz
Room: 303S.483

Outline

Strong components, and bipartite graphs

Lecture Notes 28, Textbook 5.7-8

Acknowledgment for slide content: Michael Dinneen, Simone Linz

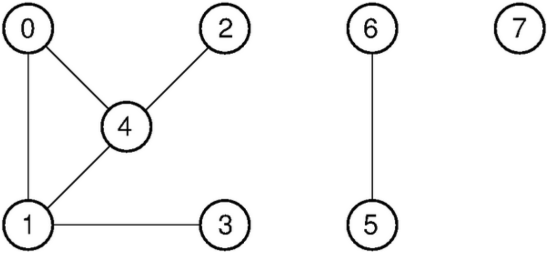
Graph connectivity – reminder

Definition
A graph G is connected if for each pair of vertices $u, v \in V(G)$, there is a path between them.

Definition
A graph G is disconnected if it is not connected and the maximal induced connected subgraphs are called the components of G.

Theorem
Let G be a graph and suppose that DFS or BFS is run on G. Then the connected components of G are precisely the subgraphs spanned by the trees in the search forest.

Connected components – reminder



Nice DFS application: strong components

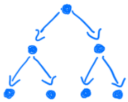
- Nodes v and w are mutually reachable if there is a directed path from v to w and a directed path from w to v. The nodes of a digraph divide up into disjoint subsets of mutually reachable nodes, which induce strong components.
- A digraph is strongly connected if it has only one strong component.
- Components of a graph are found easily by BFS or DFS (each tree in the search forest spans a component). However, this doesn't work well for digraphs (a digraph may have a connected underlying graph yet not be strongly connected). A new idea is needed.

Nice DFS application: strong components

- Nodes v and w are mutually reachable if there is a directed path from v to w and a directed path from w to v. The nodes of a digraph divide up into disjoint subsets of mutually reachable nodes, which induce strong components.
- A digraph is strongly connected if it has only one strong component.
- Components of a graph are found easily by BFS or DFS (each tree in the search forest spans a component). However, this doesn't work well for digraphs (a digraph may have a connected underlying graph yet not be strongly connected). A new idea is needed.

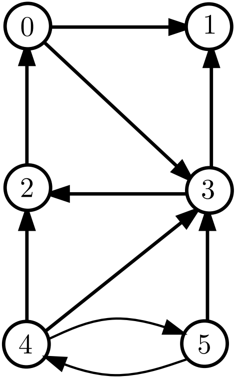
Example 27.13 – reminder

Example 27.13. Find a counter-example to show that the method for finding connected components of graphs in Theorem 27.7 fails at finding strongly connected components of digraphs.

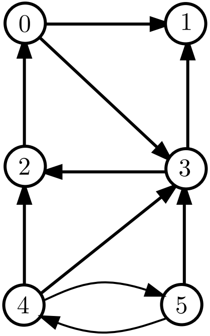


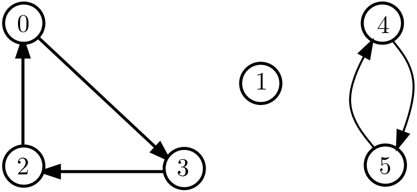
not strongly connected, yet BFS would return a single tree in the search forest

