

Flows in Networks: The Edmonds-Karp Algorithm

Daniel Kane

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

Advanced Algorithms and Complexity
Data Structures and Algorithms

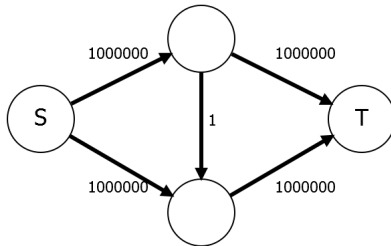
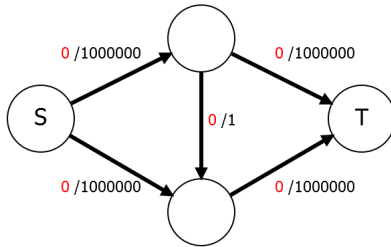
Learning Objectives

- Implement the Edmonds-Karp algorithm.
- Understand the runtime bound.

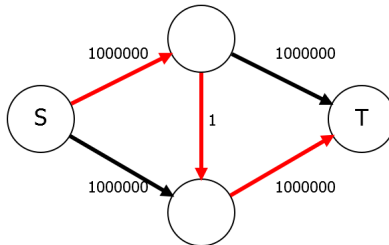
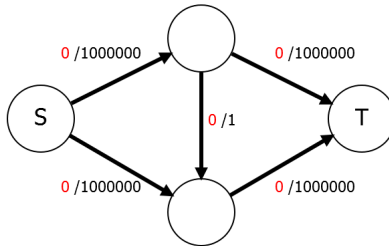
Last Time

- Ford-Fulkerson algorithm for Maxflow.
- Runtime $O(|E||f|)$.
- Sometimes very slow if graph has large capacities.

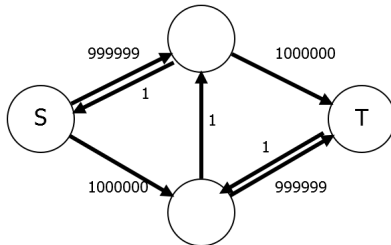
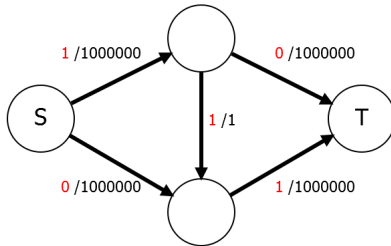
Bad Example



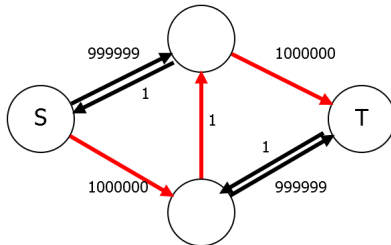
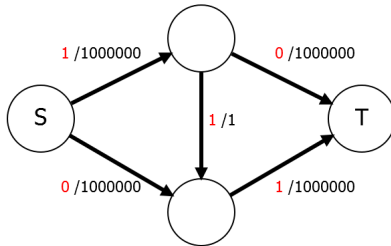
Bad Example



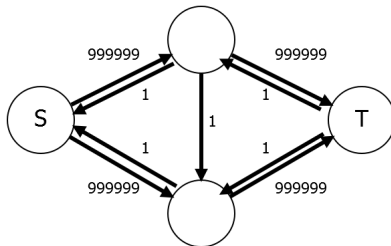
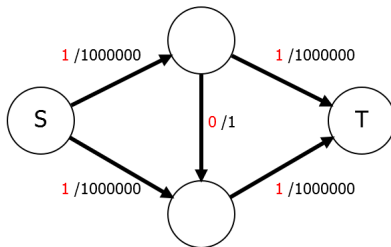
Bad Example



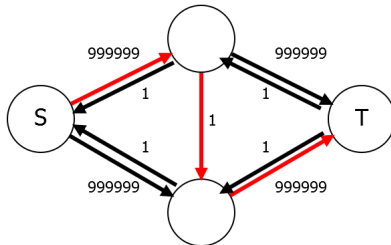
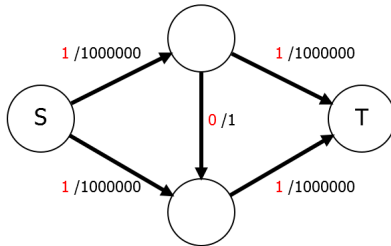
Bad Example



