Algorithm Mid-Exam Homework

(Edmond's Blossom Algorithm)

Qiu Yihang

June 2022

1 M-Augmenting Path

Proof. First we prove the necessity.

When M is a maximum matching, assume there exists an M-augmenting path $P : u_0u_1u_2...u_k$. Obvious k is odd since the number of vertices in any M-augmenting path must be even.

By the definition of M-augmenting path, we know $\{u_1, u_2\}, \{u_3, u_4\}, ... \{u_{k-2}, u_{k-1}\} \in M$, $\{u_0, u_1\}, \{u_2, u_3\}, ... \{u_{k-1}, u_k\} \notin M$. Also, $\forall e \in M, u_0 \notin e, u_k \notin e$.

Consider $M' = M \setminus \{\{u_1, u_2\}, \{u_3, u_4\}, ... \{u_{k-2}, u_{k-1}\}\} \cup \{\{u_0, u_1\}, \{u_2, u_3\}, ... \{u_{k-1}, u_k\}\}$.

By the definition of matching, we know all vertices on the path P, i.e. $u_0, u_1, ...u_k$ are not covered by $M \setminus \{\{u_1, u_2\}, \{u_3, u_4\}, ... \{u_{k-2}, u_{k-1}\}\}$. Thus, no two edges in M' share a same vertex.

Moreover, $|M'| = |M| - \frac{k-1}{2} + \left(\frac{k-1}{2} + 1\right) = |M| + 1 > |M|$. **Contradiction** to the assumption that M is a maximum matching.

Thus, when M is a maximum matching, there exists no M-aumenting path on G. □

Now we prove the sufficiency.

When exists no M-augmenting path on G, assume M is not the maximum matching.

We can always find a maximum matching M' s.t. $|M \cap M'|$ is maximized.

We can prove that $\forall \{u, v\} \in M$, at least one of u, v is also covered by M'. Otherwise, $M' \cup \{u, v\}$ is a matching with $|M' \cup \{u, v\}| > |M'|$. **Contradiction** to the assumption that M' is a maximum matching.

Meanwhile, since M and M' are both matchings, we know exist $2(|M'| - |M|)$ vertices in M' are not covered in M. Considering $|M| \leq |M'| - 1$, at least 2 vertices in M' are not covered in M. Let these two vertices be x, y.

Then we construct an M-augmenting path on G as follows.

1. Initialize P. Now P is a path with only one vertex x.

2. Suppose the last vertex in P is u.

   Find an edge $\{u, v\} \in M'$ and add the edge into P.

   (We have shown such edge always exists above.)

   If no edges in M is adjacent to v, terminate the process.

   Otherwise, find an edge $\{v, w\} \in M$ and add the edge into path P.

3. Repeat **Step 2**.

We have already shown that there exist at least two vertices covered by M' and not covered by M. Thus, the process above will eventually come to an end.

Since when constructing P, we select edges alternatively in M and M', start from a vertex not covered by M and end at a vertex not covered by M, we know P is an M-augmenting path.

Contradiction!

Thus, when there exists no M-augmenting path on G, M is a maximum matching. □

In conclusion, M is a maximum matching of G **iff.** no M-augmenting path exists. ■

2 Contract Odd-Size Cycles

Proof. First we prove the necessity, i.e. when M is a maximum matching of G, $M \setminus C$ is a maximum matching of G'.

Obvious M contains exactly k edges in C. Otherwise, we can find an edge e in C s.t. $M \cup \{e\}$ is matching with larger size than M. **Contradiction** to the assumption that M is a maximum matching.

Thus, $M \setminus C$ covers at most 1 vertex in C. Therefore, $M \setminus C$ is a valid matching on G'.

Assume exists M' is a matching on G' with larger size than $M \setminus C$. By the definition of cycle contracting, we know M' covers at most 1 vertex of C in G. Then we can find k edges s.t. no endpoints of these edges are covered by M'. Adding these k edges to M' will generated a matching $\mathcal{M}$ on G. Meanwhile, $|\mathcal{M}| = |M'| + k > |M \setminus C| + k = |M|$.

Contradiction to the assumption that M is a maximum matching.

Thus, when M is a maximum matching of G, $M \setminus C$ is a maximum matching of G'. □

Now we prove the sufficiency, i.e. when $M \setminus C$ is a maximum matching of G', M is a maximum matching of G.