A digraph's strongly connected components



A digraph's strongly connected components





A digraph's strongly connected components

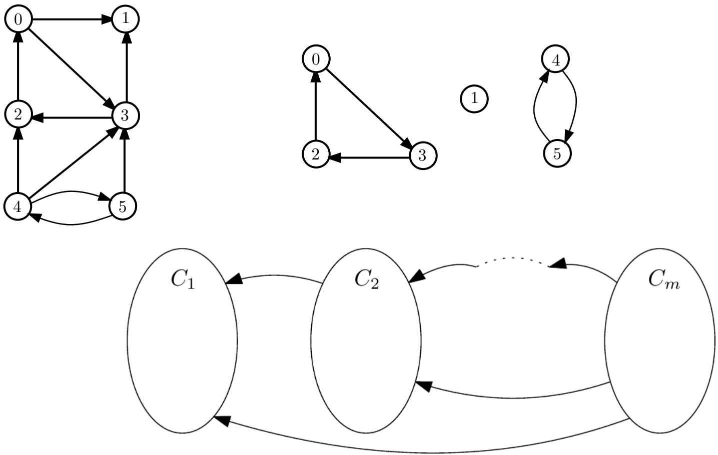
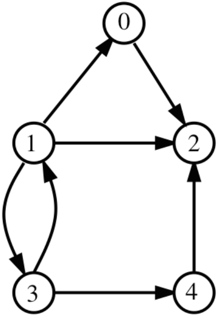
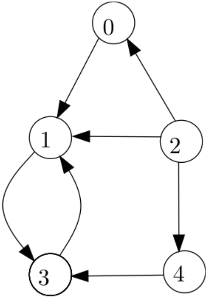


Figure 28.1: Decomposing a graph into its strongly connected components, $C_1, \ldots, C_m$. If each C_i is considered a single node, this decomposition is a DAG.

Reverse digraph



G



G_r

The strongly connected components in G are the same as those in G_r.

Strong components algorithm

- Let G be a digraph, and let G_r be the *reverse digraph* of G obtained by reversing all arcs in G.
- **Phase 1.** Run DFS on G_r, to get depth-first forest F_r. For each node w in G_r, let $done[w]$ be the time at which w turned from grey to black.
- **Phase 2.** Run DFS on G; when choosing a new root, choose a white node w such that $done[w]$ is as large as possible. This gives a forest F.

Strong components algorithm

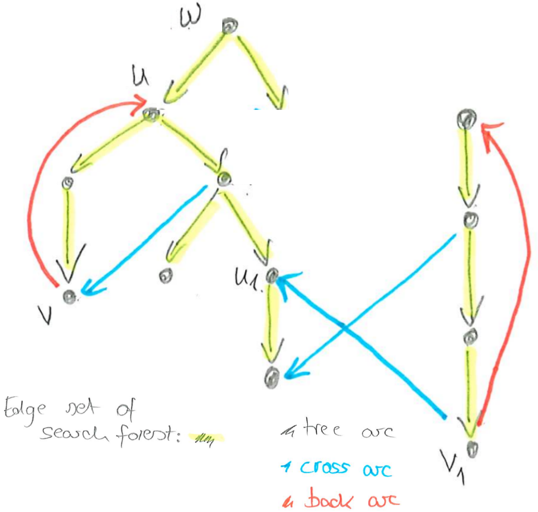
The algorithm's output is F. Want to show that the algorithm works i.e. each strongly connected component in G corresponds to a tree in F.

Claim. A subset V' of all vertices of G is a strongly connected component (i.e. elements in V' are mutually reachable) in G if and only if V' is the node set of a tree in F.

We prove the claim in 2 steps.

Strong components algorithm

Recall that the nodes of the trees in F partition the nodes of G.



Strong components algorithm

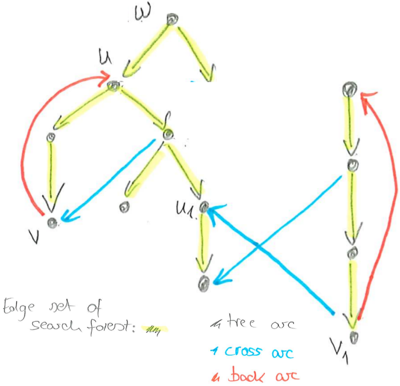
Part 1. If nodes u and v are mutually reachable in G, then the algorithm reports that they are in the same component (tree).

- Wlog, assume that u is visited (marked grey) before v in Phase 2.
- Since there is a directed path from u to v, DFS will visit v after having visited u. Hence, u and v are in the same tree of F and, therefore reported as being in the same component.

