Graph Algorithms - Matchings in Graphs

1 Problem Definition

Let $G = (V, E)$ be an undirected graph

A set $M \subseteq E$ is a matching in G if no two edges in M have an end-vertex in common

A matching M is maximum if there is no matching of G with more edges than M

The matching number ν_G of G is the size of a maximum matching in G

Definition: Matching

Instance: a graph G
Task: determine ν_G

2 Alternating Paths and Cycles

Let $G = (V, E)$ be a graph and M be a matching inG

A path P is alternating with respect to M iff among every two consecutive edges of P, exactly one edge belongs to M

A cycle C is alternating with respect to M iff among every two consecutive edges of C, exactly one edge belongs to M.

A vertex u is matched by a matching M is it is the end-vertex of an edge of M; otherwise u is unmatched by M

For two sets A and B, the symmetric difference is the set

$$A \oslash B = (A \backslash B) \cup (B \backslash A)$$

Lemma
Let G be a graph with a matching M and an alternating path P with respect to M. If each end-point of P is either unmatched by M or matched by $M \cap P$, then $M \oslash P$ is another matching

3 Augmenting Paths

An alternating path P with respect to a matching M is augmenting if both endpoints of P are unmatched by M.

Lemma
Let G be a graph with a matching M and an augmenting path P with respect to M. Then $M \otimes P$ is a matching of size $|M|+1$

Hence, M is not maximum if there exists an augmenting path P with respect to M

We now prove that the converse holds as well.

That is, M is maximum if there exists no augmenting path P with respect to M

Lemma
Let $G = (V, E)$ be an undirected graph, and let M and M* be matchings in G. Then the subgraph $(V, M \otimes M*)$ is the disjoint union of

- Isolated vertices
- Alternating cycles with respect to both M and M*

- Alternating paths with respect to both M and M*

Lemma
Let $G = (V, E)$ be an undirected graph, and let M and M* be matchings in G such that $|M* | = |M| + k$ for some $k \geqslant 1$. Then G has at least k pairwise vertex-disjoint augmenting paths with respect to M

Proof The subgraph $G' = (V, M \otimes M*)$ consists of alternating paths and cycles, with respect to M and M* (and isolated vertices)

The set $M \otimes M*$ has exactly k more edges from M* than M

Hence G' contains k pairwise vertex-disjoint paths that start and end with edges of M*. These paths are augmenting paths of G with respect to M

From this lemma we find that a matching M is maximum if there exists no augmenting path P with respect to M.

We already proved that M is not maximum if there exists an augmenting path P with respect to M. We conclude:

Theorem
A matching M in a graph G is maximum iff G has no augmenting path P with respect to M

This suggests the following approach for solving matching:

1. Let M be the empty set
2. Check if there is an augmenting path P with respect to M
3. If so, let $M := M \otimes P$ and go to step 2. Otherwise output M

4 Alternating Trees

Let M be a matching in bipartite graph $G = (R \cup B, E)$. If all red vertices are matched, then M is maximum, Otherwise we try to find an augmenting path P with respect to M (if it exists):

1. Start with an unmatched red vertex v
2. Consider all blue neighbours to v. If one of then is unmatched, STOP (P has been found)
3. Consider all matched red neighbours of the newly found blue vertices
4. Consider all new blue neighbours of red vertices from step 3. If one of then is unmatched, STOP (P has been found). If no new blue neighbour is detected, STOP. Otherwise go to strp 3

If the algorithm terminated because no new blue neighbour was found in step 4, then we found an alternating tree T.

Consider the (bipartite) graph G-T and repeat the process until an augmenting path P is found or the set of unmatched red vertices has become empty

5 Correctness

Suppose no augmenting path (with respect to M) is found in steps 2 or 4

So we have found an alternating tree T. By construction, v is not an end-vertex of an augmenting path P

We claim that no other vertex in T is used for such a P either:

1. Observe that there are no edges from red vertices to (blue) vertices outside T (as our algorithm has terminated)
2. Any edge from a blue vertex in T to a red vertex outside T cannot be on such a P either, else we can modify P into a new augmenting path P' (with respect to M) that starts in v.

So we can safely remove the vertices of T and consider G-T

6 Running time

Since we only explore each edge once, the augmenting path algorithm runs in $O(|E|)$ time

Each occasion results in a matching that has one more edge than the previous matching. Hence we run the augmenting path algorithm at most $\frac{|V|}{2}$ times

Conclusion: The total running time of the algorithm for finding a maximum matching in a bipartite graph is $O(|E||V|)$ time

General Graphs: Edmond's algorithm for matching based on the bipartite algorithm, takes $O(|E||V|^2)$ time

7 Connections to other problems: vertex cover

Suppose you have only a small number of cameras that you want to place at crossroads in a city so that every road is "covered"

Definition: Vertex cover

A vertex cover of a graph G=(V,E) is a set $S \subseteq V$ so that each edge G has at least one end-vertex in S

Definition: Vertex cover number

The vertex cover number τ_G of G is the size of a smallest vertex cover of G

8 Vertex cover for Bipartite Graphs

Theorem
Let G be a bipartite graph. Then $\tau_G = \nu_G$

So we can use the matching algorithm for bipartite graphs to compute τ_G as well

Proof

Let M be a matching and C be a vertex cover of G. At least one end-vertex of every edge in M belong to C, so $|M| \leqslant |C|$

Let M* be a maximum matching of G. We construct a vertex cover C* with $|M*| = |C*|$, so C* has minimum size

As M* has maximum size, the matching algorithm for bipartite graphs constructs a forest F with alternating trees.

Let M_1 be the set of edges of M in F. Let $M_2 = M \backslash M_1$

We let C* consist of all blue end-vertices of edges in M_1 and all red end-vertices of edges in M_2

Then $|C*| = |M_1| + |M_2| = |M*|$. Moreover C* is a vertex cover. Otherwise, there is an edge vw where v is red, w is blue and neither of them belongs to C*. This implies that v is in some tree of F and w is not. However, then out algorithm would have discovered w at some point in step 4, a contradiction

9 Vertex cover for General Graphs

Instance: a graph G and integer $k \geqslant 0$
Task: Does G have a vertex cover of size at most k?

- Recall that this problem is computationally hard
- Let n and m be the number of vertices and edges of G. Brute force requires checking $O(n^k)$ possible choices for S

- Now assume k is relatively small compared to n. Can we find an algorithm that runs in $f(k) + O(n + m)$ time?

Rule 1. If v is not an end-vertex of an edge, then remove v, so consider (G-v,k)
Rule 2. If v has more than k neighbours then put v in the vertex cover and consider (G-v,k-1)

At some point, rules 1 and 2 can't be applied any more. Let (G',k') be the resulting instance. Note that we found (G',k') in $O(n + m)$ time

- If G' has more tan $k + k^2$ vertices, then the answer is no. For contradiction, assume S is vertex cover of G' with $|S| \leqslant k' \leqslant k$. By rule 2, there are at most $k|S| \leqslant k^2$ vertices that are not in S but that have a neighbour in S. Hence, there exists a vertex u in G' that has no neighbour S. BY rule 1, u has a neighbour v, so u or v must be in S, a contradiction
- Otherwise G' has at most $k + k^2$ vertices. We solve the problem on G' using brute force. This takes $f(k)$ time for some function f that only depends on k