CMPUT 675: Topics in Combinatorics and Optimization **Fall 2016**

Lecture 1 (Sep 2 & 7): Introduction, Bipartite Matching

Lecturer: Zachary Friggstad *Scribe: Zachary Friggstad*

1.1 Introduction

The scribe notes for this course summarize the results we discussed in class with complete proofs. Do not worry about including much exposition in your writeup, this is mostly a record of the facts.

Common Assumptions
Often we let n denote the number of vertices and m the number of edges of a given graph, as long at it is clear from the context what graph is being discussed.

A *graph* is undirected, unless we explicitly call it a *directed graph*.

Unless stated otherwise, we will assume all undirected graphs are connected and all directed graphs are weakly connected (meaning their undirected version is connected). It is almost always a simple exercise to take any algorithm for a connected (or weakly connected) graph and have it work on an arbitrary graph with only $O(n+m)$ overhead. This assumption also means $m \geq n-1$, which will simplify some running time bounds.

We do not discuss the running time of each small step of the algorithms. Usually we only discuss the main ideas behind implementing it with the target running time bound, the smaller details are left up to you.

1.2 Bipartite Matching

Let $\mathcal{G} = (V;E)$ denote an undirected graph. For some $S \subseteq V$, let

$$\delta(S) = \{uv \in E : |S \cap \{u,v\}| = 1\}.$$

That is, $\delta(S)$ is all edges with precisely one endpoint in S. For a vertex $v \in V$, we let $\delta(v)$ denote $\delta(\{v\})$. This is the set of edges having v as an endpoint.

Definition 1 *A* **matching** *$M \subseteq E$ is simply a subset of edges such that $|\delta(v) \cap M| \leq 1$.*

A matching is depicted in Figure 1.1.

Definition 2 *In the* Maximum Cardinality Matching *problem, we are given an undirected graph $\mathcal{G} = (V;E)$, the goal is to find a matching M of maximum possible size.*

We will eventually describe an algorithm for solving Maximum Cardinality Matching in polynomial time in any graph. In our first lecture topic, we will solve it in bipartite graphs.

Definition 3 *An undirected graph $\mathcal{G} = (V;E)$ is* **bipartite** *if it is possible to partition V into two sets of vertices L, R such that each $e \in E$ has one endpoint in L and the other in R.*

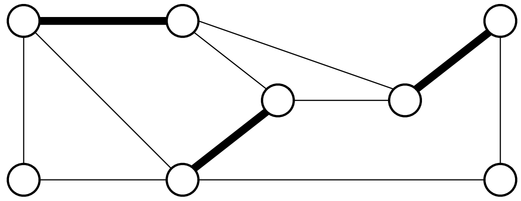

Figure 1.1: A matching (bold edges).

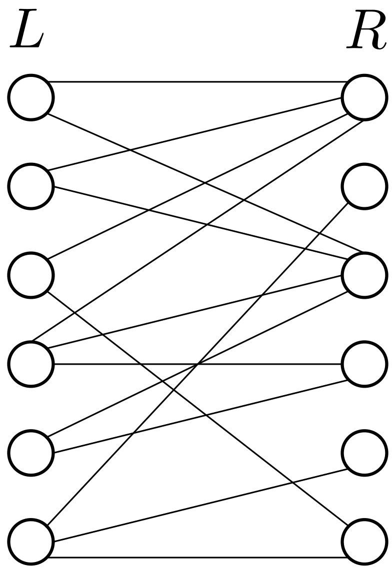

Figure 1.2: A bipartite graph.

Figure 1.2 shows a bipartite graph with "sides" L and R.

Theorem 1 *The* Maximum Cardinality Matching *problem can be solved in $O(n \cdot m)$ time in bipartite graphs.*

The algorithm iterates the following procedure: given a matching M we either find a matching M' with $|M'| = |M| + 1$ or else correctly declares that M is a maximum-size matching. The main concept that drives the algorithm is that of an alternating path.

1.2.1 Alternating Paths

Definition 4 *Let M be a matching in a graph $\mathcal{G} = (V; E)$. A vertex $v \in V$ is M-**matched** if some edge in M has v as an endpoint, otherwise v is M-**exposed**. An M-**alternating path** is a sequence of distinct vertices $v_1, v_2, \ldots, v_k$ with $k \geq 2$ where v_1, v_k are M-exposed, $v_iv_{i+1} \in M$ for every even index i, and $v_iv_{i+1} \in E - M$ for every odd index i.*

We sometimes drop the reference to M when using the term M-exposed, M-matched, and M-alternating if the matching being discussed is clear. An alternating path is depicted in Figure 1.3.