Strong components algorithm

Part 2. If the algorithm reports that u and v are in the same component (tree), then they are mutually reachable in G.

- Let T be the tree in F with root w that contains u and v.



Strong components algorithm

Part 2. If the algorithm reports that u and v are in the same component (tree), then they are mutually reachable in G.

- Let T be the tree in F with root w that contains u and v.
- As w was chosen as root, we have $done[w] > done[u]$ and $done[w] > done[v]$ in G_r.
- As w was visited (marked grey) before u in G, there is a directed path from w to u in G and, hence a directed path from u to w in G_r.
- Suppose there is no directed path from w to u in G_r. Then $done[u] > done[w]$ in G_r; a contradiction.
- Hence there is a directed path from w to u in G_r and a directed path from u to w in G.
- And so, w and u are mutually reachable.

Strong components algorithm

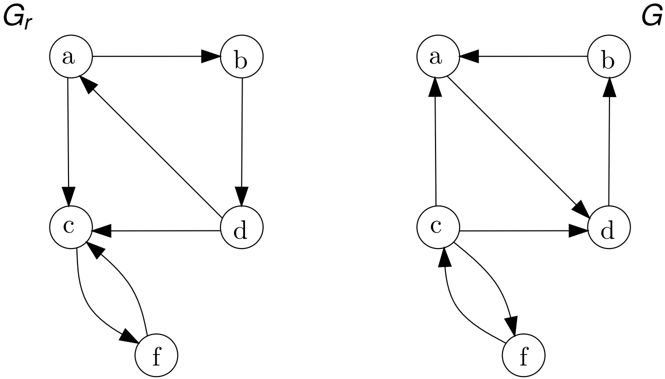
Part 2. If the algorithm reports that u and v are in the same component (tree), then they are mutually reachable in G.

- Let T be the tree in F with root w that contains u and v.
- As w was chosen as root, we have $done[w] > done[u]$ and $done[w] > done[v]$ in G_r.
- As w was visited (marked grey) before u in G, there is a directed path from w to u in G and, hence a directed path from u to w in G_r.
- Suppose there is no directed path from w to u in G_r. Then $done[u] > done[w]$ in G_r; a contradiction.
- Hence there is a directed path from w to u in G_r and a directed path from u to w in G.
- And so, w and u are mutually reachable.

Strong components algorithm

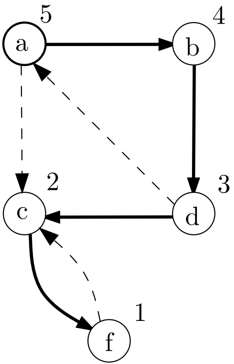
- Use analogous argument to show that w and v are mutually reachable.
- Then u and v are mutually reachable.
- This completes the proof of the claim.
- *Convince yourself that this is indeed true!*

