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

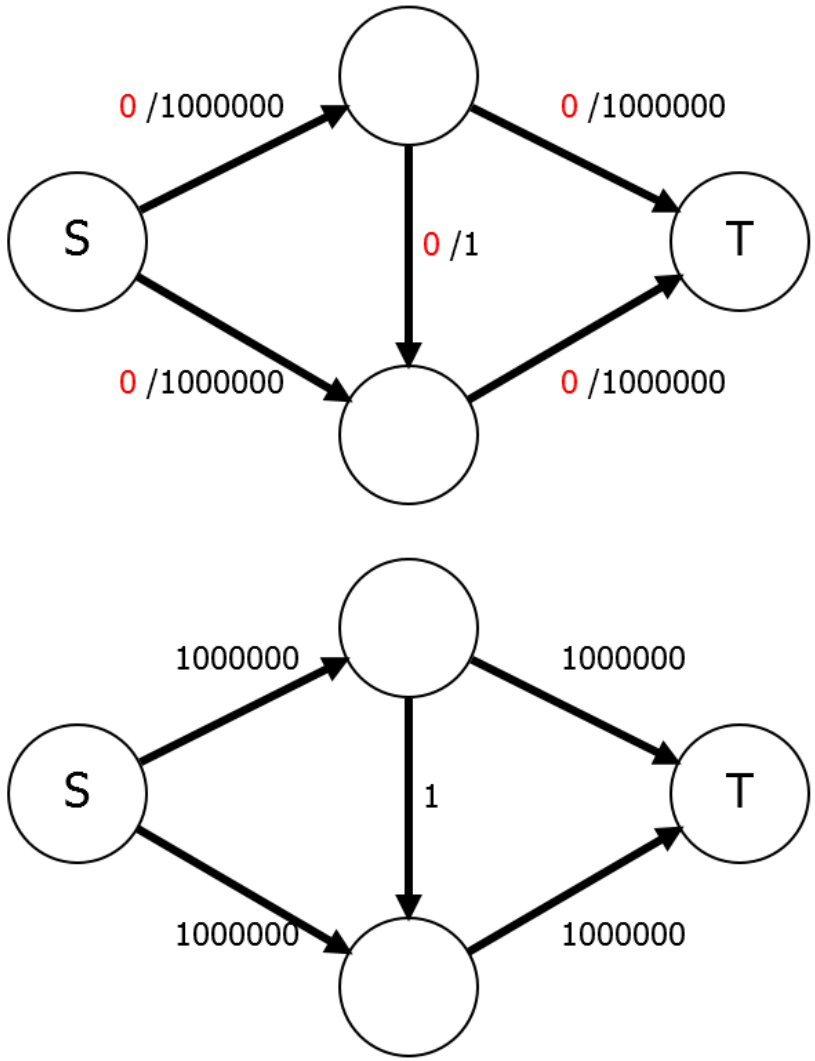
Learning Objectives

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

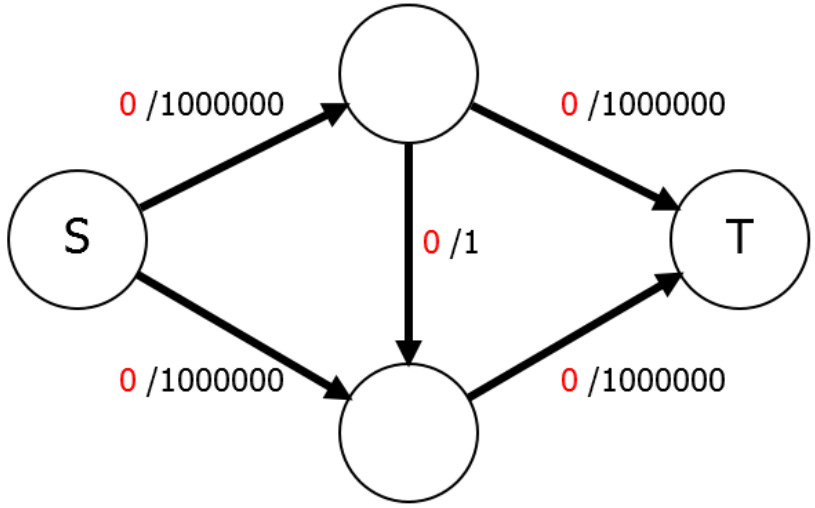
Last Time

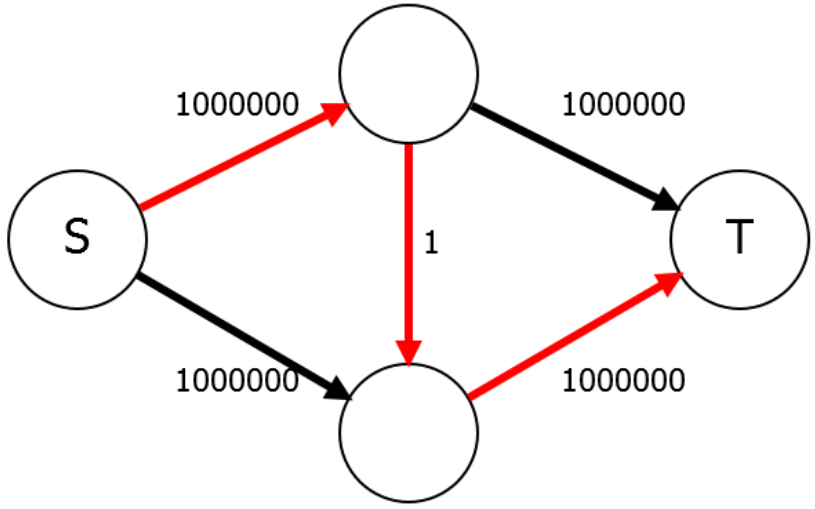
- Ford-Fulkerson algorithm for Maxflow.
- Runtime $O(|E||f|)$.
- Sometimes very slow if graph has large capacities.

