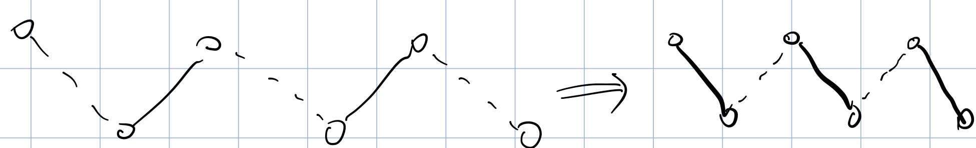
30 Aug 2021

Max bipartite matching
Min-cost bipartite perfect matching

RECAP. For G bipartite and $M \subset E(G)$ matching

- a vertex is <u>free</u> if it doesn't belong to an edge of M, else it is <u>matched</u>.
- an <u>M-alternating path</u> (or cycle) is one whose edges alternate between belonging, not belonging to M.
- an <u>M-augmenting path</u> is an M-alternating path from a free vertex to another free vertex.



M $\qquad\qquad$ $M \oplus P$

Algorithm (Partly Specified) for Max Matching

$M \leftarrow \emptyset$
while $\boxed{\exists \text{ M-augmenting path } P}$
$\quad M \leftarrow M \oplus P$
endwhile
output M

how to search efficiently?

(§2) For general graphs (incl. non bipartite) can find P in $O(mn)$ time.

Last time we saw this takes $\leq \frac{n}{2}$ iterations (n = # vertices, m = # edges)
and we saw it correctly outputs a max matching.

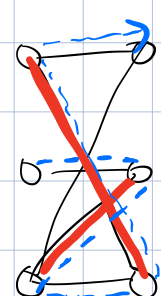
For bipartite G, we can find M-augmenting P if it exists, in linear time.

<u>Def.</u> If $G = (L \cup R, E)$ is bipartite and M is a matching, the <u>residual graph</u> G_M has vertices $L \cup R$ and <u>directed</u> edges

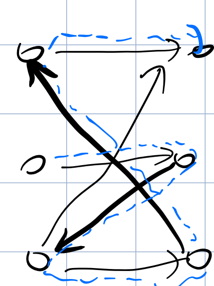
$$\{(u,v) \mid u \in L,\ v \in R,\ (u,v) \notin M\}$$

$$\cup \{(v,u) \mid u \in L,\ v \in R,\ (u,v) \in M\}$$

E.g.



G_M



$\{M\text{-augmenting paths in } G\}$ $\quad F \triangleq \{\text{free vertices}\}$

$\qquad |||$

$\{\text{directed paths in } G_M \text{ from } L \cap F \text{ to } R \cap F\}$

BFS finds a dir. path in G_M from $L \cap F$ to $R \cap F$ in time $O(m+n)$ if one exists.

$O(n)$ loop iterations, $O(m+n)$ time per iteration

$\Longrightarrow$ $O(mn + n^2)$ time altogether.

Hopcroft & Karp improved this to $O(m\sqrt{n})$.
(§1.4 of notes, to be presented in ~2 weeks)

Minimum Cost Bipartite Perf. Matching.

Input: $G = (L \cup R, E)$ bipartite
$c: E \longrightarrow \mathbb{R}$

Convention: $c(u,v) = \infty$ if $(u,v) \notin E$.

If G^+ denotes the complete bipartite graph on vertex sets L, R and M is a matching in G^+

$$c(M) = \sum_{(u,v) \in M} c(u,v).$$

matching with no free vertices

Problem. Find a perfect matching in G^+ with minimum combined edge cost.

Running time will be expressed in terms of

$n = \#$ vertices of G

$m = \#$ edges of G.

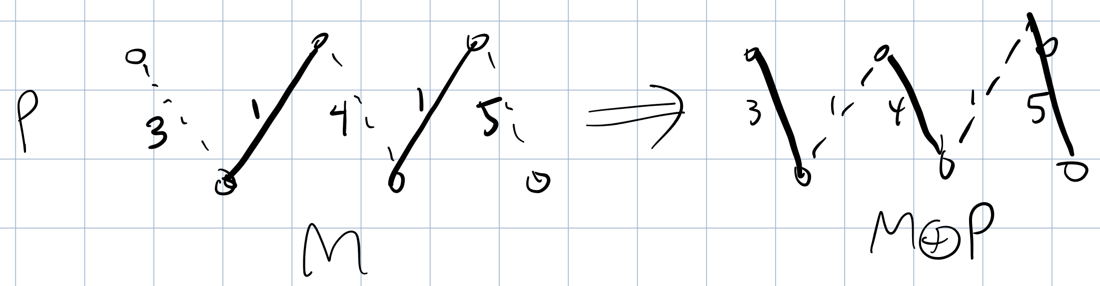
Idea: Start with empty matching M_0.

Build up a sequence of matchings $M_1, M_2, \ldots, M_{n/2}$ using one augmenting path at a time.

M_{k+1} will be equal to $M_k \oplus P_k$

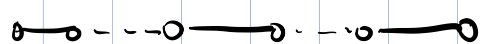
where P_k is some (carefully chosen) M_k-augmenting path.

How does the cost of a matching change when we replace M with $M \oplus P$?



$$\text{cost}(M) = 1+1 \qquad \text{cost}(M \oplus P) = 3+4+5$$

$$\Delta c(P; M) = \text{cost}(M \oplus P) - \text{cost}(M)$$
$$= 3 - 1 + 4 - 1 + 5$$



If we assign edge costs to directed edges in G_M as follows.

$$\text{cost}(u,v) = \begin{cases} c(u,v) & \text{if } (u,v) \notin M \\ -c(u,v) & \text{if } (v,u) \in M \end{cases}$$

then the bijection

$$\{M\text{-augmenting paths}\} \longleftrightarrow \left\{\begin{array}{l}\text{directed paths in } G_M \\ \text{from } L \cap F \text{ to } R \cap F\end{array}\right\}$$

is cost-preserving.

GREEDY ALGORITHM

$M_0 \leftarrow \emptyset$

for $k = 0, \ldots, \frac{n}{2} - 1$

compute G_M with its edge costs defined as above.

let $\boxed{P_k = \text{min cost directed path from } L \cap F \text{ to } R \cap F}$

let $M_{k+1} = M_k \oplus P_k$

end for

output $M_{n/2}$.

efficient search procedure?

$\boxed{\text{Correct?}}$ Yes, but surprisingly subtle to prove.

Induction hypothesis: M_k has the least cost of all matchings with k edges.
(See lecture notes for a self-contained proof of induction step.)

$\boxed{\text{Efficient?}}$ Use Bellman-Ford. Finding P_k takes time $O(mn)$.... if G_M has no negative cost cycles!

If $M = M_k$ had a negative cost cycle C in G_M, then consider $M' = M \oplus C$.
Note C is an even length M-alternating cycle. So M' is a matching which also has k edges.

$$\text{cost}(C) = \text{cost}(M') - \text{cost}(M).$$

By induction hypothesis above, $M = M_k$ has the least cost among all k-edge matchings.

still waiting to prove this.

$\therefore$ $\text{cost}(M') \geq \text{cost}(M)$

$\therefore$ $\text{cost}(C) \geq 0$

$\therefore$ we're allowed to use Bellman-Ford to find P_k.

<u>Running time:</u> $O(mn^2)$.

$\frac{n}{2}$ loop iterations, $O(mn)$ per iteration.

<u>Next lecture:</u>

1. Make this faster using Dijkstra in place of Bellman-Ford.
2. Prove the algorithm is correct.

Proving correctness holds the key to speeding up the algorithm.