Strong connected components example

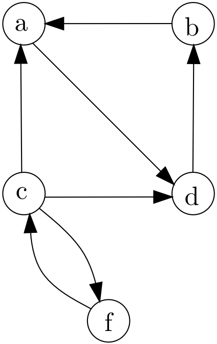


Strong connected components example

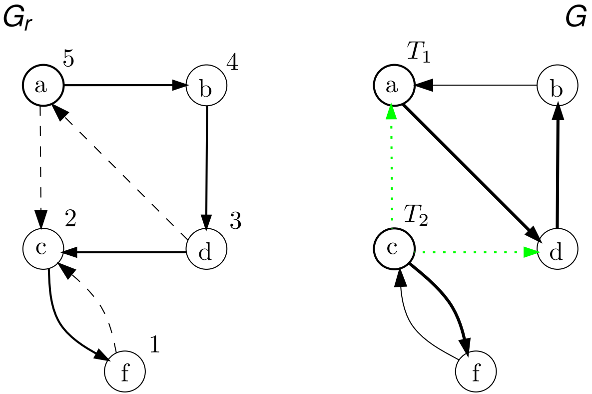
G_r



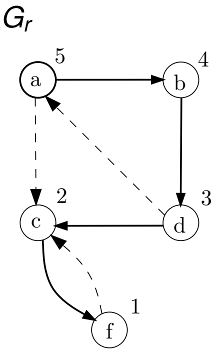
G

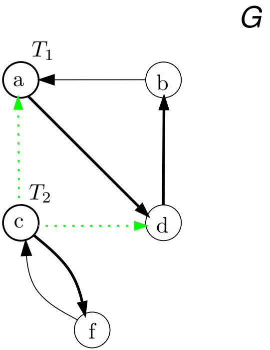


Strong connected components example



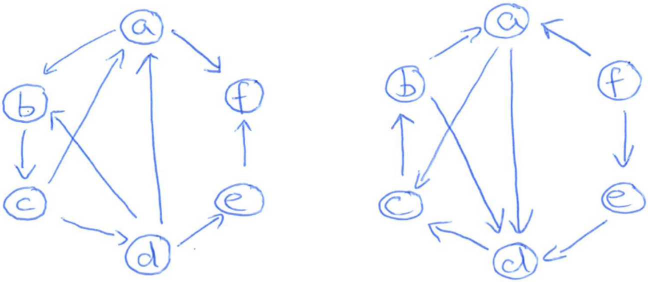
Strong connected components example





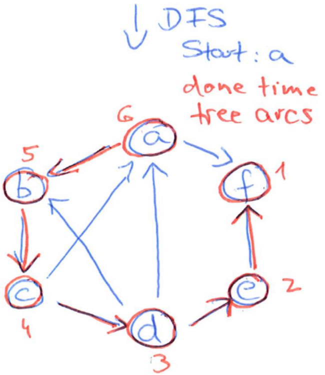
Strong connected components example

G_r G

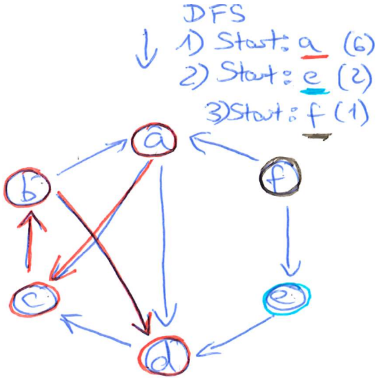


Strong connected components example

G_r

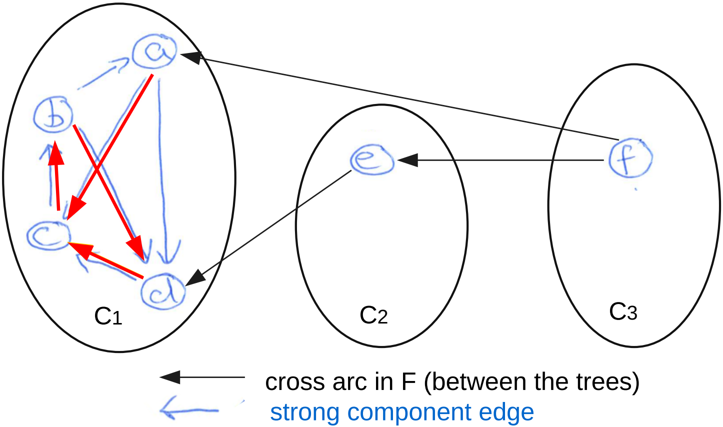


G



G has 3 strong components.

Strong connected components example

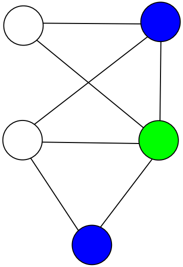


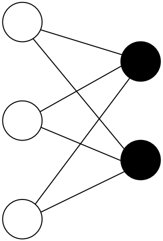
G has 3 strong components.

k-colorable graphs

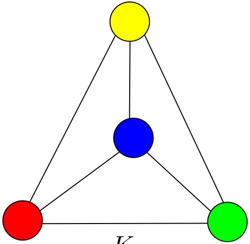
Definition

Let k be a positive integer. A graph G has a k-coloring if $V(G)$ can be partitioned into k nonempty disjoint subsets such that each edge of G joins two vertices in different subsets (colors). The smallest number of colors needed to color a graph is called chromatic number.