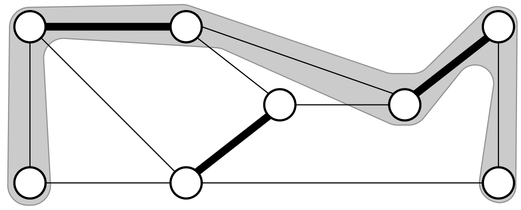

Figure 1.3: An alternating path. Replacing the highlighted edges in the matching with the highlighted edges not in the matching produces a larger matching.

Lemma 1 *Let M be a matching in a graph $\mathcal{G} = (V; E)$ and P the set of edges of some M-alternating path. Let $M' = (M - P) \cup (P - M)$. Then M' is a matching and $|M'| = |M| + 1$.*

Proof. Let $v_1, v_2, \ldots, v_k$ be the sequence of vertices of the M-alternating path.

Note $M - P$ is a matching that leaves all of $v_1, v_2, \ldots, v_k$ exposed. Also note that $P - M = \{v_1v_2, v_3v_4, \ldots, v_{k-1}v_k\}$ is a matching that only matches vertices on M. Therefore, $M' = (M - P) \cup (P - M)$ is a matching.

Conclude by noting $|M'| - |M| = |P - M| - |P \cap M| = \frac{k}{2} - \left(\frac{k}{2} - 1\right) = 1$. ■

Thus, we can find larger matchings by finding alternating paths. It turns out that the lack of an M-alternating path signals that M is a maximum matching.

Lemma 2 *M is a maximum matching if and only if there is no M-alternating path.*

Proof. We already showed the existence of an M-alternating path shows M is not a maximum matching. So now suppose M is not a maximum matching. We will find an M-alternating path.

Let M^* be a maximum matching, so $|M^*| > |M|$. Construct the graph $\mathcal{H} = (V; M \dot\cup M^*)$, meaning add each edge from M and M^* and keep two copies of each $e \in M \cap M^*$ (note $\mathcal{H}$ may have parallel edges). Since M and M^* are matchings, then $\deg_{\mathcal{H}}(v) \leq 2$ for each $v \in V$. This is equivalent to saying that $\mathcal{H}$ is comprised of vertex-disjoint paths and cycles (parallel edges are length-2 cycles).

Every cycle in $\mathcal{H}$ alternates between edges in M and edges in M^* so $|C \cap M^*| = |C \cap M|$ for every cycle C. This, plus the fact that $|M^*| > |M|$, means there is a path P in $\mathcal{H}$ with $|P \cap M^*| > |P \cap M|$. Such a path P starts and ends with edges in M^*, so its endpoints are M-exposed. That is, P is an M-alternating path. ■

Note
In proving Lemmas 1 and 2, we did not rely on the bipartite structure of $\mathcal{G}$. These will also be the foundation of our Maximum Cardinality Matching algorithm in general graphs. The main effort is then to find an alternating path. It is much simpler to find such a path in a bipartite graph.

1.2.2 Finding Alternating Paths in Bipartite Graphs

Lemma 3 *Given a matching M in a bipartite graph $\mathcal{G} = (L \cup R; E)$ we can either find an M-alternating path or determine that no M-alternating path exists in $O(m)$ time.*

Proof. Let $\overrightarrow{\mathcal{G}_M}$ be the directed graph $(V \cup \{s, t\}; \overrightarrow{E}_M)$ obtained from $\mathcal{G}$ by first directing each edge of M to point toward L and directing each edge of $E - M$ to point toward R. Then add two new *auxiliary* vertices s, t to

