Introduction to Algorithms **The Edmonds-Karp Max-Flow Algorithm**
CS 4820, Spring 2018

These lecture notes present the Edmonds-Karp maximum flow algorithm. We'll assume familiarity with the basic notions of *residual graph*, *augmenting path*, and *bottleneck capacity*. Recall that the Ford-Fulkerson algorithm is the following algorithm for the maximum flow problem.

Algorithm 1 FORDFULKERSON(G)

$f \leftarrow 0$; $G_f \leftarrow G$
while G_f contains an $s - t$ path P **do**
  Let P be one such path.
  Augment f using P.
  Update G_f
end while
return f

The algorithm's running time is pseudopolynomial, but not polynomial. We've seen an example illustrating that a bad choice of augmenting paths can cause the Ford-Fulkerson algorithm to run for an exponential number of steps. In fact, when the edge capacities are allowed to be real-valued (rather than integer-valued) there exist executions of the Ford-Fulkerson algorithm that never terminate!

The Edmonds-Karp algorithm refines the Ford-Fulkerson algorithm by always choosing the augmenting path with the smallest number of edges. In these notes, we will analyze the algorithm's running time and prove that it is polynomial in m and n (the number of edges and vertices of the flow network).

Algorithm 2 EDMONDSKARP(G)

$f \leftarrow 0$; $G_f \leftarrow G$
while G_f contains an $s - t$ path P **do**
  Let P be an $s - t$ path in G_f with the minimum number of edges.
  Augment f using P.
  Update G_f
end while
return f

To begin our analysis of the Edmonds-Karp algorithm, note that the $s - t$ path in G_f with the minimum number of edges can be found in $O(m)$ time using breadth-first search. (Generally, breadth-first search in a graph with n vertices and m edges requires $O(m + n)$ time, but our standing assumption that every vertex of the graph has at least one incident edge implies that $n \le 2m$ from which it follows that $O(m + n) = O(m)$.) Once path P is discovered, it takes only $O(n)$ time to augment f using P and $O(n)$ time to update G_f, so — again using the fact that $n = O(m)$ — we see that one iteration of the **while** loop in EDMONDSKARP(G) requires only $O(m)$ time. However, we still need to figure out how many iterations of the **while** loop could take place, in the worst case.

To reason about the maximum number of **while** loop iterations, we take an indirect approach based on thinking about the breadth-first search tree of G_f, starting from s. (Henceforth we call this the *BFS tree* for short.) Recall that the vertices of the BFS tree can be organized into levels $L_0, L_1, \ldots, L_k$, where $L_0 = \{s\}$ and L_i $(i > 0)$ consists of all the vertices v such that the path from s to v in the BFS tree has i edges. An elementary and useful property of BFS is the following: for all $v \in L_j$, every shortest path from s to v contains exactly one vertex from each of levels $L_0, L_1, \ldots, L_j$ (in that order) and no other vertices. In particular, every time the Edmonds-Karp algorithm chooses an augmenting path, that path consists of vertices $s = v_0, v_1, \ldots, v_j = t$ with $v_i \in L_i$ for $0 \leq i \leq j$.

Let us consider how the graph G_f changes when we augment f using P.

- If P contains a forward edge e, then edge e may be deleted from G_f (if the augmentation saturates e) and the backward edge $\overleftarrow{e}$ may be added to G_f (if G_f did not contain $\overleftarrow{e}$ before the augmentation).
- If P contains a backward edge $\overleftarrow{e}$, then $\overleftarrow{e}$ may be deleted from G_f (if the augmentation eliminates all flow on e) and the forward edge e may be added to G_f (if e had previously been saturated before the augmentation).
- No other edges are added or deleted.
- Thus, every new edge that is created when augmenting f using P is the reverse of an edge that belongs to P.

Recalling that every edge of P goes from level i to $i + 1$, for some $0 \leq i < j$, we see that every new edge that gets created in G_f after the augmentation must go from level $i + 1$ to level i, for some $0 \leq i < j$. In particular, for any vertex v, the distance from s to v never decreases as we run the Edmonds-Karp algorithm! (Creating edges that point from a higher-numbered level of the BFS tree to a lower-numbered level can never produce a "shortcut" that reduces the length of the shortest path from s to v.) This is the key property that guides our analysis of the algorithm.

When we choose augmenting path P in G_f, let us say that edge $e \in E(G_f)$ is a bottleneck edge for P if $c_f(e) = \text{bottleneck}(f, P)$. Notice that if $e = (u, v)$ is a bottleneck edge for P, then it is eliminated from G_f after augmenting f using P. Suppose that $u \in L_i$ and $v \in L_{i+1}$ when this happens. In order for e to be added back into G_f later on, u must occupy a higher-numbered level than v. (Recall that edges are only added to G_f when they point from one level to the immediately *preceding* level.) Since the distance from s to v never decreases, this means that v remains in level L_{i+1} or higher, and u must rise to level L_{i+2} or higher, before e is added back into G_f. The BFS tree has no levels numbered above n. Thus, the total number of times that e can occur as a bottleneck edge during the Edmonds-Karp algorithm is at most $n/2$. There are $2m$ edges that can potentially appear in the residual graph, and each of them serves as a bottleneck edge at most $n/2$ times, so there are at most mn bottleneck edges in total. Every iteration of the **while** loop identifies an augmenting path, and that augmenting path must have a bottleneck edge, so there are at most mn **while** loop iterations in total. Earlier, we saw that every iteration of the loop takes $O(m)$ time, so the running time of the Edmonds-Karp algorithm is $O(m^2 n)$.

Faster network flow algorithms have been discovered. There is an algorithm due to Dinic that is very similar in spirit to Edmonds-Karp but achieves The preflow-push algorithm, presented in Section 7.4 of Kleinberg-Tardos, has a running time of $O(n^3)$. Even faster algorithms are known, but they are beyond the scope of this course.