Bad Example


0 /1000000
0 /1000000
S
0 /1
T
0 /1000000
0 /1000000
1000000
1000000
S
1
T
1000000
1000000

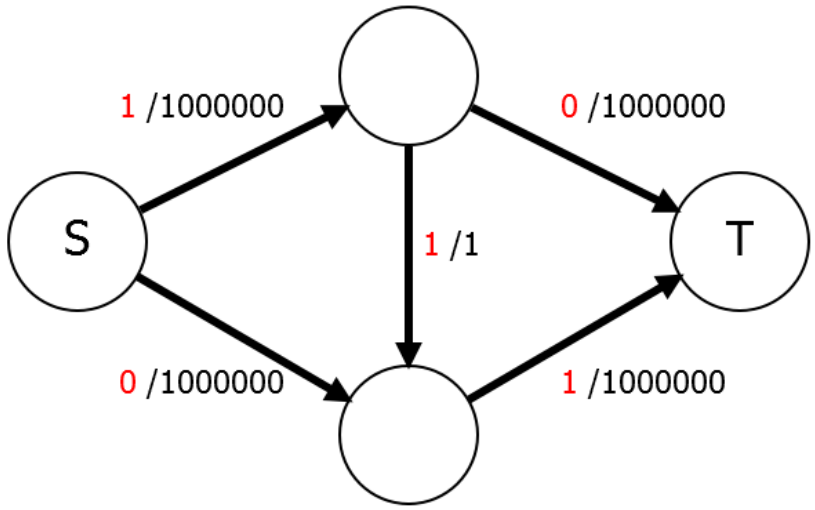
Bad Example

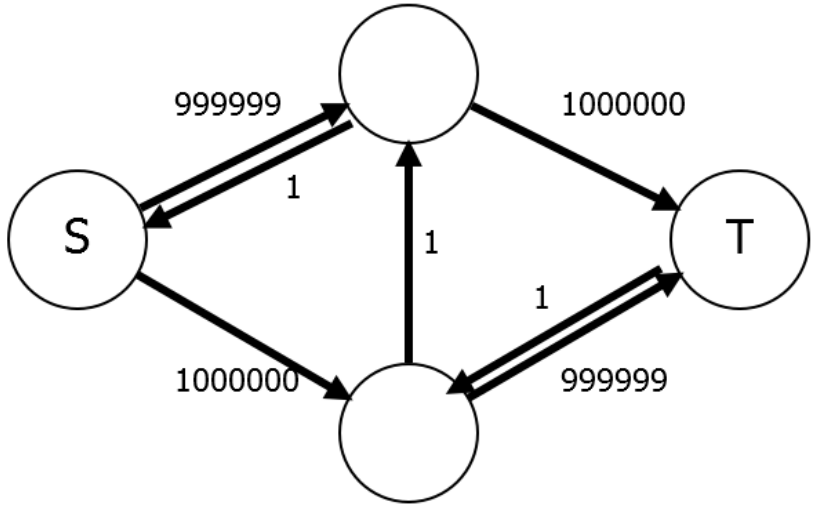




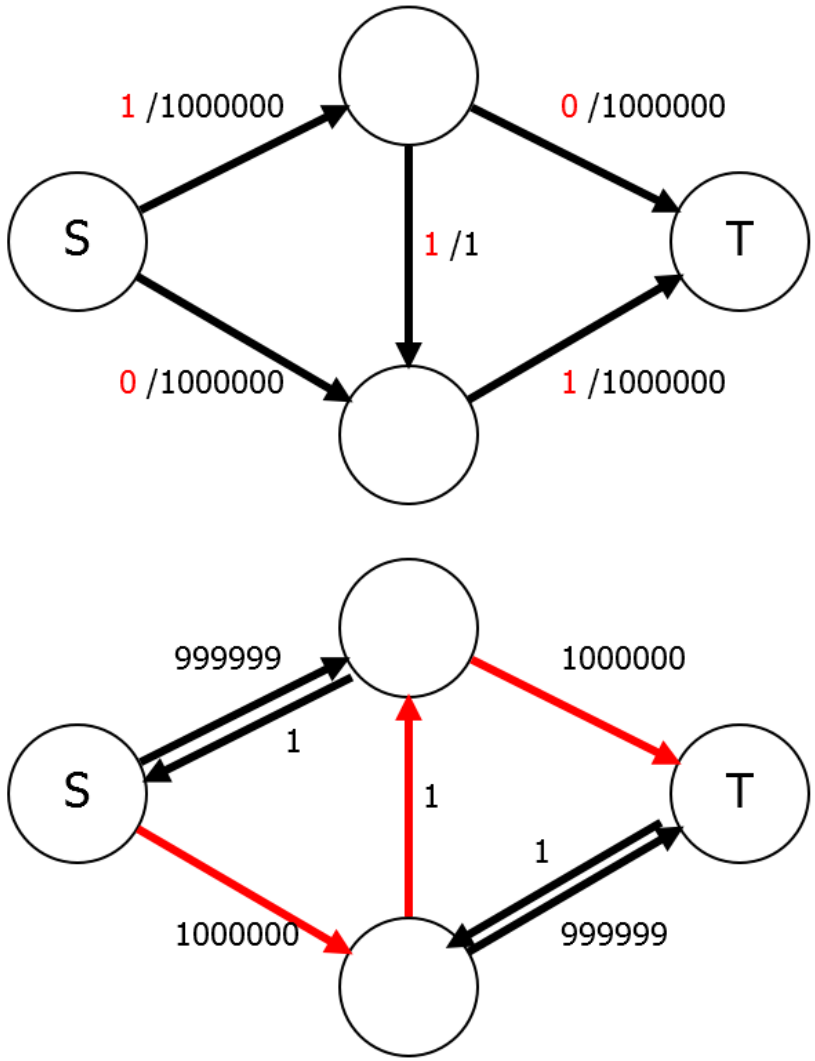
Every time if you are picking the wrong paths then you just got one unit of flow every iteration and took millions of iterations to actually finish.