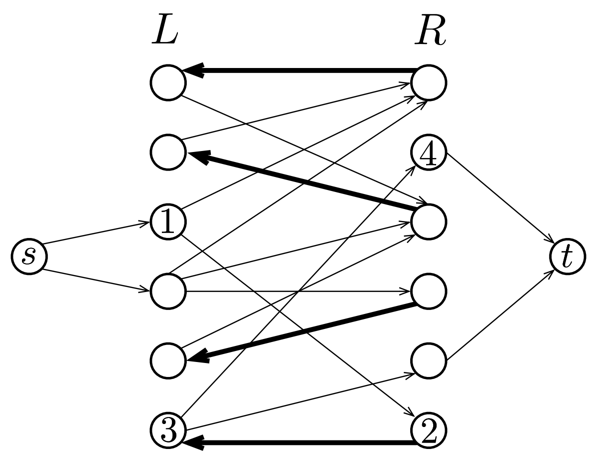

Figure 1.4: The directed graph constructed from the matching depicted in bold and an $s-t$ path $s, 1, 2, 3, 4, t$ corresponding to the alternating path $1, 2, 3, 4$ in $\mathcal{G}$.

$\overrightarrow{\mathcal{G}_M}$; s has an outgoing edge to each M-exposed vertex in L and a t has an incoming edge from each M-exposed vertex in R.

Any $s-t$ path $\overrightarrow{P}$ in $\overrightarrow{\mathcal{G}_M}$ corresponds to an M-alternating path: if we remove s, t from $\overrightarrow{P}$ then it starts and ends at M-exposed vertices. Furthermore, by how the edges of $\mathcal{G}$ were oriented, the (original undirected) edges used by $\overrightarrow{P}$ alternate between an edge of $E - M$ and an edge of M. Conversely, any M-alternating path $v_1, \ldots, v_k$ in $\mathcal{G}$ can be extended to a $s-t$ path in $\overrightarrow{\mathcal{G}_M}$ in the obvious way: in $\overrightarrow{\mathcal{G}_M}$ the path is $s, v_1, \ldots, v_k, t$.

Such an $s-t$ path can be found in $O(m)$ time using, say, a breadth-first search. ■

1.2.3 Summary

Algorithm 1 succinctly describes the main loop of the algorithm. Algorithm 2 describes how to find an M-alternating path. Note The running time of Algorithm 2 is $O(m)$; one can run a linear-time search (e.g. breadth-first or depth-first search) to find an $s-t$ path. The loop in algorithm 1 will iterate at most $\frac{n}{2}+1$ times because each iteration, apart from the last iteration, will reduce the number of M-exposed vertices by 2.

Algorithm 1 Maximum Cardinality Matching Algorithm for Bipartite Graphs

Input: Undirected bipartite graph $\mathcal{G} = (L \cup R, E)$
Output: A maximum-size matching $M \subseteq E$.

```
M ← ∅
while G contains an M-alternating path P {c.f. Algorithm 2} do
  M ← (M − P) ∪ (P − M)
end while
return M
```

Algorithm 2 Finding an M-alternating path.

Input: Undirected bipartite graph $\mathcal{G} = (L \cup R, E)$, matching M
Output: An M-alternating path or NONE
  $\overrightarrow{\mathcal{G}_M} \leftarrow (V \cup \{s, t\}, \overrightarrow{E_M})$ {as described in the proof of Lemma 3}
  if there is an $s - t$ path $\overrightarrow{P}$ in $\overrightarrow{\mathcal{G}_M}$ **then**
    return edges of $\mathcal{G}$ corresponding to edges used in $\overrightarrow{P}$
  else
    return NONE
  end if

1.3 Minimum Vertex Cover

Definition 5 *Let $\mathcal{G} = (V; E)$ be an undirected graph. A* **vertex cover** *$U \subseteq V$ is a subset of nodes such that each $e \in E$ has at least one endpoint in U. The* MINIMUM CARDINALITY VERTEX COVER *problem is to find a vertex cover U with minimum possible size.*

Unlike MAXIMUM CARDINALITY MATCHING, the MINIMUM CARDINALITY VERTEX COVER problem is **NP**-hard in general. However, it can be solved efficiently in bipartite graphs.

To start, we note the following relationship that holds for any graph.

Lemma 4 *Let $\mathcal{G} = (V; E)$ be an undirected graph, M a matching, and U a vertex cover. Then $|M| \leq |U|$.*

Proof. For each $e \in M$, let v_e be an endpoint of e that lies in U. Note that $v_e \neq v_{e'}$ for distinct $e, e' \in M$ because M is a matching. Therefore $|M| = |\{v_e : e \in M\}| \leq |U|$. ■

For bipartite graphs, we will now see that a maximum matching has the same size as a minimum vertex cover. This is an example of many *min/max relations* we will see throughout the term.

