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 6pm yesterday. Reload to get latest version.
3. Deadline is Monday, 10/18.

Hopcroft-Karp Alg: Design a deterministic 1.8-competitive algorithm.

Deadline is Monday, 10/18.

All edges in this network have capacity 1.
⇒ every residual graph G in Diniz'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

\[
\begin{align*}
\circ & \rightarrow \circ & \rightarrow \circ & \rightarrow \circ & \rightarrow \circ \\
\circ & \rightarrow \circ & \rightarrow \circ & \rightarrow \circ & \rightarrow \circ
\end{align*}
\]

- Every time a vertex is pushed onto the stack it takes \( O(1) \) time.
- Charge that cost to the operation which pops the vertex off the stack.

\[
\text{Time spent on (iii)} = O(\text{Time spent on it+i}i).
\]

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

Break execution into 2 phases.

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

Then there can be at most $\sqrt{n}$ shortest augmenting path lengths.

**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.

$$k \cdot \left\lfloor d(t) - 1 \right\rfloor \leq n$$

$$k \leq \frac{n}{d(t) - 1} \leq \sqrt{n}$$
Each round of blocking flow computation in Phase 2 grows $M$ by at least 1 edge. But $M$ only has room to grow by $k \leq \sqrt{n}$ edges, so Phase 2 has $\leq \sqrt{n}$ blocking flow computations.

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

Vertex disjoint paths in directed graphs,

Menger's Theorem. If $G = (V,E)$ is a digraph and $s,t \in V$ then

$$\left\lceil \max \text{ # of internally vertex-distinct paths from } s \text{ to } t \right\rceil = \left\lceil \min \text{ # of vertices in } V \setminus \{s,t\} \text{ whose deletion disconnects } s \text{ from } t \right\rceil$$
Def. Two s-t paths are internally \( \text{vtx-disjoint} \) 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 \( G \) transforms to a path from \( s_{\text{out}} \) to \( t_{\text{in}} \) in \( G' \).
\[ s, u, v, t \]
\[ \Downarrow \]
\[ s \rightarrow u \rightarrow u^{\text{out}} \rightarrow v \rightarrow v^{\text{out}} \rightarrow t^{\text{in}}. \]

Internally

\[ \wedge \text{ vertex-disjoint paths in } G \]

\[ \text{transform to paths that don't re-use any of} \]

the "gadget edges" with capacity 1.

Integer flows from \( s^{\text{out}} \rightarrow \) \( t^{\text{in}} \)

\[ \text{transform back to internally} \]

\[ \wedge \text{ vertex-disjoint paths}. \]

\[ \text{LHS of Menger } = \max \text{ flow in } G \]

\[ \text{RHS of Menger } = \min \text{ cut in } G. \]