** In fact, it is wired that $M \setminus C$ (on G') is defined before M is defined. Thus, I suppose the meaning of the sufficiency is that we first find a matching M on G and then we check whether $M \setminus C$ is a maximum matching on G'.*

When $M' \triangleq M \setminus C$ is a maximum matching on G', we know M' covers at most 1 vertex in C on G. Thus, we can always find exactly k edges not adjacent to each other in C on G s.t. no endpoints of these edges are covered in M'. Adding these k edges into M' generates a matching M on G.

Assume exists a matching M_0 of G with greater size than M.

Then $M_0 \setminus C$ is also a matching on G. (A subset of a matching is also a matching, which is trivial.) Given how G' is generated from G, we know $M_0 \setminus C$ is a matching on G'.

Meanwhile, obvious M_0 at most contains k edges in C. Thus, $|M_0 \setminus C| \geq |M_0| - k > |M| - k = |M'|$. **Contradiction** to the assumption that $M' = M \setminus C$ is a maximum matching on G'.

Thus, when $M \setminus C$ is a maximum matching on G', M is a maximum matching of G. □

In conclusion, $M \setminus C$ is a maximum matching on G' **iff.** M is a maximum matching of G. ■

3 M-Alternating Forest

Proof. For any given M, obvious $\mathcal{F} \triangleq (V(\mathcal{F}), E(\mathcal{F}))$ with $V(\mathcal{F}) = \varnothing,\ E(\mathcal{F}) = \varnothing$ is always a valid M-alternating forest. Thus, M-alternating forest exists. □

We design an algorithm for finding a maximal M-alternating forest as follows.

1. Start from $F = \mathcal{F}$. (Recall $\mathcal{F} = (V(\mathcal{F}), E(\mathcal{F}))$ with $V(\mathcal{F}) = \varnothing,\ E(\mathcal{F}) = \varnothing$)
   Let the set of all vertices uncovered by M be V_{un}.
2. Pick a vertex $r \in V_{un}$ uncovered by F. Add r to $V(F)$. Obvious r is an outer vertex.
   We construct a tree with root r by the following process.
   The process is a modified traverse of graph G, which can be realized through DFS or BFS.
   - Find all vertices adjacent to the current vertex u of the tree and uncovered by F so far.
   - For each vertex v satisfying the conditions above, check whether exists a w uncovered by F s.t. $\{v, w\} \in M$.
     * If so, add v, w into $V(F)$ and add $\{v, w\}$ into $E(F)$.
       Obvious v is an inner vertex while w is an outer vertex.
       Since $\{v, w\} \in M$, both v and w are covered by M.
     * If not, check the next vertex.
   - Traverse to the next outer vertex unvisited on the current tree.
3. Check if all vertices in V_{un} are covered by F , i.e. whether $V_{un} \subset V(F)$.
   If not, jump to step 2.
   If so, terminate the process. F is a maximal M-alternating forest. □

Now we prove the correctness of the algorithm.

Step 2 of the algorithm ensured that the each component contains exactly one vertex uncovered by M, i.e. the root we construct for the component (i.e. the tree). Meanwhile, for each inner vertex, it has two incident edges in F (ensured by step 2) and exactly one of the two edges is in M.

Now we prove that F is maximal, i.e. adding any vertex or edge into F is impossible. Let F' be the forest of F added with certain edges or vertices.

Since all vertices uncovered by M is already covered by F, adding any vertices will generate a component with no vertices uncovered by M. Thus, F' is not an M-alternating forest.

For any edge $\{u, v\} \in E \setminus F$,

CASE 01. Both endpoints are already covered in F. Adding $\{u, v\}$ will merge two components and give a component with two vertices uncovered by M (roots of the two trees). Then F' is not an M-alternating forest.

CASE 02. Both endpoints are not covered in F. Since $V_{un} \subset V(F)$, both u and v are covered by M. After adding $\{u, v\}$ to F, exactly one of u and v is inner vertex while the other one is outer vertex. Without loss of generality, suppose u is the inner vertex.

If u and v can be added into F, there must exist an outer vertex w s.t. $\{u, w\} \notin M$. If such condition is satisfied, u and v should be added to F in step 2. **Contradiction.**

CASE 03. Exactly one of the endpoints is in F. Suppose $u \in V(F), v \notin V(F)$.

