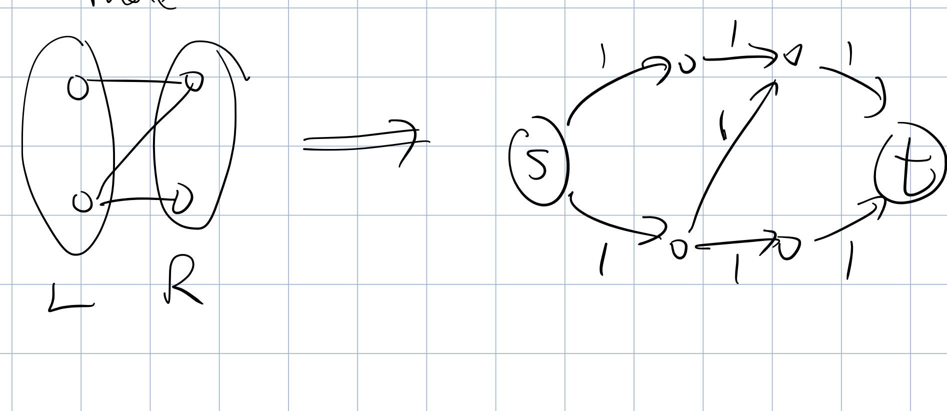
4 Oct 2021

1. Hopcroft – Karp Matching Algorithm
2. Some Applications of Max Flow

Announcements

1. Problem Set 2 is out. Groups now formed on CMS.
2. Some mistakes in the problem set were fixed at 6 pm yesterday. Reload to get latest version.
   Problem 1: design a deterministic 18-competitive algorithm.
3. Deadline is Monday, 10/18.

Hopcroft – Karp Alg: Dinitz specialized to the flow network you get by reducing bipartite perf matching to max flow.



All edges in this network have capacity 1.

$\Rightarrow$ every residual graph G_f in Dinitz's alg has edges of residual capacity 1.

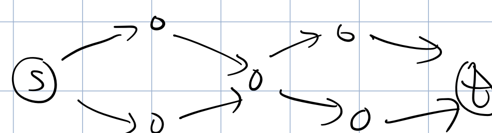
Work done in one blocking flow computation

(i) augmenting a path of length L
Takes $O(L)$ time, but deletes L edges from H. (Graph of advancing edges.)

Total work $O(m)$.

(ii) deleting vertices and edges
Total work $O(m+n) = O(m)$.

(iii) pushing vertices onto the stack.



every time a vtx is pushed onto the stack it takes $O(1)$ time. Charge that cost to the operation which pops the vtx off the stack.

TIME SPENT ON (iii) = O(TIME SPENT ON i + ii).

In all, computing a blocking flow takes $O(m)$.

Hopcroft & Karp's amazing insight: you only need to compute $\leq 2\sqrt{n}$ blocking flows!

$\Longrightarrow$ Computes max matching in $O(m\sqrt{n})$.

Break execution into 2 phases.

Phase 1: $d(t) \leq \sqrt{n}+1$

shortest augmenting path length.

Then there can be at most $\sqrt{n}$ blocking flow iterations in Phase 1 because each increases $d(t)$.

Phase 2. $d(t) > \sqrt{n}+1$

If M^* is the max matching and M is the matching at start of Phase 2, and k denotes $|M^*|-|M|$

then $M^* \oplus M$ contains at least k M-augmenting paths. These are vertex-disjoint and each contains at least $d(t)-1$ vertices.

$$k \cdot [d(t)-1] \leq n$$

$$k \leq \frac{n}{d(t)-1} \leq \sqrt{n}$$

Each round of blocking flow computation in Phase 2 grows M by at least 1 edge.
But M only has room to grow by $k \le \sqrt{n}$ edges.
So Phase 2 has $\le \sqrt{n}$ blocking flow computations.