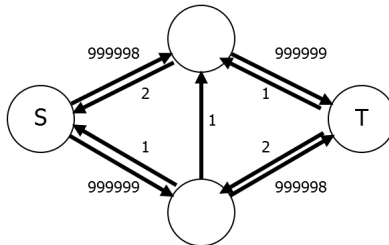
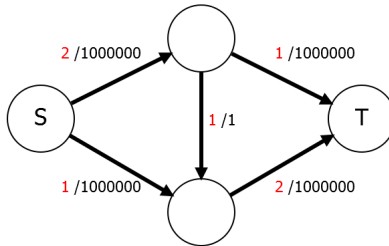
Bad Example



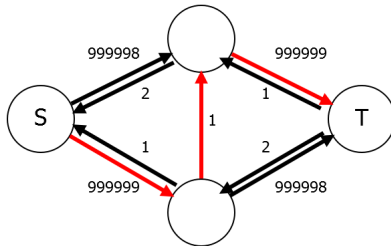
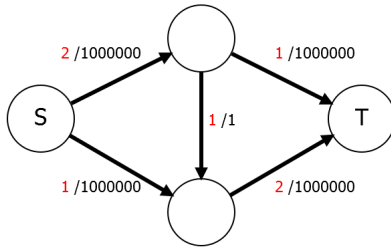
Bad Example



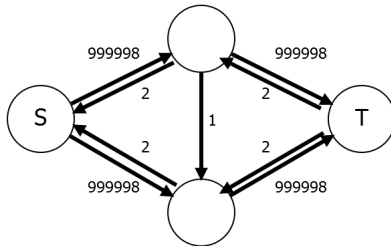
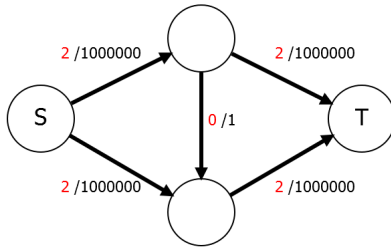
Bad Example



Bad Example



Bad Example



Fix

Fortunately, the Ford-Fulkerson algorithm gives us a **choice** as to which augmenting path to use.

Fix

Fortunately, the Ford-Fulkerson algorithm gives us a **choice** as to which augmenting path to use.

Is there a way to choose augmenting paths to avoid this kind of runtime problem?

Edmonds-Karp Algorithm

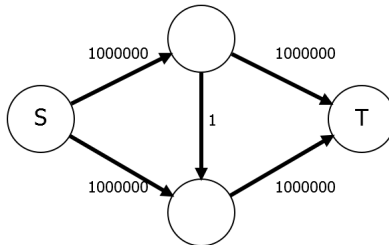
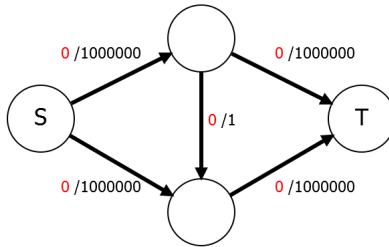
Use the Ford-Fulkerson algorithm, always choosing the **shortest** (in terms of number of edges) augmenting path.

Edmonds-Karp Algorithm

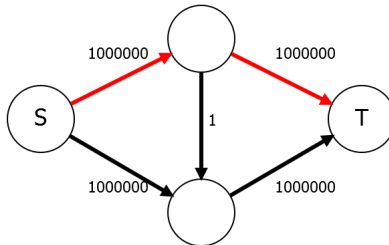
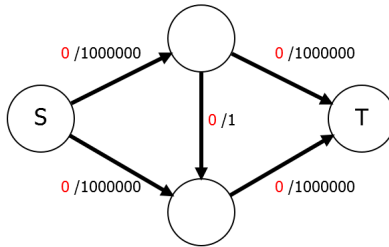
Use the Ford-Fulkerson algorithm, always choosing the **shortest** (in terms of number of edges) augmenting path.

Find augmenting paths using BFS instead of DFS.

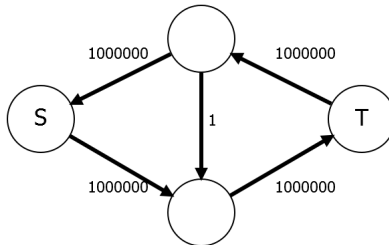
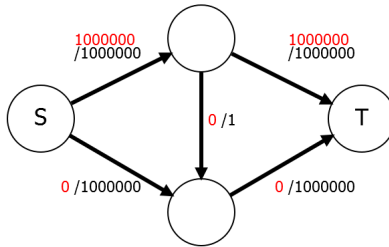
Edmonds-Karp Execution



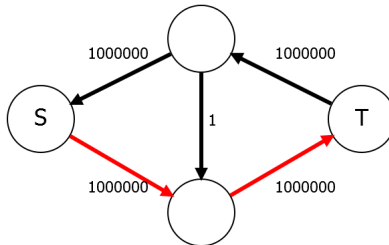
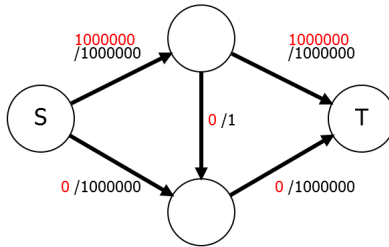
Edmonds-Karp Execution