Theorem 2 (König-Egervary Theorem) *Let $\mathcal{G} = (L \cup R; E)$ be a bipartite graph and M a maximum matching. Furthermore, let Z be the vertices that can be reached by a directed path from s in $\overrightarrow{\mathcal{G}_M}$. Then $U = (L - Z) \cup (R \cap Z)$ is a vertex cover with $|U| = |M|$.*

Proof. We first show that U is a vertex cover. Consider any $e = uv \in E$ with $u \in L, v \in R$. If $e \in M$ then the only edge in $\overrightarrow{\mathcal{G}_M}$ entering u is the directed version of e in $\overrightarrow{\mathcal{G}_M}$. So it cannot be that $u \notin Z$ yet $v \in Z$, so $u \in U$. If $e \notin M$ then e is directed from u to v in $\overrightarrow{\mathcal{G}_M}$. Again, it cannot be that $u \in Z$ yet $v \notin Z$, so $u \in U$.

We finish by showing $|U| = |M|$. Note that $v \in U$ means v is not M-exposed: all M-exposed vertices in L are reachable from s by one edge (so are in Z) and no M-exposed vertices in R are in Z otherwise we have an M-alternating path. We also cannot have $u, v \in U$ for any $uv \in M$ because $v \in U \cap R$ and the fact that the edge uv is directed from v to u in $\overrightarrow{\mathcal{G}_M}$ would imply $u \in Z \cap L$, contradicting $u \in U$.

Thus, each edge of M has precisely one endpoint in U and no M-exposed nodes are in U, so $|U| = |M|$. ■

This is illustrated in Figure 1.5. Note the assumption that $\mathcal{G}$ is bipartite is necessary, the complete graph K_3 on 3 vertices has maximum matching size 1 but minimum vertex cover size 2.

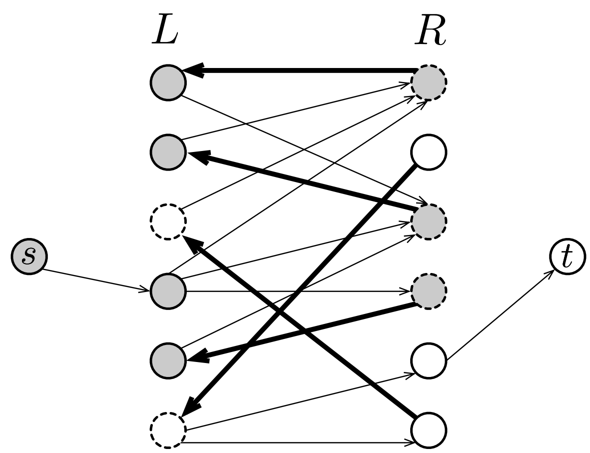

Figure 1.5: The shaded vertices are reachable from s and the vertices with dashed boundaries are the vertex cover. Also, the set $S \subseteq L$ of shaded nodes on the left has $|N(S)| < |S|$ as per Hall's Criteria (Section 1.4).

1.4 Hall's Criteria

We can get another useful statement from these observations. A trivial bound on the size of a maximum matching is $|L|$ (and also $|R|$) There is a useful condition that tells us precisely when this bound can be attained.

Notation: For an undireced graph $\mathcal{G} = (V; E)$ and some $S \subseteq V$, let $N(S)$ be the vertices not in S that are adjacent to a vertex in S. More precisely, $N(S) = \{v \in V - S : uv \in E \text{ for some } u \in S\}$.

Theorem 3 (Hall's Criteria) *Let $\mathcal{G} = (L \cup R; E)$ be a bipartite graph. The size of a maximum matching is $|L|$ if and only if $|N(S)| \geq |S|$ for every $S \subseteq L$.*

Proof. If $|N(S)| < |S|$ for some $S \subseteq L$ then clearly no matching can match all vertices in S so the maximum matching has size $< |L|$.

Conversely, suppose the maximum matching M has $|M| < |L|$. As in the proof of Theorem 2, let Z be all vertices reachable from s in $\overrightarrow{\mathcal{G}_M}$ and let $S = L \cap Z$. We claim every $w \in N(S)$ is matched to some node in S.

First, note that w cannot be M-exposed, otherwise the fact that $w \in N(S)$ and that w is not matched means $w \in Z$. But this is impossible; a path from s to an M-exposed node in R yields an M-alternating path contradicting the fact that M is a maximum matching.

