CMSC 451: Edge-Disjoint Paths

Slides By: Carl Kingsford

Department of Computer Science
University of Maryland, College Park

Based on Section 7.6 of *Algorithm Design* by Kleinberg & Tardos.

Edge-disjoint Paths

Suppose you want to send k large files from s to t but never have two files use the same network link (to avoid congestion on the links).

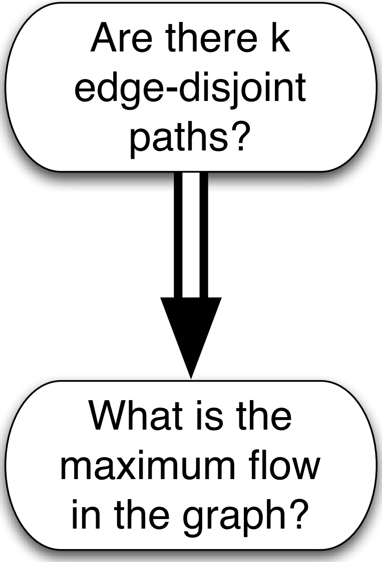
Leads naturally to the Edge-Disjoint Paths problem:

k Edge-disjoint Paths

Given directed graph G, and two nodes s and t, find k paths from s to t such that no two paths share an edge.

Again a Reduction!

- Given an instance of k-EDGE-DISJOINT PATHS,
- Create an instance of MAXIMUM NETWORK FLOW.
- The maximum flow will used to find the k edge disjoint paths.



Paths $\implies$ Flow

There is a nice correspondence between paths and flows in unit capacity networks.

Suppose we had k edge-disjoint $s - t$ paths.

We could sent 1 unit of flow along each path without violating the capacity constraints.

Lemma (Paths $\implies$ Flow)

If there are k edge-disjoint $s - t$ paths in directed, unit-weight graph G, then the maximum $s - t$ flow is $\geq k$.

Flow $\implies$ Paths

Theorem (Flow $\implies$ Paths)

If there is a flow of value k in a directed, unit-weight graph G, then there exist at least k edge-disjoint $s - t$ paths.

In other words: if we can find a flow of value k, then we know it's possible to "pack" at least k edge-disjoint paths into the graph.

If we can prove this, then we know how to check whether the k disjoint paths exist. The proof will also show how we can **find** the k disjoint paths.

Note: by our previous discussion, we can assume that flow f is a 0-1 flow: each edge contains either no flow, or 1 unit.

Flow $\implies$ Paths, 2

Theorem

If f is a 0-1 flow of value k, then the set of edges where $f(e) = 1$ contains set of k edge-disjoint paths.

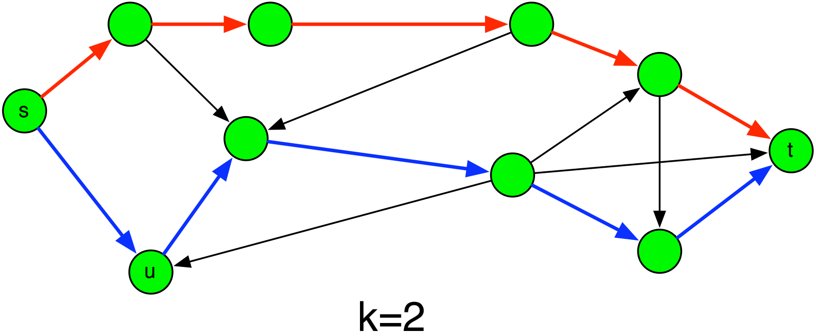
Proof: By induction on the number of edges with $f(e) = 1$.

IH: Assume the thm holds for flows with fewer edges used than f.

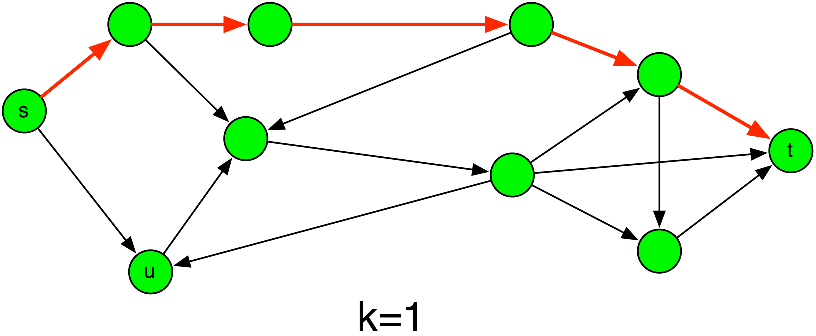
Let (s, u) be an edge that carries flow. Then by conservation we can find some edge leaving u that also has 1 unit of flow.

Repeating this, either (1) we reach t or (2) we loop around. We look at each of those cases on the next slides.

(1) Reach t:



(1) Reach t:



So,

We find an $s - t$ path, reduce the flow along it to 0, creating new flow f'.

