# 15-451 Algorithms, Spring 2015

**Recitation # 6** **Feb 16th, 2015**

1. **Strongly connected components**

   A strongly connected graph is a directed graph with the property that for any pair of vertices u, v, there exists a path from u to v and a path from v to u. A strongly connected component of a graph is a maximal set of vertices with the strongly connected property. Note that any directed graph can be decomposed into strongly connected components; that is, a directed graph's strongly connected components form a partition over the graph's vertex set.

   Recall that a partition determines an equivalence relation such that the sets in the partition form equivalence classes. As such, the strongly connected components of a graph form equivalence classes over its vertices.

   (a) How can the number of strongly connected components of a graph change if a new edge is added?

   > **Solution:** The number of strongly connected components can only decrease as the number of edges increase. If the edge is between two vertices that are already in the same strongly connected component, the number of components will not change. If the edge connects two vertices that belong to distinct scc's, then the number of components will decrease by at most one.

   (b) One interesting property of strongly connected components is that when you contract each strong component into a single vertex, the resulting graph is a directed acyclic graph. The contracted graph is sometimes known as the component graph. More precisely: let $C$ and $C'$ be distinct strongly connected components in directed graph G. Consider vertices $u, v \in C$ and $u', v' \in C'$. Show that if there exists a path in G from $u \to u'$, then there does not exist a path from $v' \to v$.

   > **Solution:** Suppose for the sake of contradiction that there exists a path from $v' \to v$. It follows from the strongly connected nature of $C$ and $C'$ that there are paths $u' \to v'$ and $v \to u$. But then there exist both paths $u \to u'$ and $u' \to v' \to v \to u$, so u and v' are reachable from each other. This contradicts the assumption that $C$ and $C'$ are distinct strongly connected components. Done.

   (c) Give an algorithm to find the component graph of a directed graph. That is, given a connected directed graph G and an $O(V + E)$ algorithm to find the strongly connected components, compute the dag C whose vertices are each a strongly connected component of G. What is the runtime of this algorithm?

**Solution:** Assume that we have some mapping from the vertices to components $scc(v)$. Let the components be labeled $\{1, 2, ..., |V|\}$, since there can't be more strongly connected components than there are vertices. First, we want to construct the vertex set of C, let's call it $V_C$. We take the vertex set of G and calling *scc* on each one, so we now have a list of components. Using counting sort, sort this list. Then walk through it, discarding any element that is the same as the previous element. This gives us a set $\{1, ..., |V_C|\}$. Constructing the vertex set took $O(V)$ time.

Now to find the edge set of C, $E_C$. First construct the set of ordered pairs $S = \{(x, y) |$ edge $(u, v) \in E, x = scc(u), y = scc(v), x \neq y\}$. This takes $O(E)$ time, since we simply iterate through each edge of G and call scc on each edge's endpoints. S contains at most $|E|$ elements. Now we just need to do two passes of counting sort, which takes $O(V + E)$ time, in order to sort the elements of S and fold them to remove duplicities. This will yield the edge set $E_C$.

The total runtime is $O(V + E)$.

2. **Biconnected Components**

A biconnected graph is a 2-vertex connected graph, so it cannot be disconnected by the removal of a single vertex. In other words, a biconnected graph has no articulation vertices.

Note that for any pair of vertices in a biconnected component, there are two disjoint paths between them. (This follows from Menger's theorem, which states that for any k-vertex connected graph G, there exist k distinct paths between every pair of vertices in G). This implies that for any pair of of vertices in a biconnected component, there must be a simple cycle.

As we saw above, strongly connected components determine equivalence classes of vertices in a graph. How do biconnected components form a partition over a graph?

(a) Show that the biconnected components of a graph divide the edges into equivalence classes. That is, show that two edges in a graph G are equivalent if and only if there exists a simple cycle in the graph that includes both of the edges.

**Solution:** This relation is obviously reflexive and symmetric. Showing transitivity requires a little more work.

Claim: If $e_1 \sim e_2$ and $e_2 \sim e_3$, then $e_1 \sim e_3$.

Proof: Let C be the cycle containing $e_1$ and $e_2$. If $e_3$ is on this cycle, then

we are done. If not, then we know that there exists a cycle that contains both $e_2$ and $e_3$. I claim that the symmetric difference of the $e_1e_2$ cycle and $e_2e_3$ cycle will form a cycle containing $e_1$ and $e_3$. Consider the endpoints of $e_2$ and $e_3$, let's call them (x, y) and (u, v), respectively. Since they are part of a biconnected component, there must be distinct paths from $u \to x$ and $v \to y$. Follow each of these paths until they meet a vertex on C. This must happen, since at the very least we know that both these paths and C contain x and y, the endpoints of $e_2$. Then we can simply concatenate each path with the remainder of C that leads to an endpoint of $e_1$.

Less formally, you can see that no matter how these paths meet the cycle C, there is a way to create a simple cycle containing $e_1$ and $e_3$ by gluing part of cycle C and the blue and green paths (in the examples below) together. For example, to construct the desired simple cycle in the first picture, we go from the top end of $e_3$, along the green path, then continue counterclockwise around C passing through $e_1$, then continue around C exiting along the blue path back to $e_3$. The second case is similar but when we hit C on the green path we go around C clockwise, pass through $e_1$, then exit on the blue path.

Done.