CS314 - Network Flow part 4

Note Title 4/19/2010

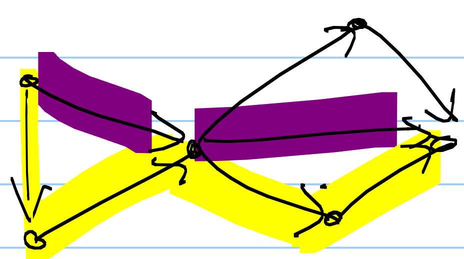
Announcements

- HW due
- Next HW will be posted today
  due next Friday

7.6: Disjoint Paths in Directed and Undirected graphs

Dfn: Two paths are edge-disjoint if they do not have any edges in common.

Goal: Find the maximum number of edge disjoint paths between 2 vertices s & t.



Given a directed graph G, & s & t how can we reduce this to a flow problem?

- Give every edge capacity <u>1</u>.

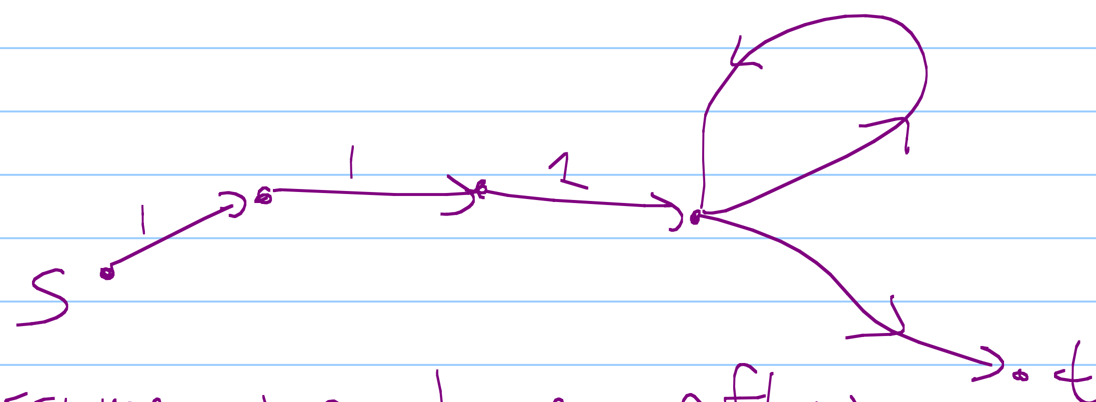
Find max flow in G'.

Return that value.

Claim: G has k edge-disjoint s-t paths
$\iff$ G' has a flow of value k.

pf: $\Rightarrow$: Given k s to t paths
send 1 unit of flow
along each.

$\Leftarrow$:

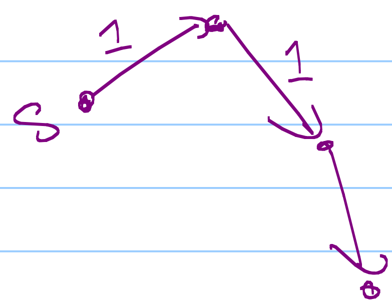


Assume we have a flow
in G' of value k.

pf cont: Induction on # edges that carry flow.

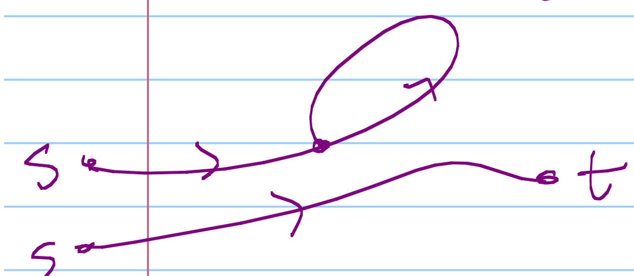
Base case: If $v = 0$, done.

Inductive Step: If $v \geq 1$, we can find an edge out of s that carries 1 unit of flow.



Trace a set of edges which all carry one unit of flow (since flow is conserved)

Either we repeat a vertex or we hit t.



If we hit t, have a path. Make this one of s-to-t paths & reset flow on those edges to 0.

If we repeat a vertex, consider the set of edges between those 2 vertices which have flow = 1.

Create a new flow f' which is the same as f, except all edges on cycle are now $f'(e) = 0$.

<u>Claim:</u> f' is a valid flow with the same value & fewer edges (so IH covers it).

① capacity constraint: now carry 0 flow (still ≤ 1)

② conservation constraint: for any vertex on this cycle, reduce f^{in} & f^{out} by 1.

& $v(f') = v(f)$ since we haven't changed any edge out of s. ∎

Runtime:

$O(mC) = O(mn)$

$O(mn^2)$

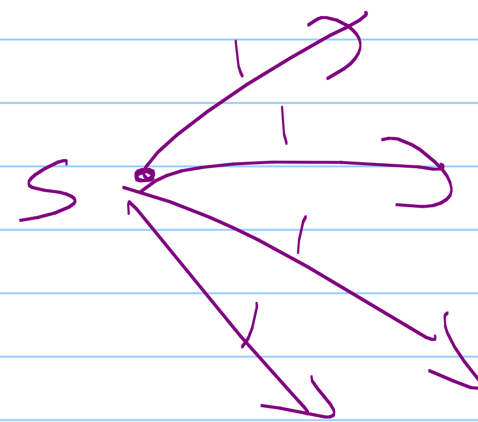
$O(n^3)$

$O(m^2 \log_2 C)$

flow runtimes

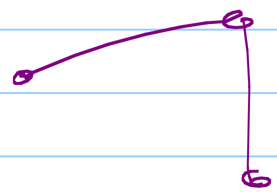
Conversion: $O(m)$

total: $O(mn)$



$C \leq n$

What about undirected graphs?