So w is matched to a vertex, say to $u \in L$. We now show $u \in S$. Assume otherwise. Then $vw \in E - M$ for some $v \in S, v \neq u$ (in order for w to be in $N(S)$). But $vw \notin M$ means vw is directed toward w in $\overrightarrow{\mathcal{G}_M}$. This, plus the fact that $v \in S$ means $w \in Z$, so then $u \in S$ as well because $uw \in M$.

The fact that every $w \in N(S)$ is matched to some vertex in S shows $|N(S)| \leq |S|$. But $|M| < |L|$ means there is an M-exposed vertex in L. All M-exposed vertices of L also lie in S, so in fact $|N(S)| < |S|$. ∎

An example of a set $S \subseteq L$ with $|N(S)| < |S|$ when $\mathcal{G}$ does not have a matching of size $|L|$ is also depicted in Figure 1.5.

1.5 The Hopcroft-Karp Matching Algorithm

The previous MAXIMUM CARDINALITY MATCHING algorithm in bipartite graphs finds a maximum matching in time $O(n \cdot m)$ by iterating an $O(m)$-time algorithm that either certifies the current matching M is maximum or finds a matching M' with $|M'| = |M| + 1$.

We can speed this up by sometimes finding a matching M' with $|M'|$ being much larger than $|M|$ in $O(m)$ time, which should reduce the number of iterations. This is accomplished by identifying many vertex-disjoint alternating paths and simultaneously alternating the matching along each such path.

Definition 6 *Let M be a matching in a graph $\mathcal{G} = (L \cup R; E)$. Let α_M denote the length of a shortest M-alternating path. A collection of M-alternating paths $P_1, P_2, \ldots, P_b$ is M-**blocking** if:*

- *The paths are vertex-disjoint.*
- *$P_i = \alpha_M$ for each $1 \leq i \leq b$.*
- *Every M-alternating path of length α_M shares a vertex with at least one P_i.*

The following is proven in the same way as Lemma 1, keeping in mind the paths are vertex-disjoint.

Lemma 5 *If $P_1, \ldots, P_b$ are vertex-disjoint M-alternating paths, then $M' = (M - \cup_i P_i) \cup (\cup P_i - M)$ is matching with $|M'| = |M| + b$.*

The improved algorithm (Algorithm 3) simply searches for a blocking collection of alternating paths in each iteration, rather than just a single alternating path as before. We will soon show how to find such a blocking collection in linear time.

Algorithm 3 Hopcroft-Karp Algorithm for MAXIMUM CARDINALITY MATCHING in Bipartite Graphs

Input: Undirected bipartite graph $\mathcal{G} = (L \cup R, E)$
Output: A maximum-size matching $M \subseteq E$.

$M \leftarrow \emptyset$
while $\mathcal{G}$ contains an M-alternating path **do**
  $P_1, P_2, \ldots, P_b \leftarrow$ an M-blocking collection of paths {c.f. Algorithm 4}
  $M \leftarrow (M - \cup_i P_i) \cup (\cup P_i - M)$
end while
return M

To analyze the running time, we first show the length of the shortest alternating path strictly increases if we alternate a matching by a collection of blocking paths.

Lemma 6 *Let M be a matching and $P_1, \ldots, P_k$ be a blocking collection of paths. Let $M' = (M - \cup_i P_i) \cup (\cup_i P_i - M)$. Then $\alpha_{M'} \geq \alpha_M + 1$.*

Proof. Suppose otherwise and let Q be an M'-alternating path with $|Q| \leq \alpha_M$. Say Q follows vertices $v_1, v_2, \ldots, v_k$ where v_1, v_k are M'-exposed. Note that v_1 and v_k are then also M-exposed as every vertex lying on some P_i is matched in M'.

Consider the directed graph $\mathcal{H}$ with edges formed by taking the union of the directed edges (from $\overrightarrow{\mathcal{G}_M}$) lying on the paths P_i and the directed edges (from $\overrightarrow{\mathcal{G}_{M'}}$) of Q. If there are vertices u, v such that uv and vu are both edges of $\mathcal{H}$ then remove both copies. Let the resulting set of directed edges be F. Now, any edge uv of Q that is not an edge of $\overrightarrow{\mathcal{G}_M}$ must have that vu was on some P_i. So all edges of F are also edges in $\overrightarrow{G_M}$.