Bad Example

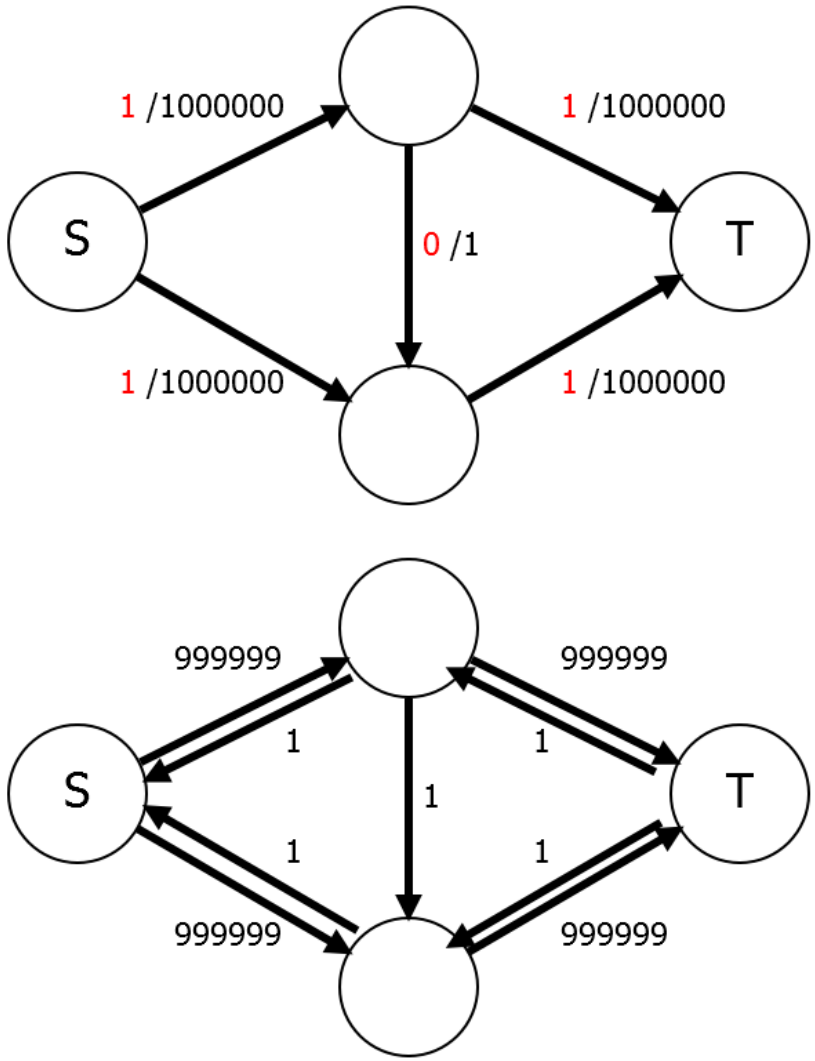

1 /1000000
0 /1000000
S
1 /1
T
0 /1000000
1 /1000000


999999
1000000
1
S
1
T
1
1000000
999999

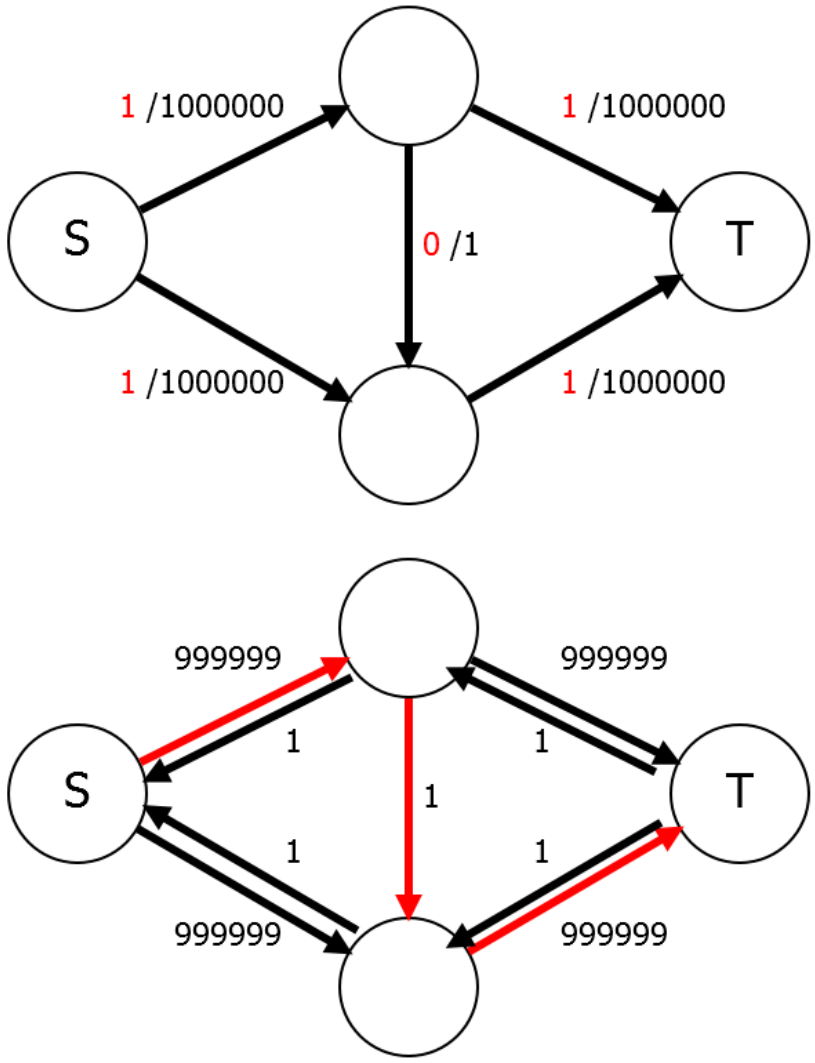
Bad Example


1 /1000000
0 /1000000
S
1 /1
T
0 /1000000
1 /1000000
999999
1000000
1
S
1
T
1
1000000
999999

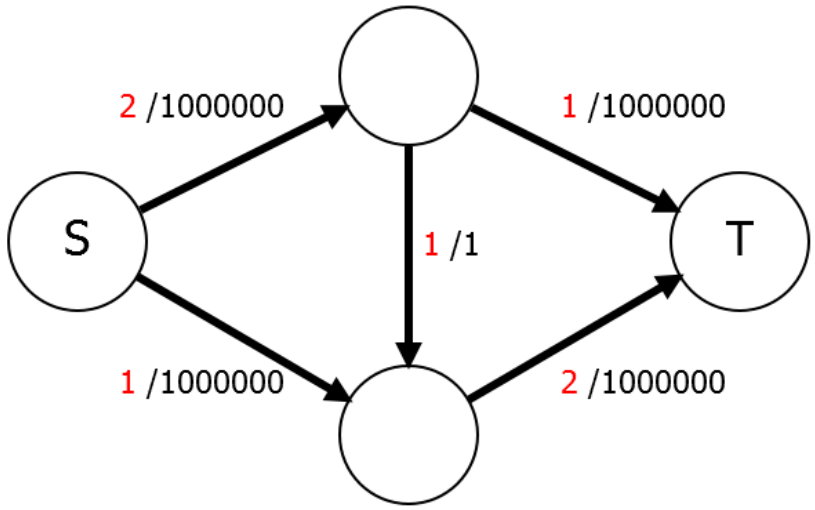
Bad Example

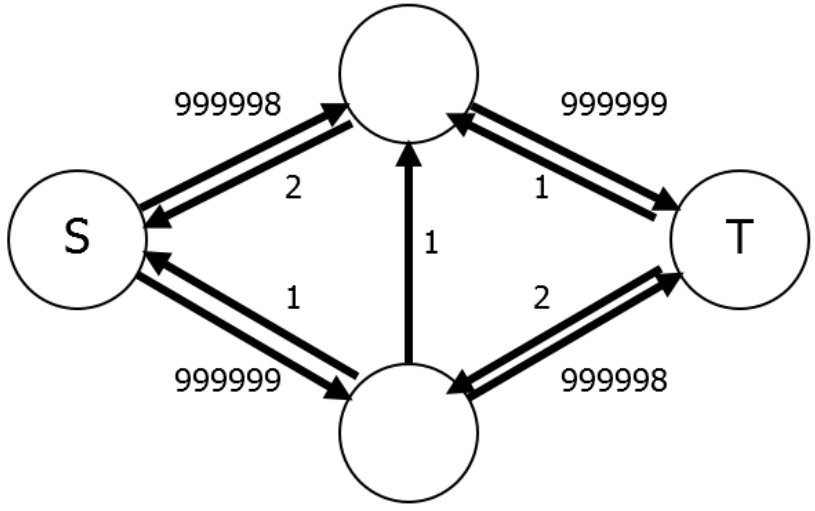

1 /1000000
1 /1000000
S
0 /1
T
1 /1000000
1 /1000000
999999
999999
1
1
S
1
T
1
1
999999
999999

Bad Example

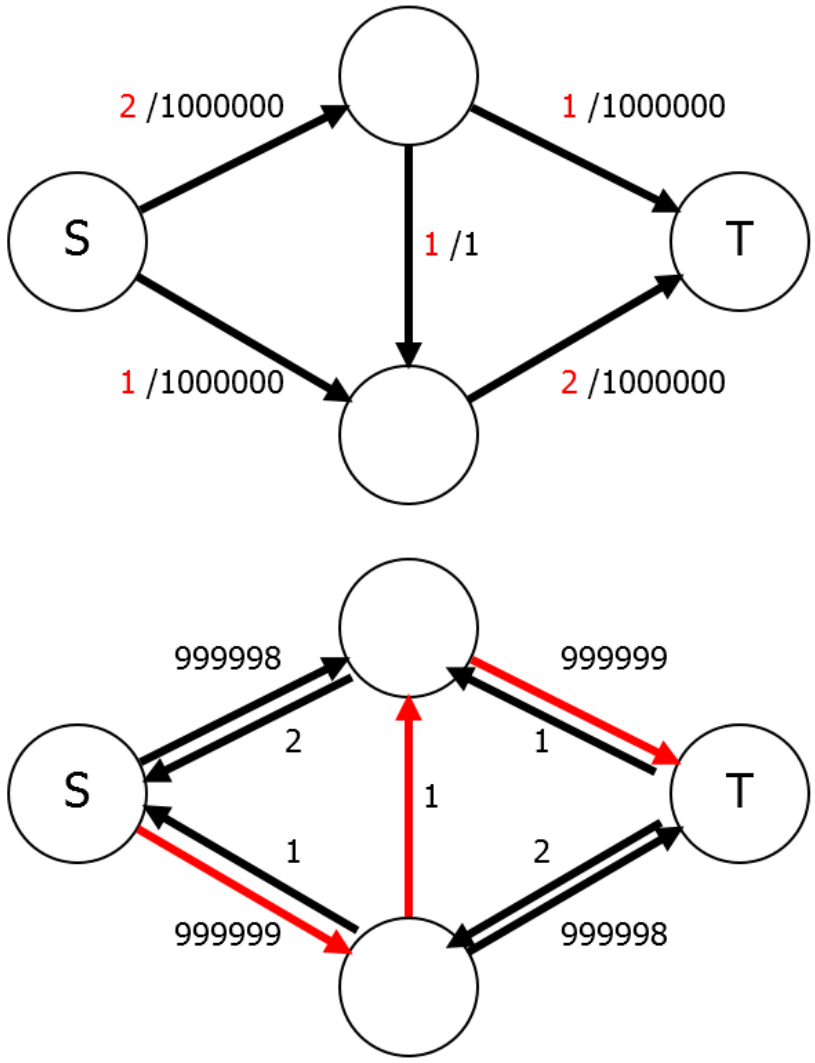

1 /1000000
1 /1000000
S
0 /1
T
1 /1000000
1 /1000000
999999
999999
1
1
S
1
T
1
1
999999
999999

Bad Example

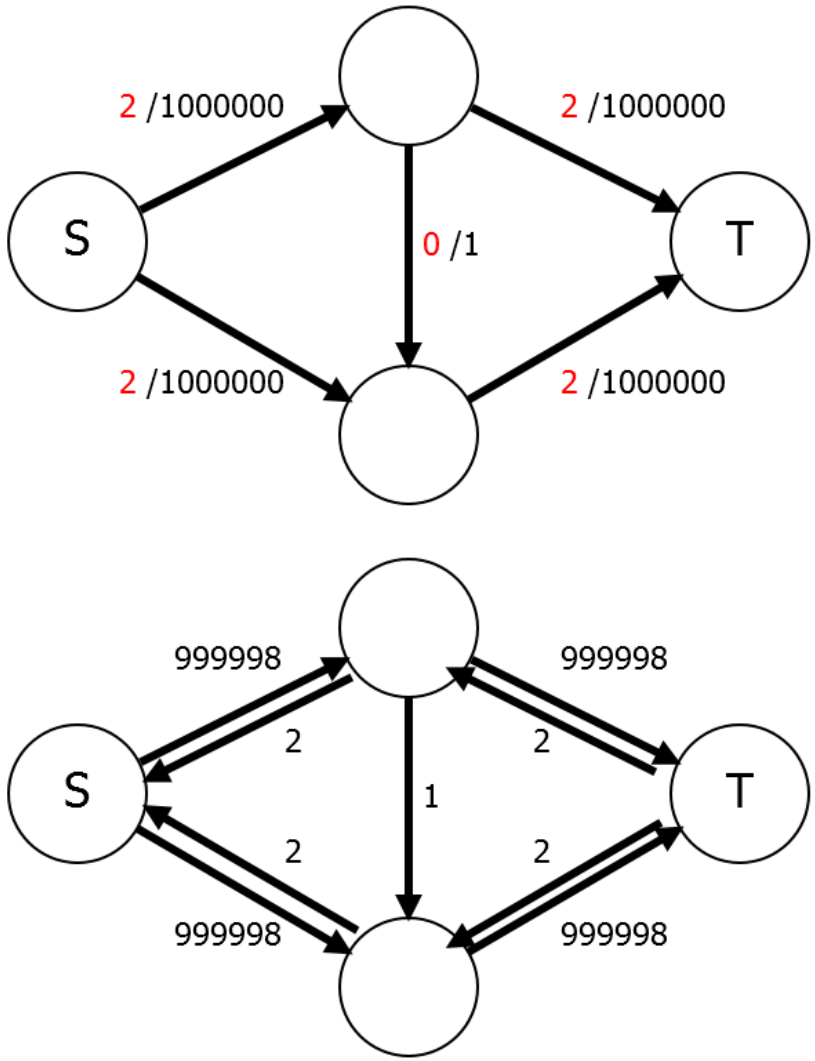

2 /1000000
1 /1000000
S
1 /1
T
1 /1000000
2 /1000000


999998