Applications of Maximum Flow
(Kleinberg & Tardos Ch. 7)

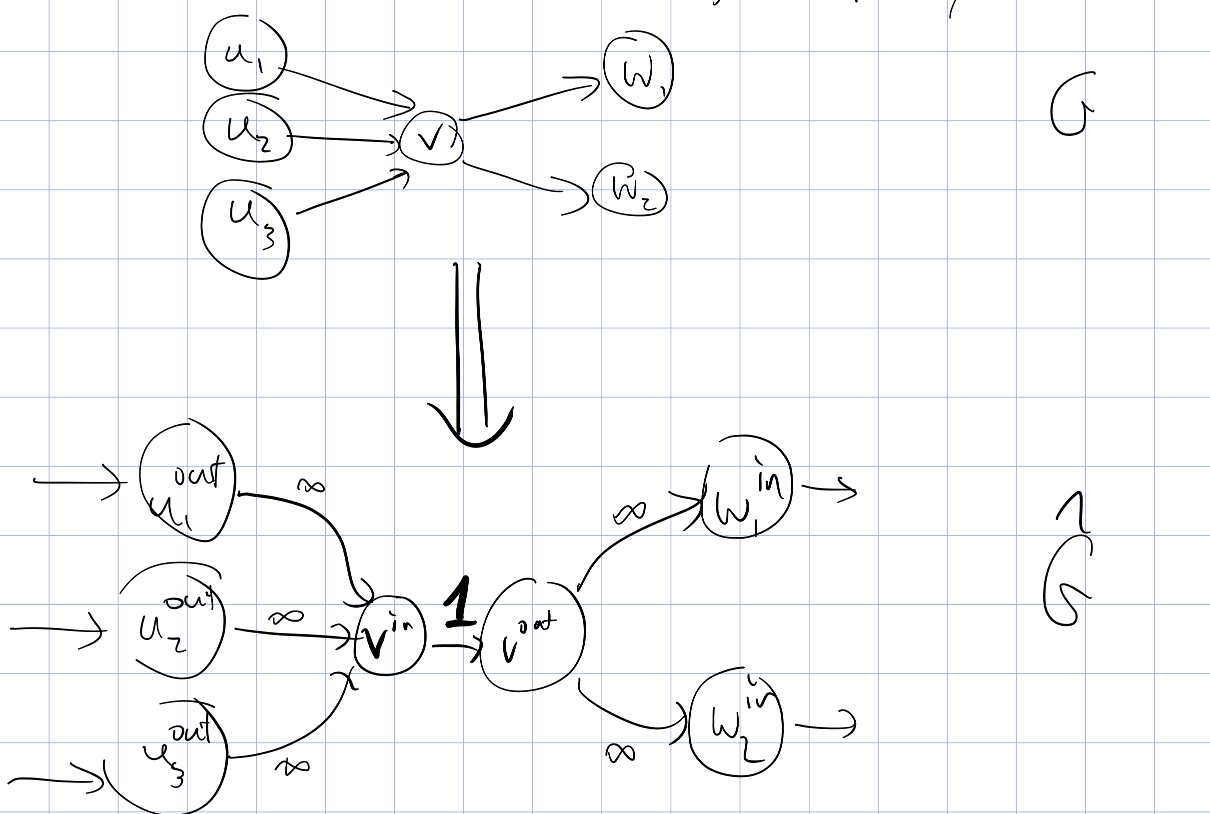
Vertex disjoint paths in directed graphs.

<u>Menger's Theorem</u>. If $G=(V,E)$ is a dir graph and $s,t \in V$ then

$$\begin{bmatrix} \text{max \# of} \\ \text{internally} \\ \text{vertex-disjt} \\ \text{paths from} \\ s \text{ to } t \end{bmatrix} = \begin{bmatrix} \text{min \# of} \\ \text{vertices in} \\ V \setminus \{s,t\} \text{ whose} \\ \text{deletion} \\ \text{disconnects} \\ s \text{ from } t \end{bmatrix}$$

Def. Two s-t paths are internally vtx-disjoint if they have no vertices in common other than their endpoints, s and t.

A gadget to transform vertex-disjointness constraints into edge-capacity constraints.



A path from s to t in G transform to a path from s^{out} to t^{in} in $\hat{G}$.

$s, \quad u, \quad v, \quad t$

$$\Downarrow$$

$$s^{out} \to u^{in} \to u^{out} \to v^{in} \to v^{out} \to t^{in}.$$

Internally vertex-disjoint paths in G transform to paths that don't re-use any of the "gadget edges" with capacity 1.

Integer flows from s^{out} to t^{in} transform back to internally vtx-disjoint paths.

LHS of Menger = max flow in $\hat{G}$

RHS of Menger = min cut in $\hat{G}$.