If u is an inner vertex, after adding $\{u, v\}$, exists an inner vertex u with more three incident edges. Thus, F' is not an M-alternating forest.

If u is an outer vertex and $\{u, v\}$ can be added into F, there must exist a w uncovered by F and $\{v, w\} \in M, \{u, v\} \notin M$. If these conditions are satisfied, u and v should be added to F in step 2. **Contradiction**.

Thus, the algorithm finds a maximal M-alternating forest. □

Now we analyze the complexity of the algorithm.

We visit every vertex constant times. Thus, the algorithm takes $O(|V|)$, i.e. polynomial-time.

Thus, we can find a maximal M-alternating forest in polynomial-time. ■

4 M-Alternating Path Must Contain Two Consecutive Outer Vertices

Proof. We know an M-alternating path starts from a vertex not covered by M and ends at another vertex not covered by M.

By the definition of maximal M-alternating forest, we know all vertices not covered by M are in $V(F)$ and every component of F contains exactly one vertex not covered by M, i.e. the root of the component.

Thus, an M-alternating path must start from the root of one component in F and ends at the root of another component of F.

Then the M-augmenting path must contain an edge whose two endpoints are in two different components of F. We will use the word *crosswalk* to refer to such edge in the following proof.

Now we prove at least exists a crosswalk, whose two endpoints are both outer vertices.

Let the M-augmenting path be P. Use T_x to denote the component of F which x belongs.

It is trivial that all crosswalks are not in M. Otherwise, the endpoints of a certain crosswalk are both not covered by M, i.e. the endpoints are both the root of their own respective components. Then exists an edge e between vertices not covered by M, i.e. $M \cup \{e\}$ is a matching with larger size than M. **Contradiction.**

Since the crosswalk $\{u_i, v_i\} \notin M$, then the two edges next to $\{u_i, v_i\}$ in P must be in M.

Assume for any crosswalk, at least one of its endpoint is an inner vertex. Let the i-th crosswalk in P be $\{u_i, v_i\}$ with u_i being an inner vertex. Then u_i can't be the root of the T_{u_i}.

For any crosswalk $\{u_i, v_i\}$, there are two types.

(By assumption, **TYPE 00** crosswalks, i.e. u_i and v_i are both outer vertices, do not exist.)

TYPE 01. v_i is outer vertex. Then the edge in T_{u_i} next to $\{u_i, v_i\}$ in P is between the child of u_i and u_i and the edge T_{v_i} next to $\{u_i, v_i\}$ in P is between v_i and its parent.

TYPE 02. v_i is also an inner vertex. Then the edge in T_{u_i} next to $\{u_i, v_i\}$ in P is from the child of u_i to u_i and the edge T_{v_i} next to $\{u_i, v_i\}$ in P is from v_i to its child.

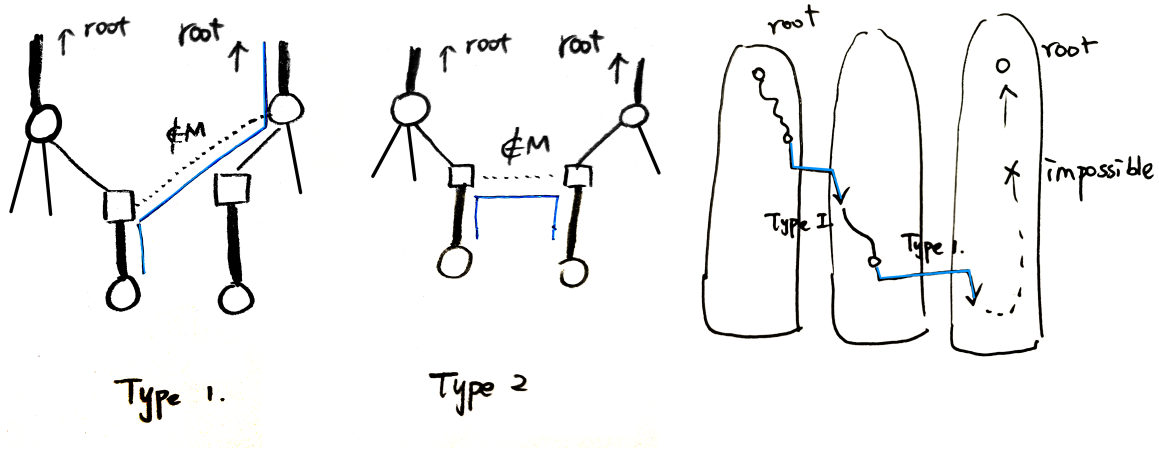

Figure 1: Type 01 and Type 02 Crosswalk; A Visualization of Proof by Contradiction

We know P must start from the root of one component of F and end at the root of another component of F. Suppose the edge in P before the first crosswalk is $\{u_0, v_0\}$. Then u_0 is the parent of v_0 in T_{u_0}.