999999
2
1
S
1
T
1
2
999999
999998

Bad Example


2 /1000000
1 /1000000
S
1 /1
T
1 /1000000
2 /1000000
999998
999999
2
1
S
1
T
1
2
999999
999998

Bad Example


2 /1000000
2 /1000000
S
0 /1
T
2 /1000000
2 /1000000
999998
999998
2
2
S
1
T
2
2
999998
999998

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.

because the longer the path, the smaller the bottleneck value, which results in a longer runtime.

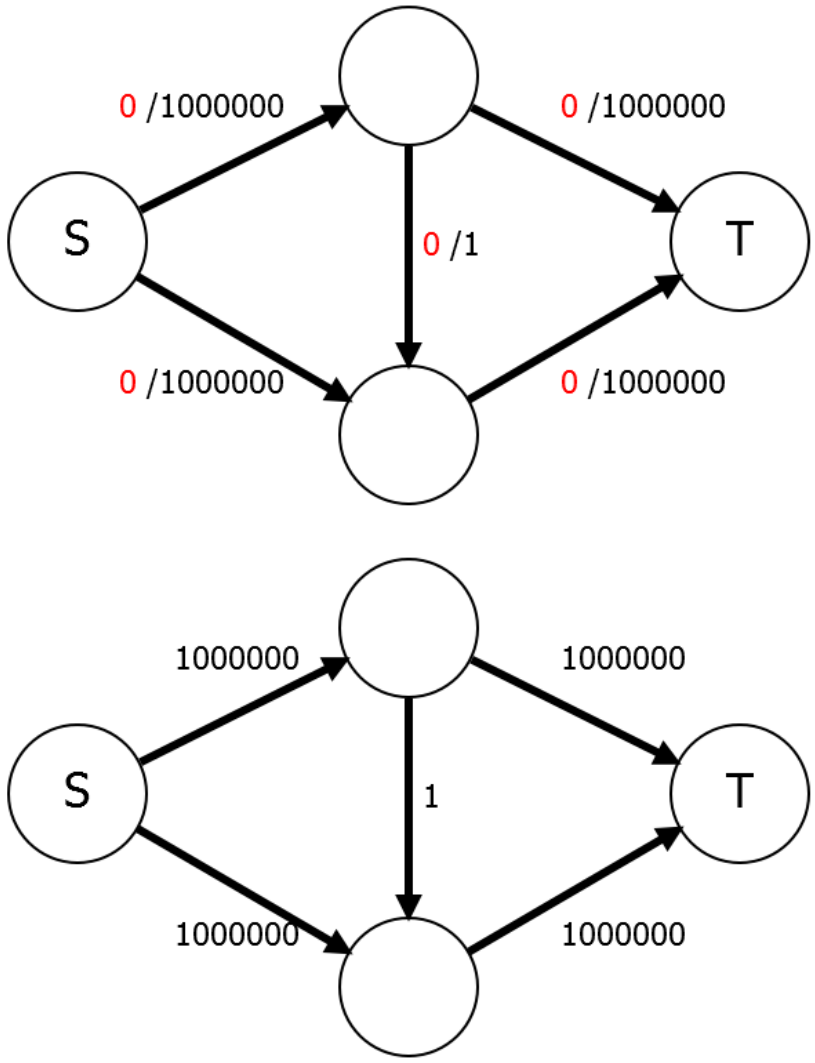
Edmonds-Karp Algorithm

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

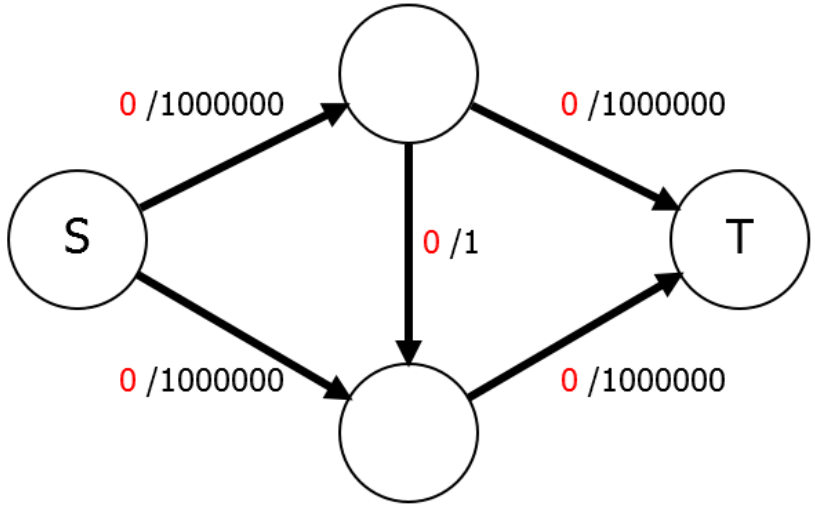
Find augmenting paths using BFS instead of DFS.

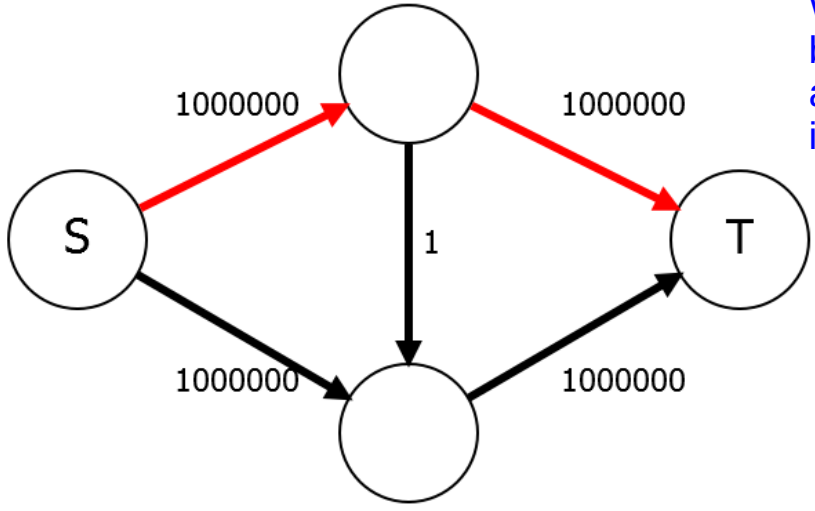
Edmonds-Karp is just implementation of Ford-Fulkerson method that uses BFS to find the augmenting path