Convert to directed graph by drawing 2 edges (1 each way) for every $e \in E$.

A few extensions (7.7)

What if we have multiple sources & sinks?

Let d_v = demand at node v.

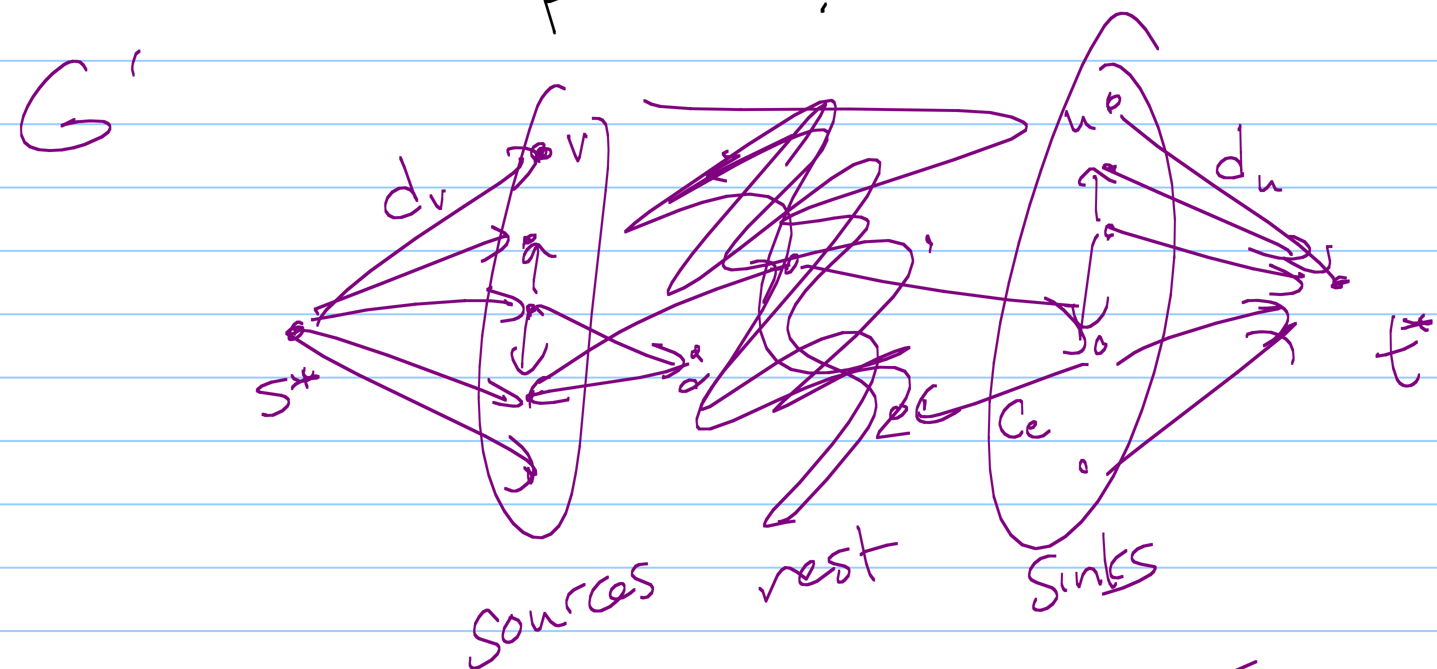
- if $d_v < 0$, it is a source
- if $d_v > 0$, it is a sink

replaces conservation constraint

Goal: For every v, $d_v = f^{in}(v) - f^{out}(v)$

(Call this a circulation.)

How can we reduce to a normal flow problem? add s^* & t^*



Need flow in G' of value $\sum_{\text{sources } v} d_v$

Circulation in G of value k

$\iff$ flow in G' of value k

Adding lower bounds

Suppose we want flow on each edge to meet a certain lower bound, also.

So want flow with:

← was 0

① $\forall e \in E,\ \ell_e \le f(e) \le c_e$

② $f^{in}(v) - f^{out}(v) = d_v$

same as with circulations

"proto-flow"

design f_0

To start — put ℓ_e units of flow on each edge, so $f_0(e) = \ell_e$

Satisfies (1) but not (2)

Have: $f_0^{in}(v) - f_0^{out}(v) = \sum_{e \text{ into } v} \ell_e - \sum_{e \text{ out of } v} \ell_e$

Call this L_v.

If $L_v = d_v$, done.

If not – need to get $(d_v - L_v)$ more into v!
(Note – no lower bound!)

So create G' with same V & E:

- Each edge now has capacity $c_e - \ell_e$
- Each node has demand $d_v - L_v$.

This is a circulation!

Need circulation in G (with lowerbounds met)

$\Longleftrightarrow$ circulation in G'

(see book)