Let S be the set of $b+1$ vertices at the start of Q or some P_i and let T be the set of $b+1$ vertices at the end of Q or some P_i. By construction

- $\delta^{in}_{\mathcal{H}}(v) = \delta^{out}_{\mathcal{H}}(v)$ for each $v \in V - (S \cup T)$,
- $\delta^{in}_{\mathcal{H}}(v) = 0, \delta^{out}_{\mathcal{H}}(v) = 1$ for each $v \in S$, and
- $\delta^{in}_{\mathcal{H}}(w) = 1, \delta^{out}_{\mathcal{H}}(w) = 0$ for each $w \in T$.

We can decompose $\mathcal{H}$ into $b+1$ edge-disjoint paths $\overrightarrow{\mathcal{G}_M}$, each of which corresponds to an M-alternating path. To see this, pick any vertex in S, arbitrarily walk until some end vertex in T, remove the edges of the path, and repeat until all start vertices are used. The condition $\delta^{in}(v) = \delta^{out}(v)$ for every $v \in V - (S \cup T)$ guarantees this will succeed.

The total number of edges in these $b+1$ paths is at most

$$\sum_{i=1}^{b} |P_i| + |Q| - 2|\{uv \in Q : vu \in \cup_i P_i\}| \leq (b+1)\alpha_M - 2|\{uv \in Q : vu \in \cup_i P_i\}|.$$

If the latter term is not 0, then the total number of edges on these $b+1$ paths is strictly less than $(b+1)\alpha_M$ meaning some M-alternating path has length strictly less than α_M, a contradiction.

So the latter term is 0 meaning Q is in also an M-alternating path (in addition to being an M'-alternating path). Since it is a shortest M-alternating path (by assumption $|Q| \leq \alpha_M$) and since the P_i are a blocking collection, there is some v_j lying on both Q and some P_i.

Note that $j \neq 1, k$ because v_1, v_k are M'-exposed so they do not lie on any P_i. If $v_j \in L$ then $v_{j-1}v_j$ is the only edge entering v_j in $\overrightarrow{\mathcal{G}_M}$ so $v_{j-1}v_j \in Q \cap P_i$. On the other hand, if $v_j \in R$ then $v_j v_{j+1}$ is the only edge exiting v_j in $\overrightarrow{\mathcal{G}_M}$ and we see $v_j v_{j+1} \in Q \cap P_i$. This yields a contradiction: the path Q is both M-alternating and M'-alternating so none of the directed edges it traverses can by traversed by some P_i (as the edge would be directed different ways in $\overrightarrow{\mathcal{G}_M}$ and $\overrightarrow{\mathcal{G}_{M'}}$).

In all cases considered, we arrived at a contradiction so it must be that $\alpha_{M'} \geq \alpha_M + 1$. ∎

Finally, we bound the number of iterations.

Lemma 7 *The number of iterations of the loop in Algorithm 3 is at most* $2\sqrt{n} + 1$.

Proof. Since α_M strictly increases in each iteration, then after $\sqrt{n}$ iterations we have $\alpha_M \geq \sqrt{n}$. Let M^* be a maximum matching and M the matching after $\sqrt{n}$ iterations. Consider the (multi) graph $\mathcal{H} = (G; M^* \dot{\cup} M)$ constructed in the proof of Lemma 2. Let $\mathcal{C}$ be the cycles in $\mathcal{H}$ and $\mathcal{P}$ the paths in $\mathcal{H}$.

For each $P \in \mathcal{P}$, if P begins and ends with M^*-exposed vertices then it is an M^*-alternating path, contradicting the fact that M^* is a maximum matching. So we write $\mathcal{P} = \mathcal{P}_0 \cup \mathcal{P}_1$ where $\mathcal{P}_0$ are the paths with one endpoint being M-exposed and the other being M^*-exposed and $\mathcal{P}_1$ are the paths with both endpoints being M-exposed. Then

$$\begin{aligned} |M^*| - |M| &= \sum_{C \in \mathcal{C}} (|M^* \cap C| - |M \cap C|) + \sum_{P \in \mathcal{P}_0} (|M^* \cap P| - |M \cap P|) + \sum_{Q \in \mathcal{P}_1} (|M^* \cap Q| - |M \cap Q|) \\ &= 0 + 0 + |\mathcal{P}_1| \end{aligned}$$