Edmonds-Karp Execution



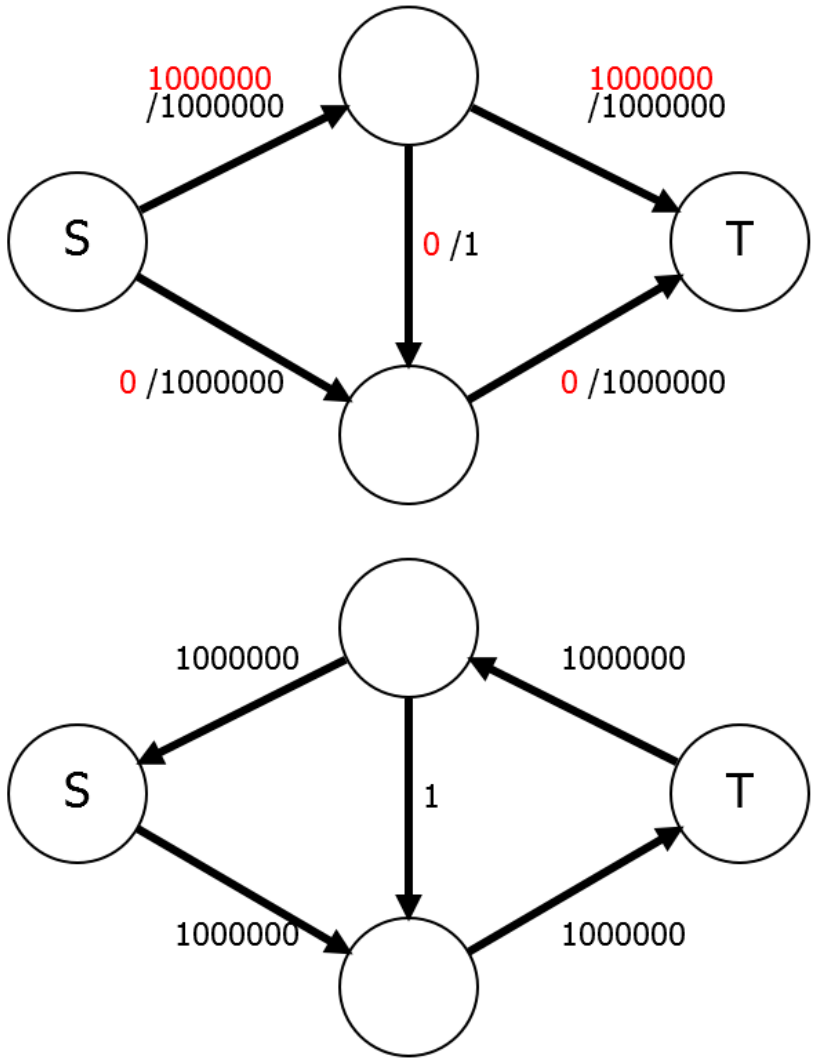
Edmonds-Karp Execution



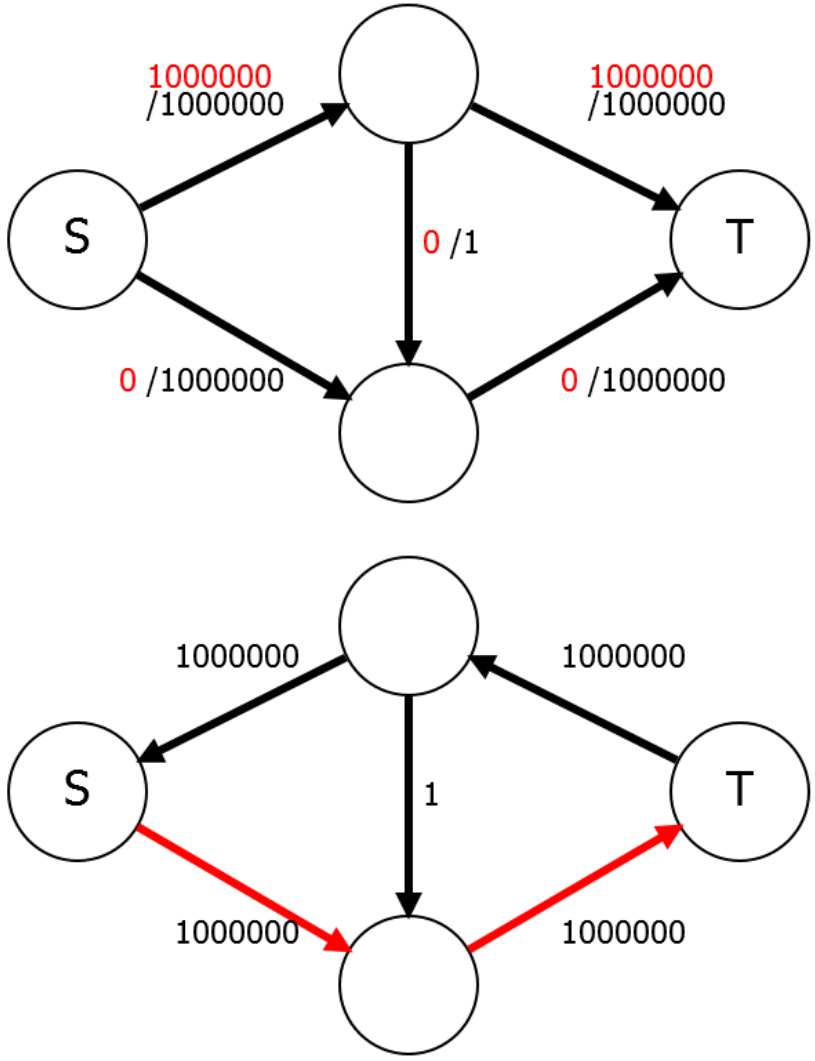


We can't use zig-zag path (as before), we're required to use the augmenting path with two edges instead (because it is the shortest)