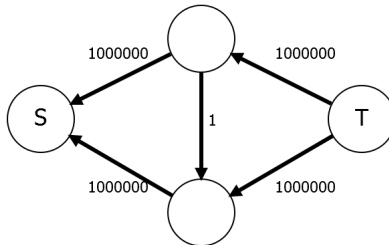
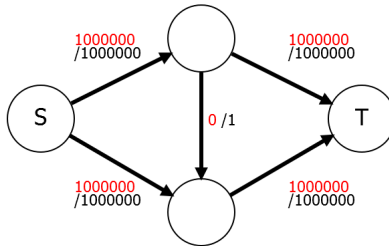
Edmonds-Karp Execution



Edmonds-Karp Execution



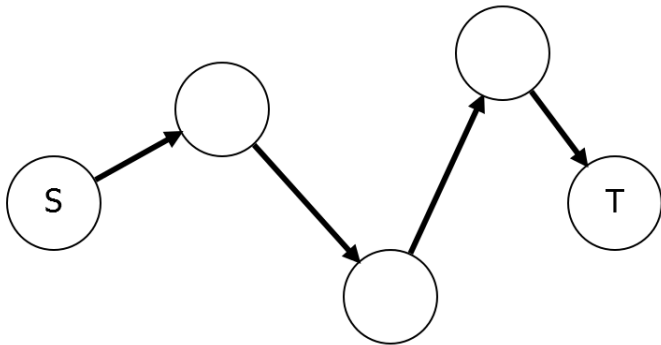
Edmonds-Karp Execution



Augmenting Paths

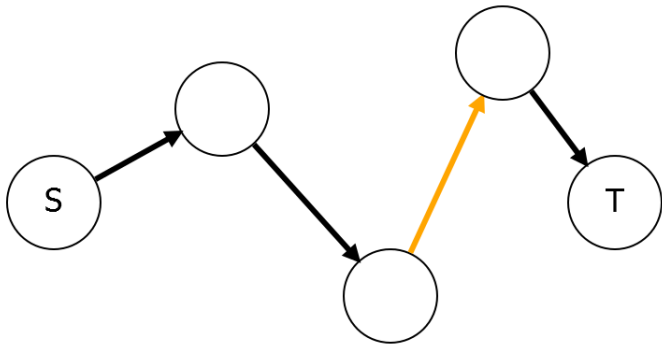
Need to analyze augmenting paths.

.



Augmenting Paths

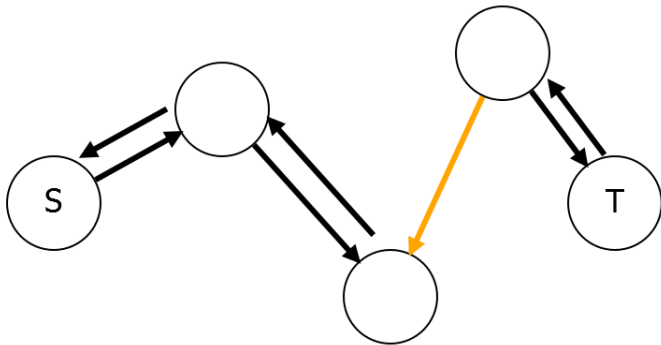
Augmenting flow always saturates (uses all the available flow from) an edge.



Augmenting Paths

Changes to residual network.

.



Analysis Idea

- Show that no edge is saturated too many times.
- Fails to hold in the bad case, where the middle edge is repeatedly saturated.

Increasing Distances

We will need this critical Lemma:

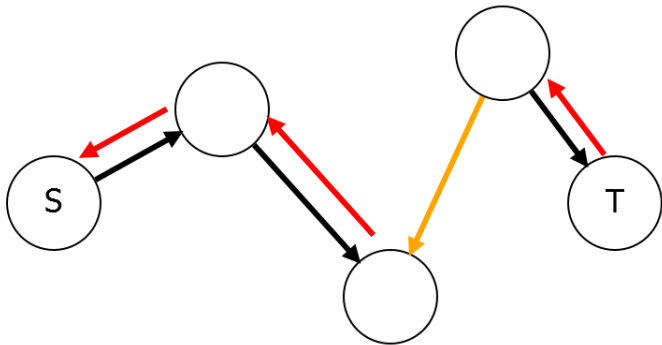
Lemma

As the Edmonds-Karp algorithm executes, for any vertex $v \in V$ the distance $d_{G_f}(s, v)$ only increases.

