

Matching

A matching in an undirected graph is a set of edges, no two having a common end vertex.

Bipartite graph: vertices can be partitioned into two sets, such that every edge has one end vertex in each set.

Maximum cardinality matching: find a matching containing as many edges as possible.

Maximum weight matching: in a graph with edge weights, find a matching with maximum total weight.

Bipartite vs. general graphs

Max card matching

Begin with empty matching.

Repeatedly find an augmenting path, augment.

Stop when no more augmenting paths.

Bipartite case:

$O(m)$ time per augmentation.

$O(n)$ augmentations

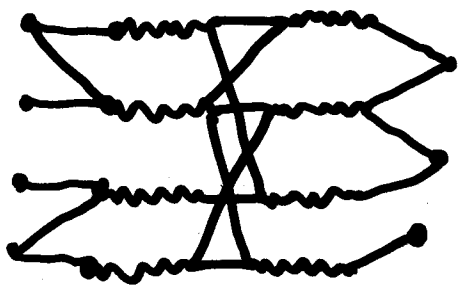
$\Rightarrow O(nm)$ time total.

Bipartite case faster

Build layered subgraph containing all

shortest aug paths by BFS

A B A B A B



free

free

Find aug paths in S 1 at a time by DFS

Total time per phase $\neq O(n)$.

Length of shortest aug path strictly

increases after a phase

$O(\sqrt{n})$ phases $\Rightarrow O(\sqrt{n}n)$ time

Each phase increases any path length:

Let $d(v)$ be shortest dist from an A-free vertex to v via an alternating path.

$d(v)$'s strictly increase along any shortest

any. path. New edges created by a

shortest any. go from larger to smaller $d(v)$.

Thus no shorter any path created by a

shortest any; after a phase, every

any path contains at least one edge

from larger to smaller $d(v) \Rightarrow$ longer

path.

$2\sqrt{n}$ phases:

Each phase increases matching size.

If $|\bar{M}| - |M| > \sqrt{n}$, $M \oplus \bar{M}$ contains

$> \sqrt{n}$ any paths, at least one of
length $< \sqrt{n}$ (only n vertices).

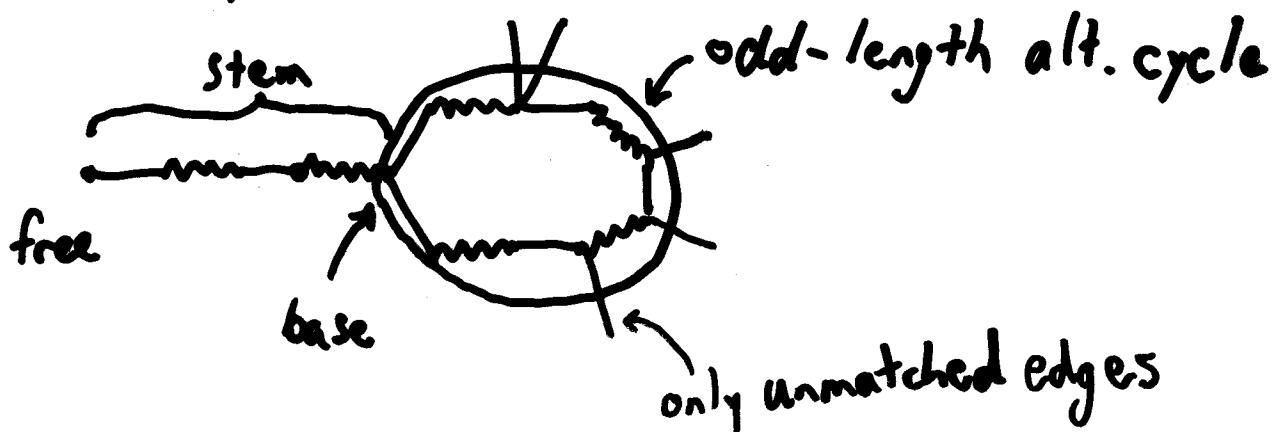
\Rightarrow After \sqrt{n} phases, shortest any path
has length $\geq \sqrt{n} \Rightarrow$ within \sqrt{n} of
max $\Rightarrow \leq \sqrt{n}$ more phases.

Max card matching on general graphs

Basic problem: how to find one
any path

(a vertex can be an A-vertex or a B-vertex;
a priori, one doesn't know which)

Edmonds: blossom-shrinking to find any
paths

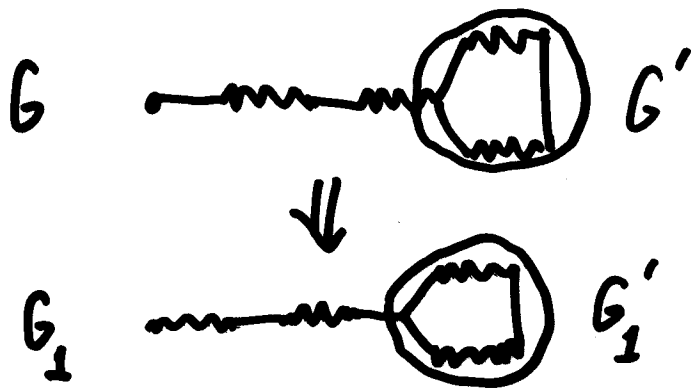


Thm: Let G' be formed from G by shrinking a blossom. Then G' contains an any path iff G does.

Pf. If G' contains an any path, then G does: expand blossom, link broken ends of path by going around blossom in correct direction (one broken end is blossom base).

Other direction is the hard part.

If the blossom has a non-trivial stem, sweep edges along it to make the base of the blossom free, obtaining G_1 from G (and G'_1 from G').



$G(G')$ has an any path iff $G_1(G_1')$ does.

Thus we need only show that if G_1 has an

any path, so does G_1' . Thus suppose G_1 has

an any path. Either it is an any path in G_1' or

it hits the blossom, in which case the part

from ~~the~~ end not the blossom base until it

first hits the blossom is an any path

in G_1' .

Edmonds' alg to find an any path
via blossom-shrinking (DFS version)

Start at any free vertex.

Grow an alt. search path.

If an edge extending the path hits the path,
shrink a blossom if the path is of odd length;
otherwise discard the edge.

When reaching a new free vertex, stop with
success.

When at a vertex or blossom with no unexplored
edges, delete the vertex or blossom.

After deleting a free vertex, start a new search
at an undeleted free vertex.

Time per any path: $O(m \alpha(n))$

(need set union to maintain blossoms)

Total time = $O(n m \alpha(n))$

Can improve to take advantage of

shortest any path idea:

very complicated

Nothing better is known, even though
sum of lengths of shortest aug
paths is $O(n \log n)$.

Note: k phases \Rightarrow max to within

$(1 - 1/k)$ factor: fast approximation

Generalizes to general graphs, weighted
matchings, shortest paths, max flows

$O(\sqrt{n}) \times \alpha$ and/or log factors