$K_{3,2}$

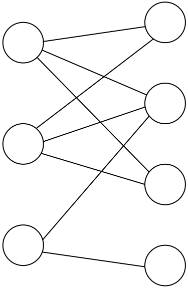


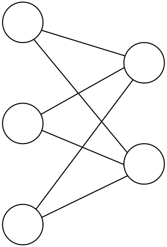
K_4

Bipartite graphs (digraphs)

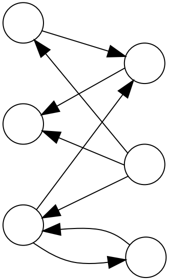
Definition

A graph G is bipartite if $V(G)$ can be partitioned into two nonempty disjoint subsets V_1, V_2 such that each edge of G has one endpoint in V_1 and one in V_2. [Similar for digraphs.]





$K_{3,2}$



Bipartite graphs

Theorem

The following conditions on a graph G are equivalent.

1. *G has a 2-coloring;*
2. *G is bipartite;*
3. *G does not contain an odd length cycle.*

Proof: show that 1 implies 2, then 2 implies 3, and 3 implies 1.

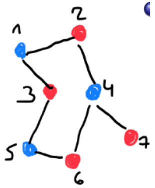
Bipartite graphs

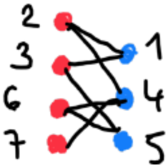
Theorem

The following conditions on a graph G are equivalent.

1. *G has a 2-coloring;*
2. *G is bipartite;*
3. *G does not contain an odd length cycle.*

- Suppose G has a 2-coloring. Let V_1 be the set of vertices with color c_1, and let V_2 be the set of vertices with color c_2. Then each edges joins a vertex in V_1 with a vertex in V_2. By definition, $G = (V_1 \cup V_2, E)$ is bipartite.





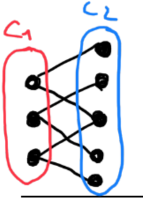
Bipartite graphs

Theorem

The following conditions on a graph G are equivalent.

1. *G has a 2-coloring;*
2. *G is bipartite;*
3. *G does not contain an odd length cycle.*

- Suppose $G = (V_1 \cup V_2, E)$ is bipartite. Each edges joins a vertex in V_1 with a vertex in V_2. Color each vertex in V_1 with color c_1 and each vertex in V_2 with color c_2. Since G is bipartite , this induces a 2-coloring of G.



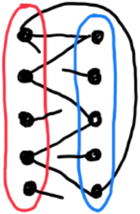
Bipartite graphs

Theorem

The following conditions on a graph G are equivalent.

1. *G has a 2-coloring;*
2. *G is bipartite;*
3. *G does not contain an odd length cycle.*

- Suppose G is bipartite. Let C be a cycle in G. Then, since G is 2-colorable, C has even length (start and end vertex have different colors). Hence, G does not contain a cycle of odd length.



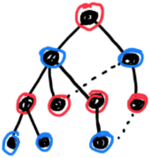
Bipartite graphs

Theorem

The following conditions on a graph G are equivalent.

1. *G has a 2-coloring;*
2. *G is bipartite;*
3. *G does not contain an odd length cycle.*

- Suppose G has no cycle of odd length. Obtain a 2-coloring as follows: Start BFS at v, assign v to color c_1, assign all neighbors of v to color c_2, assign all neighbors of neighbors of v to color c_1 and continue in this way until all vertices are colored. Since there is no odd cycle, each cross edge joins vertices of different color. (Why?)



Deciding if a graph is bipartite

Fact

A version of BFS can be used to check if a graph is bipartite (e.g. 2-colorable).

Thank you!