Similarly, $d_{G_f}(v, t)$ and $d_{G_f}(s, t)$ can only increase.

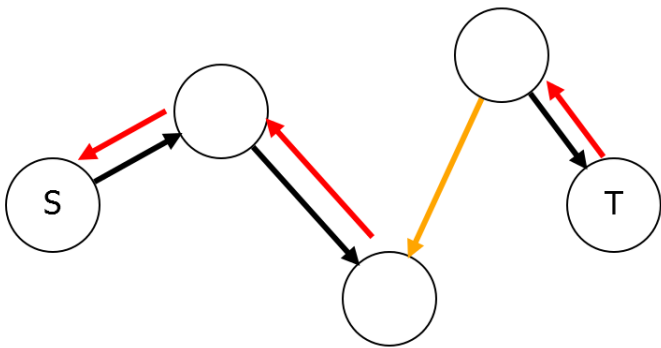
Proof

- New edges all point backwards along augmenting path.



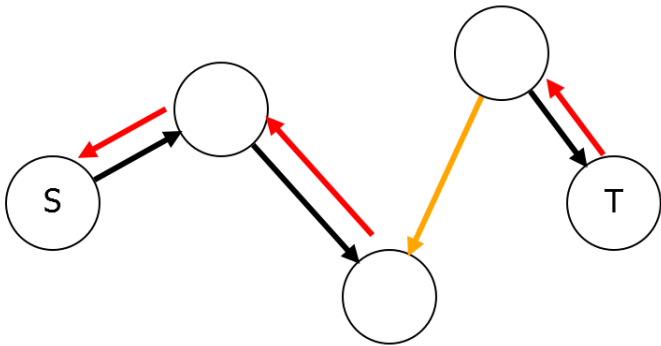
Proof

- Augmenting path is a shortest path.



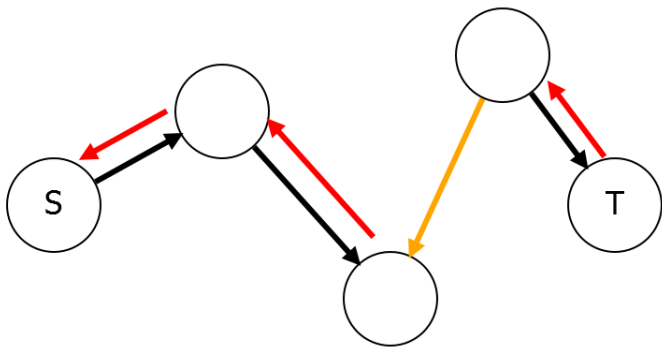
Proof

- All new edges point from vertices further from s to vertices closer.



Proof

- New edges do not help you get from s to v faster.



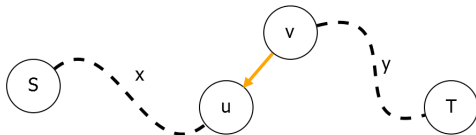
Reuse Limit

Lemma

When running the Edmonds-Karp algorithm, if an edge e is saturated, it will not be used in an augmenting path again, until $d_{G_f}(s, t)$ increases.

Proof

- Initially, $d(s, u) = x$, $d(v, t) = y$, $d(s, t) = x + y + 1$.
- When used again, $d(s, v) \geq x + 1$, $d(u, t) \geq y + 1$.
- Therefore, when used again, $d(s, t) \geq (x+1) + (y+1) + 1 = x + y + 3$.



Analysis

- $d_{G_f}(s, t)$ can only increase $|V|$ times.
- Each time can only have $O(|E|)$ many saturated edges.
- Therefore, only $O(|V||E|)$ many augmenting path.
- Each path takes $O(|E|)$ time.
- Total runtime: $O(|V||E|^2)$.

Problem

Which of the following is true about the Edmonds-Karp algorithm:

- 1 No edge is saturated more than $|V|$ times.
- 2 The lengths of the augmenting paths decrease as the algorithm progresses.
- 3 Changing the capacities of edges will not affect the runtime.

Solution

Which of the following is true about the Edmonds-Karp algorithm:

- 1 No edge is saturated more than $|V|$ times.
- 2 The lengths of the augmenting paths decrease as the algorithm progresses.
- 3 Changing the capacities of edges will not affect the runtime.

Summary

- Choose augmenting paths based on path length.
- Removes runtime dependence on numerical sizes of capacities.

Summary

- Choose augmenting paths based on path length.
- Removes runtime dependence on numerical sizes of capacities.
- There are better, much more complicated algorithms.
- State of the art $O(|V||E|)$.

Next Time

Applications of Maxflow algorithms.