Every path in $\mathcal{P}_1$ has length $\geq \alpha_M \geq \sqrt{n}$, so

$$|\mathcal{P}_1| \leq \frac{|M^*| + |M|}{\sqrt{n}} \leq \frac{n/2 + n/2}{\sqrt{n}} = \sqrt{n}.$$

Since each iteration is guaranteed to find at least one alternating path (provided there is one), then in at most $|\mathcal{P}_1| + 1 \leq \sqrt{n} + 1$ further iterations we will find that there is no further alternating path. The total number of iterations is then at most $2\sqrt{n} + 1$. ∎

1.5.1 Finding Blocking Paths

We can find a collection of M-blocking paths in linear time.

1. Compute the shortest $s - v$ path length $\text{dist}_M(v)$ in $\overrightarrow{\mathcal{G}_M}$ in $O(m)$ time using a breadth-first search. If $\text{dist}_M(t) = \infty$ (i.e. is unreachable from S) then there is no M-alternating path and M is a maximum matching and we quit. Note $\alpha_M = \text{dist}_M(t) - 2$.
2. Otherwise, let $\mathcal{I} = (V'; E')$ be the following directed graph with V' being the disjoint union of $V_i = \{v : \text{dist}_M(v) = i\}$ and edges uv from $\overrightarrow{\mathcal{G}_M}$ with $\text{dist}_M(v) = \text{dist}_M(u) + 1$. Note that $\mathcal{I}$ is a directed and acyclic graph.
3. Perform a DFS from each exposed node in L with the following modification. If an exposed node $v \in R$ is reached then add the current $s - v$ path (excluding s) to the collection of blocking paths and restart a DFS from the next exposed node (remembering which nodes have already been visited).

Algorithm 4 Finding Blocking Paths

Input: Undirected bipartite graph $\mathcal{G} = (L \cup R, E)$, matching M
Output: A collection of M-blocking paths or NONE

```
compute dist_M(v) for every v ∈ V ∪ {s,t} in G_M→ using a breadth-first search
if dist_M(t) = +∞ (i.e. t not reached) then
  return NONE
end if
construct I as described above
P = ∅ {the blocking collection of paths}
seen[v] ← false for each v ∈ V
for each exposed v ∈ L do
  if DFS-Blocking(I, v, {u ∈ R : u is M-exposed}, seen) returns a path then
    P ← P ∪ {P}
  end if
end for
return P
```

Lemma 8 *Algorithm 4 finds a collection of blocking paths in $O(m)$ time.*

Proof. The BFS to compute distances takes $O(m)$ time as does constructing $\mathcal{I}$. The DFS will never expand a vertex more than once, so the total running time is $O(m)$.

Any path discovered by the DFS is a shortest M-alternating path because any path in $\mathcal{I}$ ending at an exposed vertex must be an shortest M-alternating path (by construction). The paths returned are vertex-disjoint because

Algorithm 5 DFS-Blocking

Input: Directed graph $\mathcal{I}$, vertex v, target set of nodes T, boolean table *seen*
Output: A path or NONE

```
if seen[v] = true then
  return NONE
end if
seen[v] ← true
if v ∈ T then
  return trivial path starting and ending at v
end if
for each edge vw in I do
  if DFS-Blocking(I, w, T, seen) returns a path P then
    prepend edge vw to P
    return P
  end if
end for
return NONE
```

the DFS never expands a vertex more than once. Suppose otherwise, and that Q is a shortest M-alternating path that is disjoint from paths in $\mathcal{P}$. Then Q is a path between exposed vertices in $\mathcal{I}$. Note the start node v_0 of Q has $seen(v_0) = \textbf{true}$ because DFS-Blocking was called with v_0 from Algorithm 4. Let v be the node of P that had $seen[v]$ marked **true** first in the algorithm.

At this point of the algorithm, there was a path from v to the endpoint of Q consisting vertices u with $seen[u] =$ **false**. But since v does not lie on any path in $\mathcal{P}$ we know that the recursive call from v failed to find such a path. This is a contradiction. ∎

Summary
We can find a collection of blocking paths in $O(m)$ time. The procedure that iterates finding a collection of blocking paths and alternating the current matching with this collection runs for $O(\sqrt{n})$ iterations. So in $O(\sqrt{n} \cdot m)$ time we find a maximum matching.