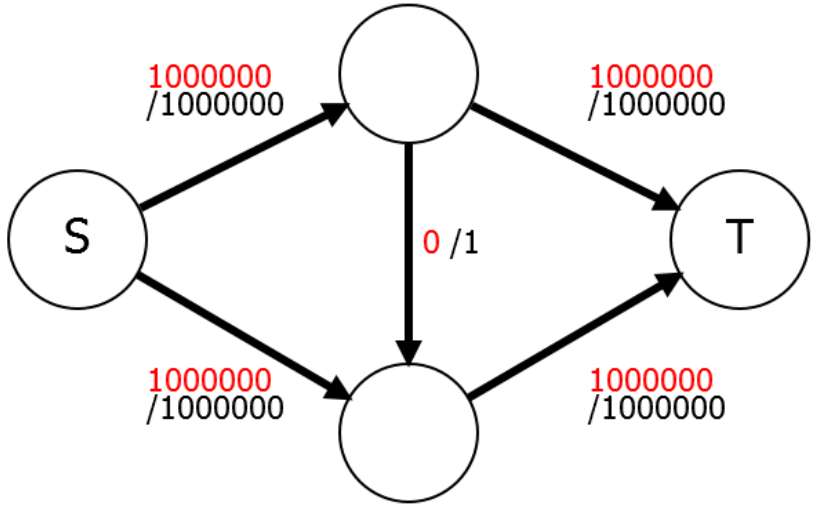
Edmonds-Karp Execution

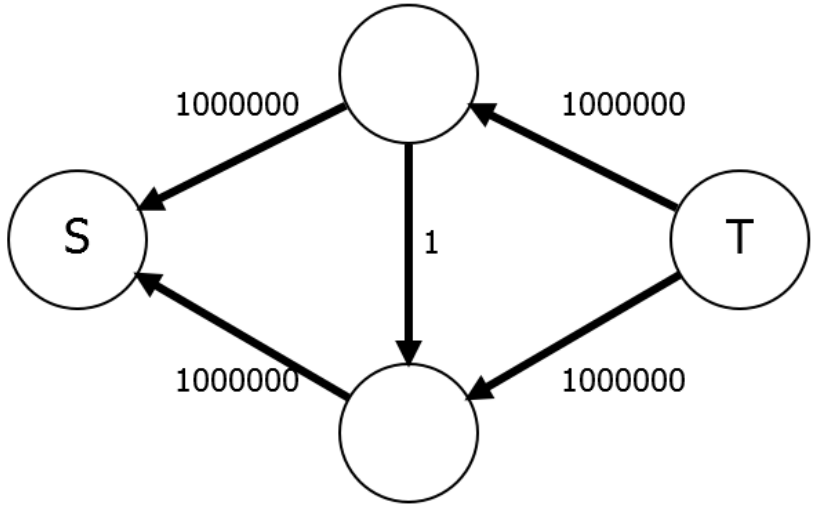


Edmonds-Karp Execution



Edmonds-Karp Execution



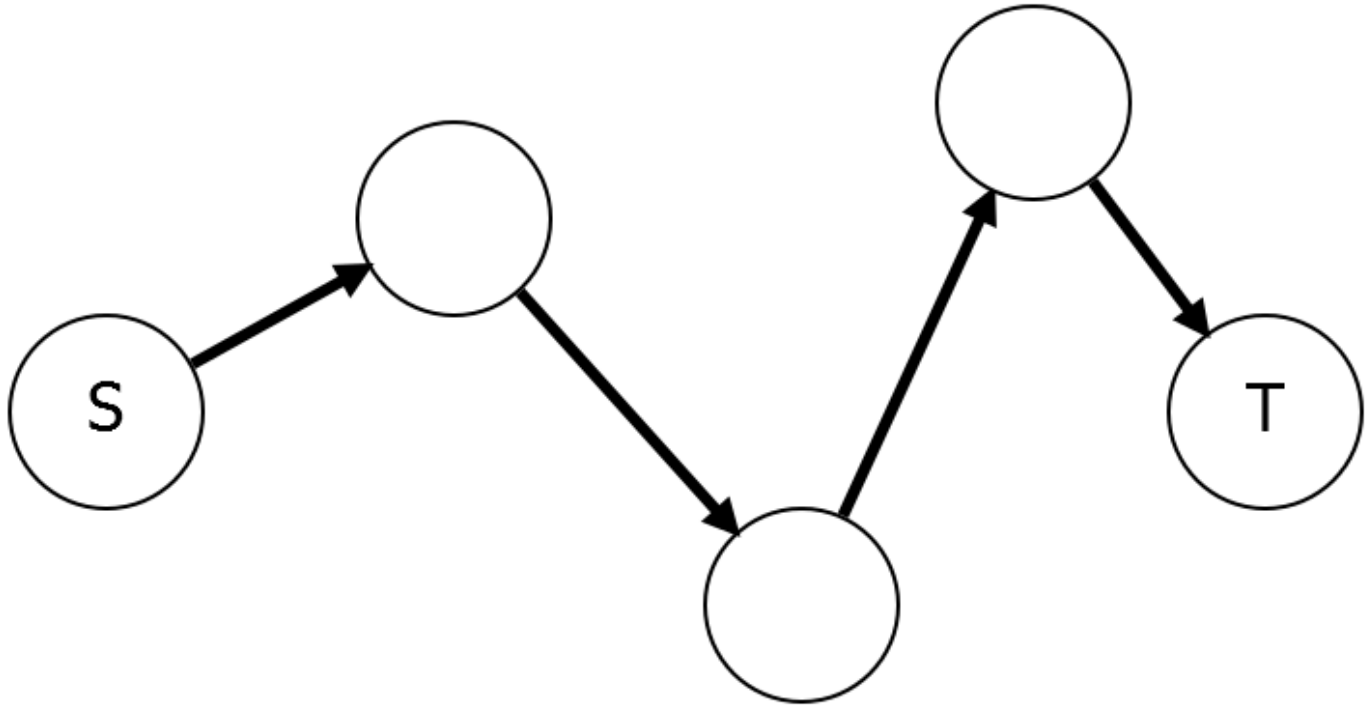


After we have selected first path there is another path with two edges and then its done we can't add anymore paths

Augmenting Paths

Need to analyze augmenting paths.

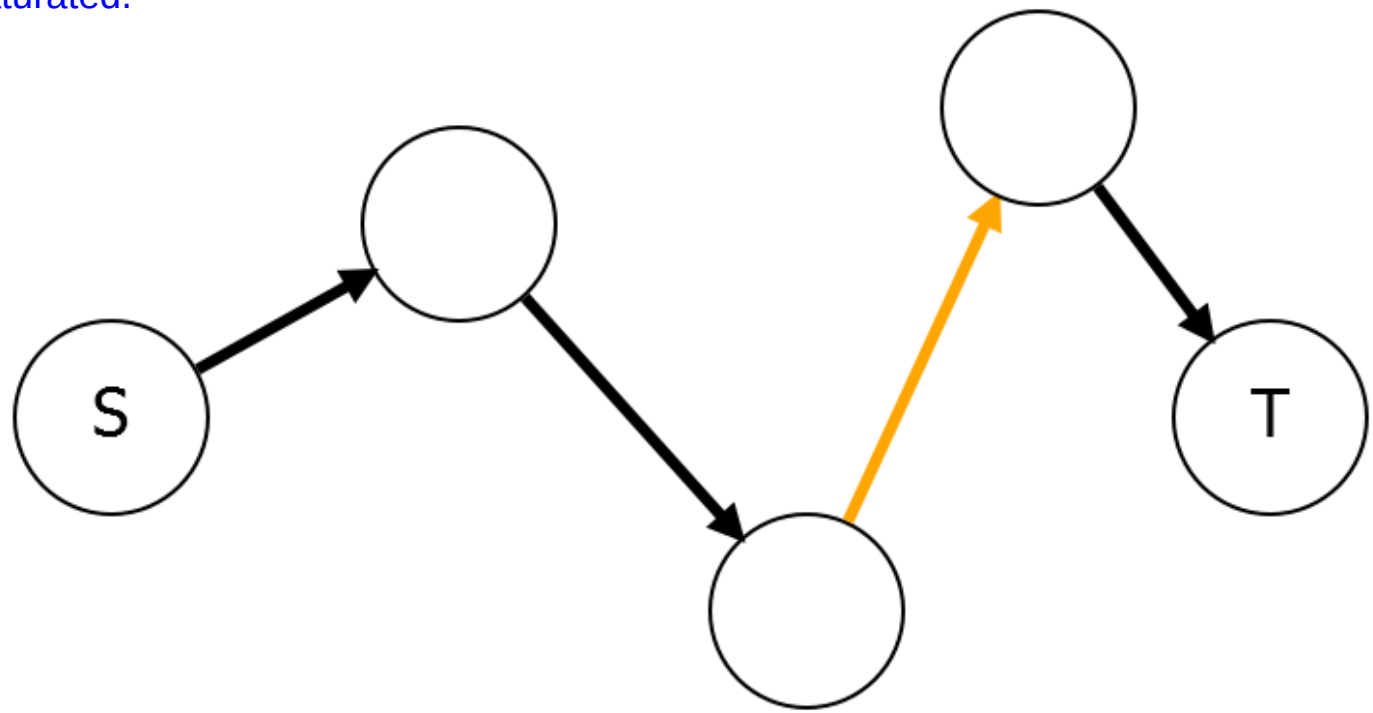
.