By the assumption that all crosswalks are not between two outer vertices, the crosswalk next to $\{u_0, v_0\}$ must be between an inner vertex and an outer vertex. Then we know the edge after the first crosswalk is from a parent to child. Thus, the edge before the next crosswalk is also from a parent to its child in F's components since each vertex has exactly one parent.

Then we know the next crosswalk should be between an inner vertex and an outer vertex.

By induction, we know all crosswalks are between an inner vertex and an outer vertex. Then in the last component of F passed in P, all edges in the path is from a parent to its child.

Nevertheless, P should end at the root of a component in F, i.e. the last edge should go from a child to its parent. **Contradiction.**

Thus, M-alternating path must contain 2 consecutive outer vertices. ■

5 Either M-Alternating Path Or Blossom

5.1 If u and v Belong to Distinct Components

Proof. We can construct an M-augmenting path as follows.

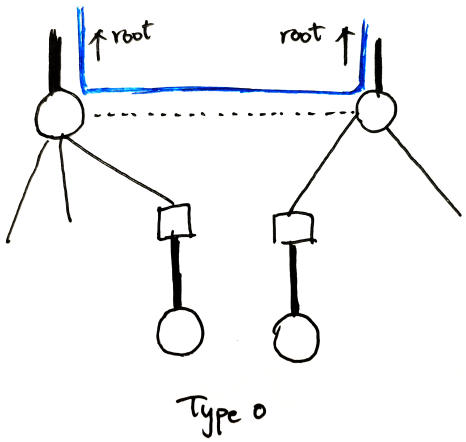

Figure 2: Type 00 Crosswalk; Construction of M-Augmenting Path in Problem 5

Let T_x be the component where x belongs in M-alternating forest. Since u and v belong to distinct components, we can always find a path P_u from the root of T_u to u in T_u and a path P_v from v to the root of T_v in T_v.

Construct path $P = P_u \cup \{u, v\} \cup P_v$. Now we prove P is an M-augmenting path.

The first and the last vertex in P is not covered by M. (since the root of any component in M-alternating forest is not covered by M.)

By the definition of M-alternating forest, we know all edges in P_u and P_v are alternatively in M and not in M.

Meanwhile, since u and v are both outer vertices, we know the last edge in P_u and the first edge in P_v are in M. Given that $\{u, v\} \notin M$ (all crosswalks are not in M, already shown in **4**), we know all edges in P are alternatively in M and not in M.

Thus, if u and v belong to distinct components, an M-aumenting path exists. ■

5.2 If u and v Belong to the Same Components

Proof. We can construct a blossom as follows.

Since u and v belong to the same component T of an M-alternating forest, we can always find a path $P_{v\to u}$ from v to u. (Since T is a tree, u and v are connected.)

Construct Cycle $C = P_{v\to u} \cup \{u, v\}$. Now we prove C is a blossom.

Since all inner vertices in a component of M-alternating forest has exactly two incident edges, i.e. to its parent and to its child, we know the common ancestor of two vertices in the component must be an outer vertex.

Since both u and v are outer vertices, we know the distance from their common ancestor to u is even and so is the distance to v. Thus, the length of the path from v to u is also even.

Therefore, $|C| = |P_{v\to u}| + 1$ is odd, i.e. cycle C is a blossom. ■

6 Edmond's Blossom Algorithm

Proof. Based on the observations above, we design the following algorithm.