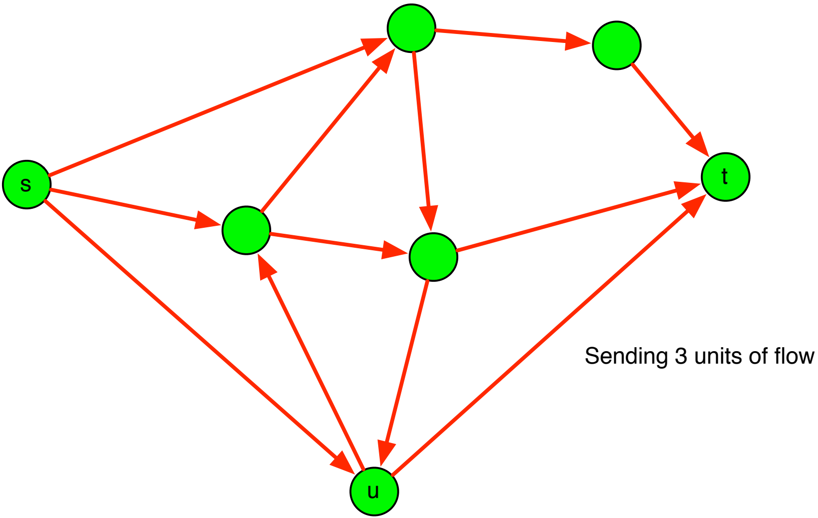
Value of new flow is $k - 1$.

And fewer edges have flow, so we apply our induction hypothesis: there are $k - 1$ edge-disjoint paths in flow f'.

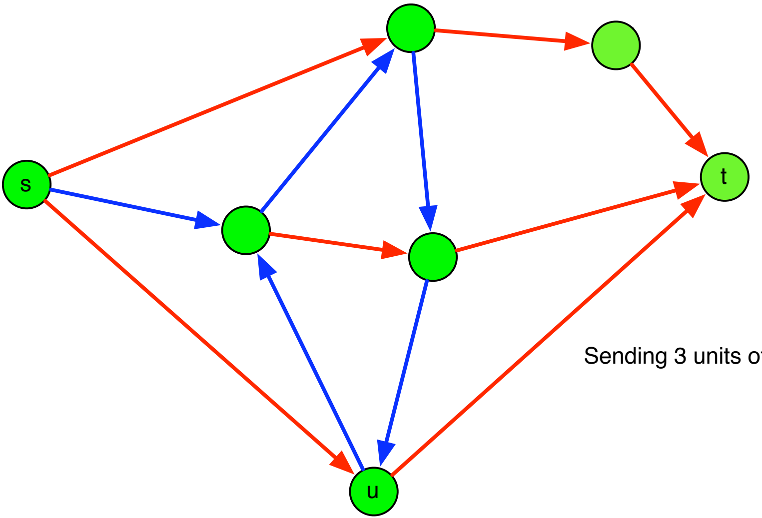
Hence, in this case, there are $1 + k - 1 = k$ edge-disjoint paths.

Suppose, instead we loop back to some node we've already visited:

(2) Create a cycle:

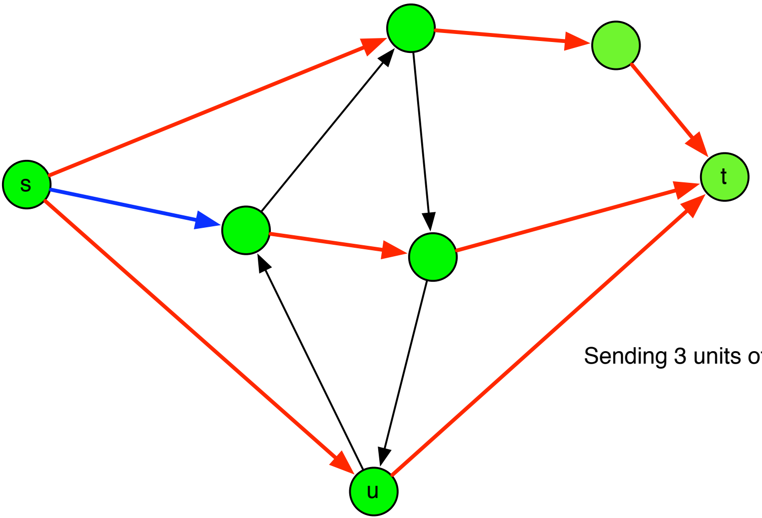


(2) Create a cycle:



Sending 3 units of flow

(2) Create a cycle:



Sending 3 units of flow

So,

We find a cycle, reduce the flow around it to 0, creating a new flow f'.

Value of new flow is still k.

BUT there are fewer edges that have flow: so we can still apply our induction hypothesis: there are k edge-disjoint paths in flow f'.

Hence, in either case, there are k edge-disjoint paths.

Base case: When $k = 1$ there is clearly 1 edge disjoint path.

Path Decomposition Algorithm

The proof gives us a way to actually **find** the paths:

1. Find the maximum flow in G.
2. Start walking from s.
3. If you create a cycle, eliminate the flow around the cycle.
4. If you reach t, output the path you used to reach t.

Summary

We can use a maximum flow algorithm to find k edge-disjoint, s-t paths in a graph.

Embedded within any flow of value k on a **unit-capacity** graph there are k edge-disjoint paths.

In other words, the value of the flow gives us the the number of edge disjoint paths.

Menger's Theorem

Theorem (Menger)

Given a directed graph G with nodes s, t the maximum number of edge-disjoint s-t paths equals the minimum number of edges whose removal separates s from t.

Useful: Suppose you are a hacker who wants to disrupt communications between the US and Russia. You know the network. How many edges must you knock out?