Augmenting Paths

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

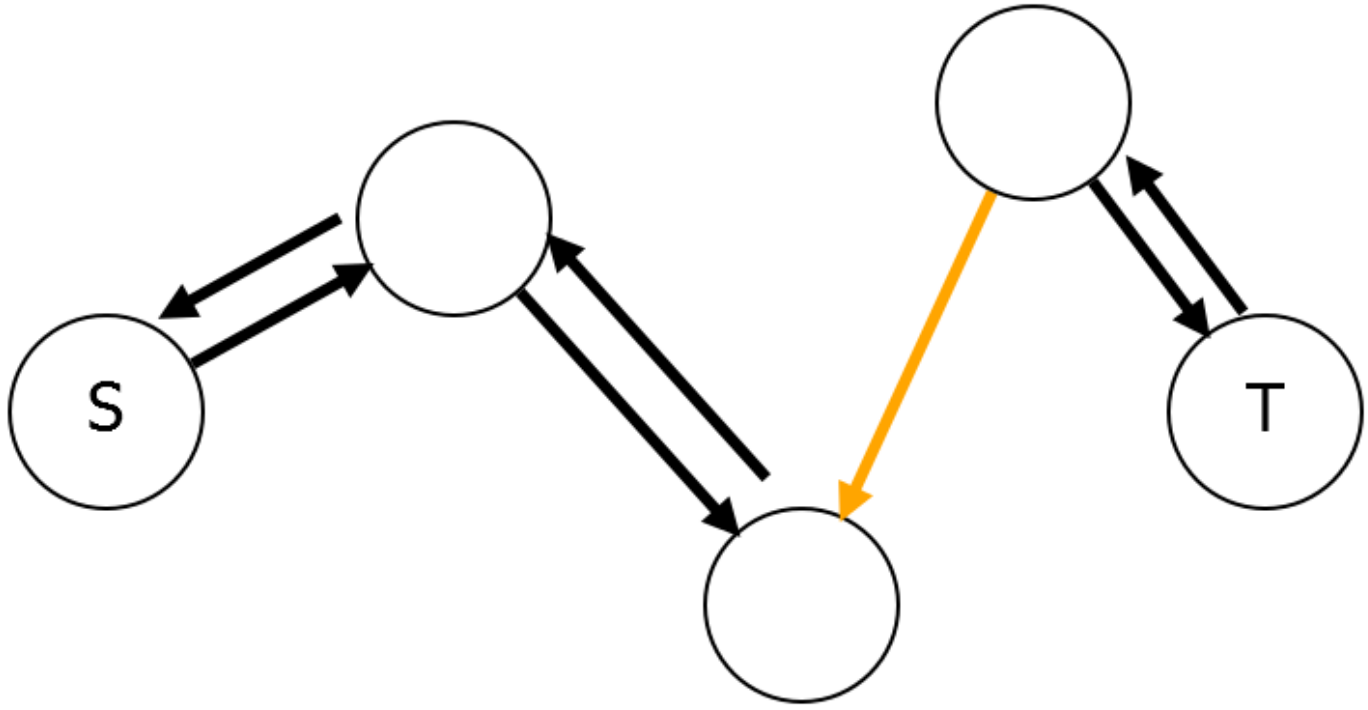
this is because the way we decided the amount of flow to run along this path was we took the minimum capacity of any of these edges in the residual graph and so whichever edge had only that much capacity left got saturated.



Augmenting Paths

Changes to residual network.

.



Edge getting repeatedly saturated because it has reached the capacity and we cannot add any more flow on it

Analysis Idea for Edmund-karp's Algo

- Show that no edge is saturated too many times. meaning there are no too many saturating augmenting paths
- Fails to hold in the bad case, where the middle edge is repeatedly saturated. this was the real thing that was limiting to us to adding one unit of flow per iteration

Increasing Distances

We will need this critical Lemma:

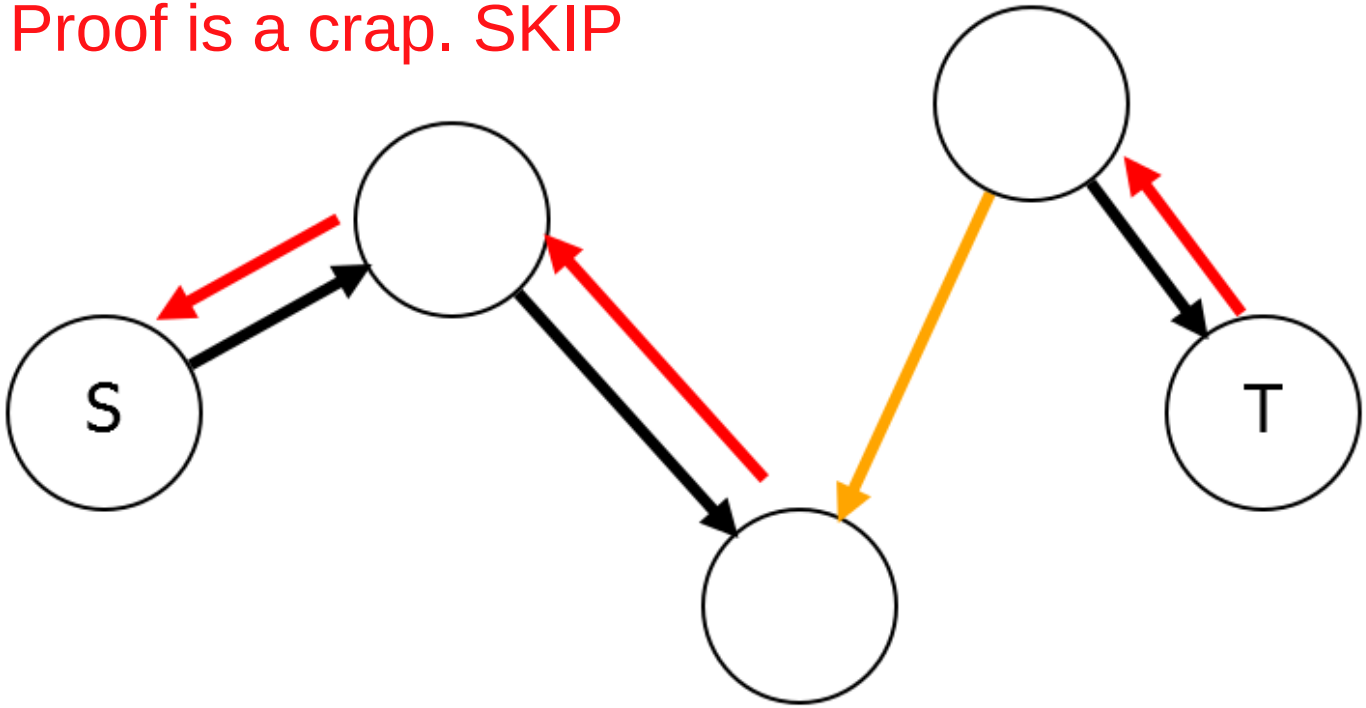
Lemma

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. augmenting path for Edmund-karp's by assumption is the shortest path from source to sink