1. Start with $M = \varnothing$. Repeat adding edges whose endpoints are not covered by M to M.
   Then we have a maximal matching M.
2. Construct a maximal M-alternating forest F. The process is described in **3**.
3. Find all $\{u, v\} \in E \setminus F$ s.t. both u and v are outer vertices.
   Let the set of these edges be $E_{\texttt{Out}}$.
4. For each $\{u, v\} \in E_{\texttt{Out}}$, check whether u and v belong to the same component in F.
   - If so, contract the blossoms constructed by u and v in **5.2** in the original graph G.
     Note that in this process, F and M should also be updated. (Blossoms should be contracted in F and M corresponding to the change of G).
   - If not, then u and v belong to different components.
     Construct the M-augmenting path P.
     Note if exists several M-augmenting paths, we just take one of them.
5. If no $\{u, v\} \in E_{\texttt{out}}$ satisfies that u and v belong to different components, go to step 7.
   Otherwise, go to step 6.
6. Let the M-augmenting path we construct in step 4 be $P : u_0u_1...u_{2k+1}$.
   Construct a new matching $M' = M \setminus \{\{u_1, u_2\}, ... \{u_{2k-1}, u_{2k}\}\} \cup \{\{u_0, u_1\}, ... \{u_{2k}, u_{2k+1}\}\}$.
   Jump to step 2 with this new matching M'.
7. Restore M to the maximum matching on G by opening all blossoms.
   The opening process is given in **2**.
   Terminate the algorithm. M is the maximum matching on G.

Now we prove the correctness of the algorithm.

(1) First we prove that the result of the algorithm is correct.

By **4**, we know any M-augmenting path must contain two consecutive outer vertices. By the definition of M-alternating forest F, we know an edge with two endpoints being consecutive outer vertices must be an edge in $E \setminus F$.

Thus, if exists no edge in $E \setminus F$ whose endpoints belong to different components of F, there exists no edge in E whose endpoints are in different components of F. Then M-augmenting path does not exist, i.e. M is a maximum matching (By **1**).

Thus, the M returned by our algorithm is a maximum matching on G.

(2) Then we prove each step can function as expect correctly.

Step 1. The process ensures the definition of matching and maximal matching. Thus the initial M is a maximal matching.

Step 2. The correctness is proved in **3**.

Step 3-4. The construction of blossoms and M-augmenting paths is justified in **5**. The correctness of contracting blossoms is justified in **2**.

Step 5-6. By **1**, we know $\{\{u_1, u_2\}, ... \{u_{2k-1}, u_{2k}\}\} \in M$ and u_0, u_{2k+1} are not covered by M. Thus, M' is a valid matching.

Assume M' is not a maximal one. Then exists $\mathcal{M}$ s.t. $M' \subset \mathcal{M}', |\mathcal{M}| > |M'|$. Then adding $\mathcal{M} \setminus M'$ into M will generate a matching with size $|M| + 1$. **Contradiction**.

Thus, M' is also a maximal matching.

Step 7. The correctness is given in **2**.

(3) Now we prove that our algorithm will terminate and return a result in finite time.

In step 6, we generate a new maximal matching M' from M. $|M'| = |M| - k + (k+1) = |M| + 1$. Thus, after an iteration, the size of the matching increases.

Obvious $|M| \leq |V| = n$. Thus, the algorithm will iterate for at most $|V| = n$ times.

Therefore, the algorithm will terminate in finite time.

In conclusion, the algorithm we designed is correct. □

Now we analyze the time complexity of the algorithm.

Step 1 takes at most $O(|E|)$ time since at most $|E|$ edges exist in M.

Step 2 takes takes $O(|V|)$ time to construct an M-alternating forest. (By **3**)

Step 3 takes at most $O(|E|)$ time to scan all edges in $E \setminus F$.

Obvious there are $O(|V|)$ blossoms. For each blossom, the contraction takes at most $|E|$ time. Also, there are at most $|E|$ edges in an M-augmenting path. Thus, Step 4 takes at most $O(|V||E|)$ time in total.

Step 5-6 takes $O(|E|)$ to generate a new maximal matching since P contains $O(|E|)$ edges.

Step 7 takes at most $O(|E|)$ time since there are $O(|E|)$ edges in all blossoms.

For each iteration, we run step 2 to step 6. Thus, an iteration takes $O(|V||E|)$ time.

Since each iteration will increase the size of maximal matching M by 1, the algorithm will terminate after at most $|V|$ time.

Thus, the time complexity of the algorithm is $O(|V| \cdot |V||E|) = O(n^2 m)$. ■