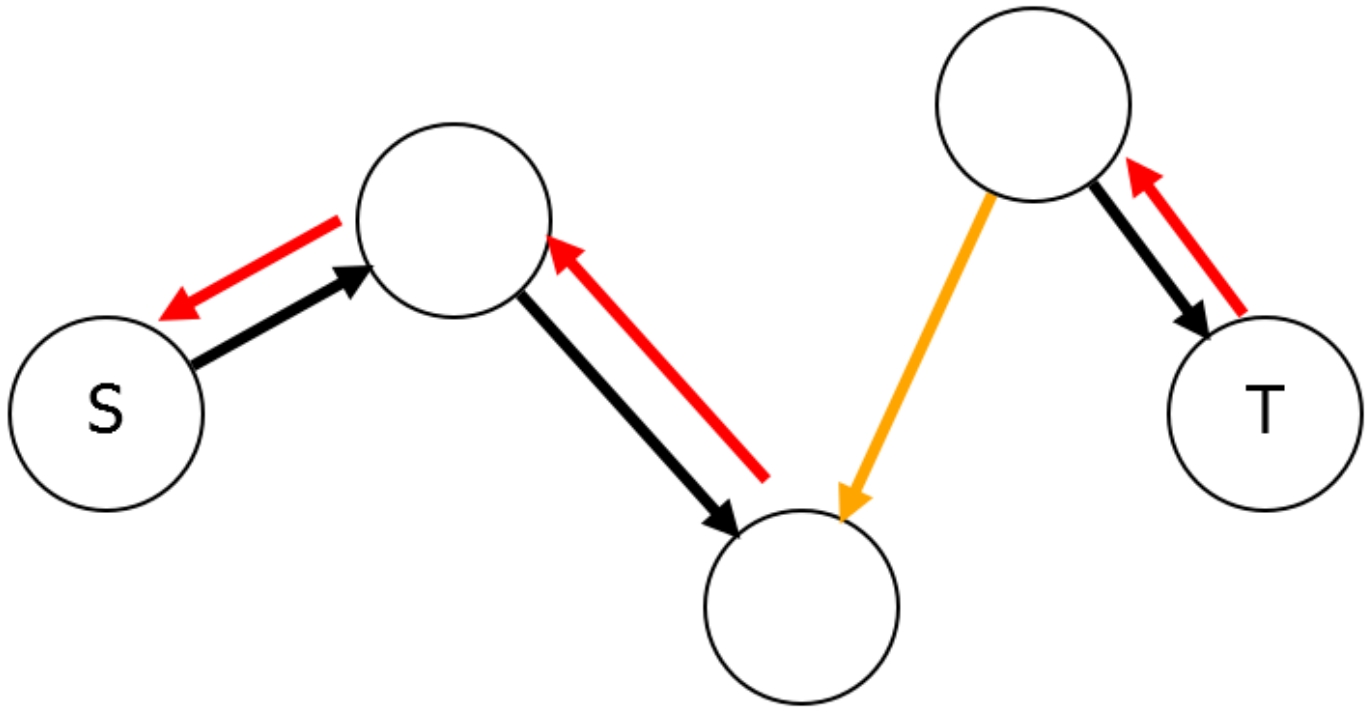
Proof is a crap. SKIP



Proof

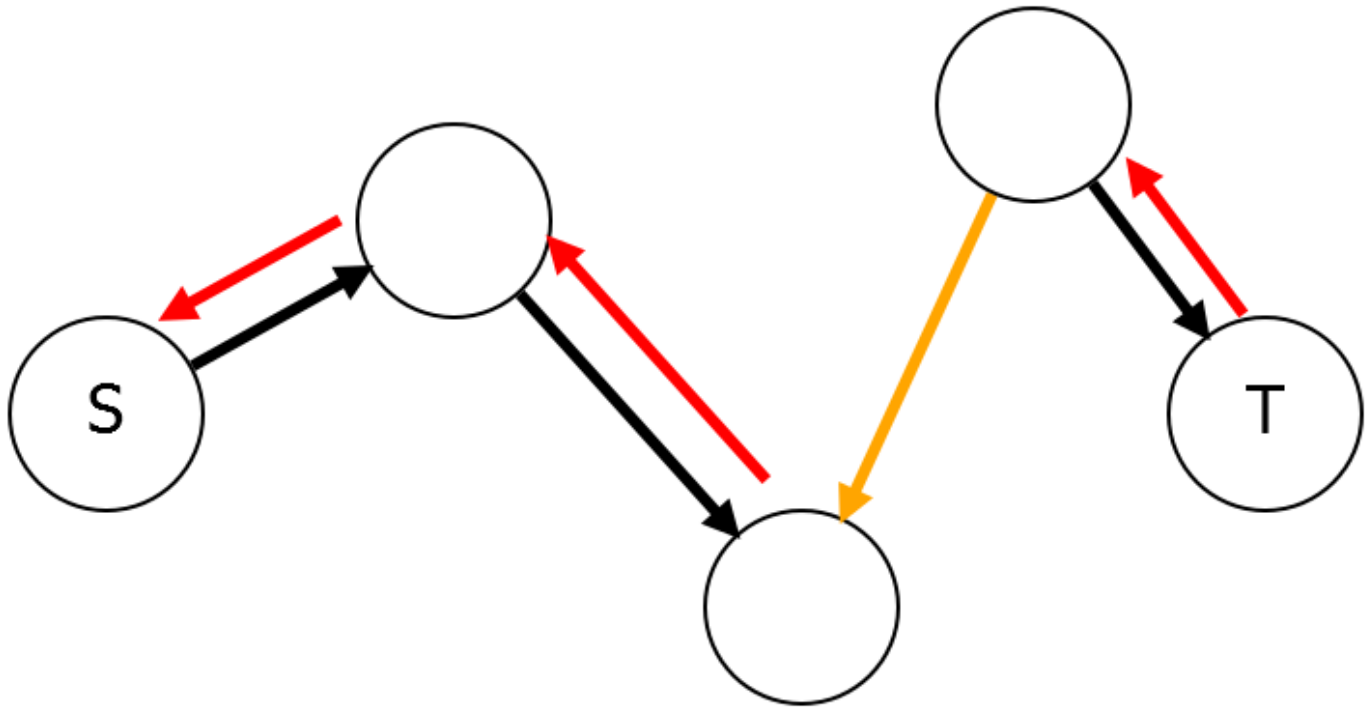
- Augmenting path is a shortest path.

.



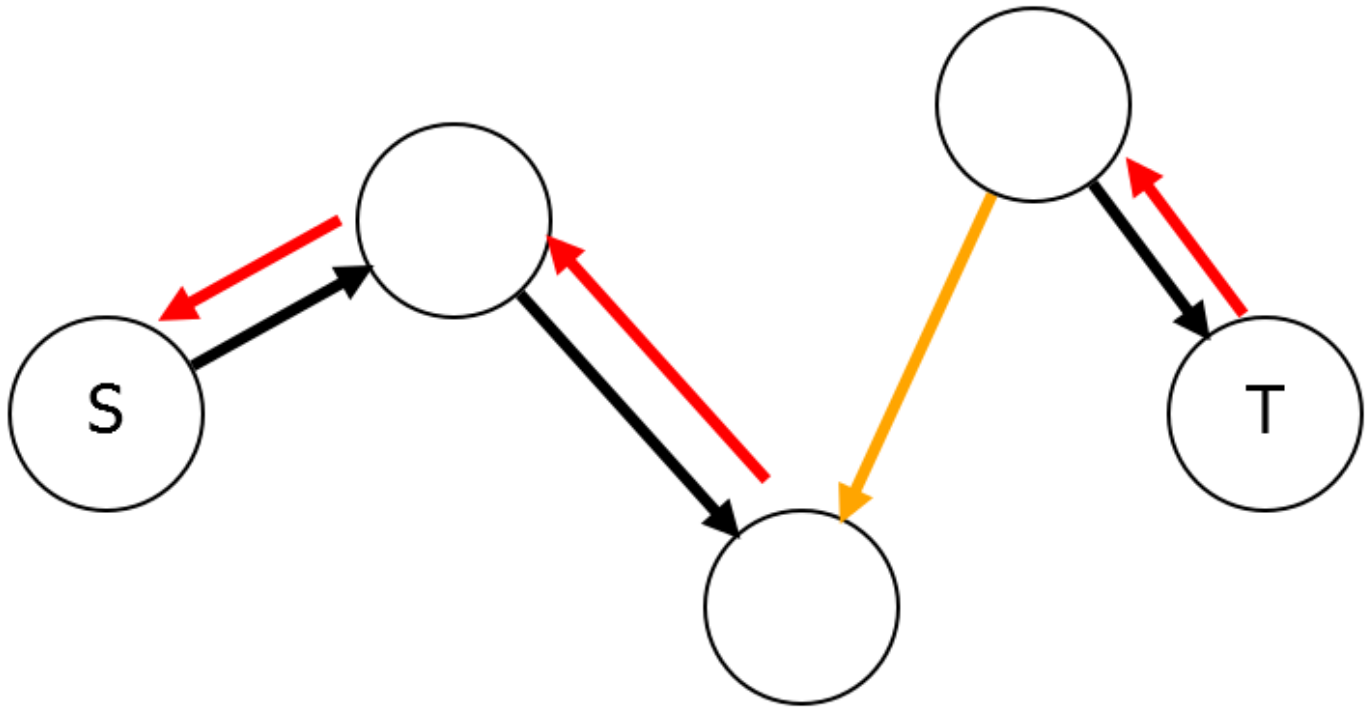
Proof

- 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.

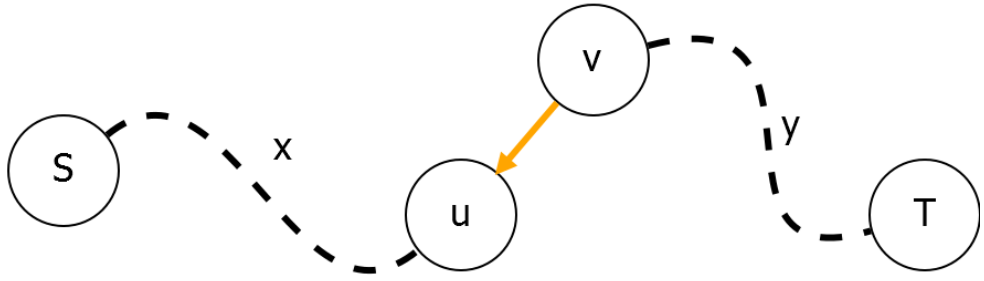
Proof

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

Applying Ford-Fulkerson, distance d(s,u) will be x + 1 hence d(s,v) will be >= x+1

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

This completes our proof as saturated edge u->v is used again in augmenting path only when total distance d(s,t) is increased from x+y+1 to x+y+3



Analysis

because different path has to be used if the distance has to increase and hence is limited by total no of vertices

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

No of vertices times number of edges squared

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.

Run time may not be depending explicitly on edge capacities like it did in case of Ford-Fulkerson Algorithm they can still affect the runtime. if all the capacities are zero, you don't need to do any augmenting paths.

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.