1 Definitions

**Matching** A matching $M$ of a graph $G = (V, E)$ is a subset of edges with the property that no two edges of $M$ share the same node, i.e., a set of pairwise non-adjacent edges. The **matching problem** is to find a maximum matching: the largest possible matching in a graph with $V$ vertices is $|V|/2$, called a **complete** or **perfect** matching.

**Matched edges** Refer to the edges in a matching $M$.

**Free edges** Edges in $E$ that are not in $M$.

**Mate** If $[u, v]$ is a matched edge, then $u$ is the mate of $v$.

**Exposed vertices** Vertices that are not incident upon any matched edges.

**Alternating path** A sequence of edges $p = [u_1, u_2, ...u_k]$ where $[u_1, u_2], [u_3, u_4], ...[u_{2j-1}, u_{2j}], ...$ are free, whereas $[u_2, u_3], [u_4, u_5], ...[u_{2j}, u_{2j+1}], ...$ are matched.

**Outer vertices** Those vertices on an alternating path that start with an exposed vertex and have an odd rank (position) along the path. For example, from Figure 17, alternating path $p_1 = [v_1, v_4, v_5, v_6, v_8, v_7, v_10, v_9]$ of Graph $G$ starts at exposed vertex $v_1$ and has outer vertices $v_1, v_5, v_8, v_{10}$. Conversely, **inner vertices** satisfy the same criteria but have an even rank in the path.

**Augmenting Path** An alternating path $[u_1, u_2, ...u_k]$ where $u_1$ and $u_k$ are exposed vertices. Figure 17 has an augmenting path $p_2 = [v_1, v_4, v_5, v_6, v_8, v_7, v_{10}, v_9]$ (highlighted in blue).

2 The Matching Problem

From these definitions, we can discuss an approach for finding a maximum matching. Firstly, the importance of augmenting paths is captured by the following lemma.

**Lemma 2.1.** Let $P$ be the set of edges on an augmenting path $p = [u_1, u_2, ...u_{2k}]$ in a graph $G$ with respect to the matching $M$. Then $M' = M \oplus P = (M - P) \cup (P - M)$ is a matching of cardinality $|M| + 1$.

**Proof** To begin, show that no two edges of $M \oplus P$ share the same node of $G$. Let’s say by contradiction that two edges $e$ and $e'$ in $M \oplus P$ are incident upon the same node. Three cases follow:

1. $e, e' \in M - P$
2. $e, e' \in P - M$
3. $e \in M - P, e' \in P - M$
In case 1, there are two edges in $M$ sharing the same node, contradicting the definition of a matching. In case 2, $P - M$ are the (unmatched) edges of the form $[u_{2j-1}, u_{2j}]$, so two of them cannot be incident upon the same node. For case 3, suppose that an edge $e' = [u_{2j-1}, u_{2j}] \in P - M$ shares node $u_{2j}$ with $e \in M - P$. But since $u_{2j}$ is also a node of edge $e'' = [u_{2j}, u_{2j+1}]$, this would mean again that two edges of $M$ share a common node, a contradiction. Therefore, no two edges in $M' = M \oplus P$ share the same node, making it too a matching. As well, $P$ contains $2k - 1$ edges where $k$ of them are free and $k - 1$ are matched. Since $M'$ contains $k$ edges, $|M'| = |M| + 1$ edges.

Figure 1: For a graph $G$ we see augmenting path $p$ (where $v_1$ and $v_9$ are outer vertices) highlighted in blue and matching $M$ highlighted in red.

Figure 2: Same graph and path $p_2$ with and matching $M'$ of cardinality $|M| + 1$

From Figure 17 and 18 we see an example of when given a matching $M$ with respect to an alternating path $p$, we can find another matching $M'$ of greater cardinality. In fact, $M'$ is a maximum matching because $|M'| = |V|/2$. The relationship between augmenting paths and the maximum matching is as follows . . .

**Theorem 2.2.** (Berge’s Theorem) A matching $M$ in a graph $G$ is maximum iff there is no augmenting path in $G$ with respect to $M$.

**Proof** Firstly, there can be no augmenting paths with respect to a maximum matching since such a path could be used to augment the matching by Lemma 2.1, which contradicts the definition of maximum. As well, we prove the converse by contradiction: say there is no augmenting path in $G$ with respect to $M$, but our matching $M$ is not maximum. So, $\exists$ another matching $M'$ where $|M'| > |M|$. Because two edges of a matching cannot be incident on the same vertex, the subgraph $G' = (V, M \oplus M')$ has the property that all its edges have degree $\leq 2$. It follows that a degree two vertex in $G'$ has one edge in $M$ and the other in $M'$, and that all connected components of $G'$ will be either paths of circuits of even length. In all circuits we have the same number of edges in $M$ as we do in $M'$. Therefore $|M'|$ being greater than $M$ means that there must be some path $p$ with more edges in $M'$ than in $M$, and so $p$ is an augmenting path with respect to $M$ - a contradiction. 


The basic idea behind every Matching problem algorithm is to start with any matching, and repeatedly discover augmenting paths until there aren’t any. Even still, there are inobvious complexities in the implementation of this approach. As such, we start with a base example of matchings of Bipartite Graphs, to build a foundation for for understanding the general case.

An Equivalent Problem

The Assignment Problem: Some jobs are available to be filled. Given a group of applicants for these jobs, fill as positions as possible, employing people in only positions for which they are qualified. This situation can be represented by means of a bipartite graph $B = (V, U, E)$ in which $V$ represents the set of applicants, $U$ the set of jobs, and an edge $e = (v, u) \in E$ with $v \in V$ and $u \in U$ signifies that applicant $v$ is qualified for job $u$. An assignment of applicants to jobs, one person per job, corresponds to a matching in $M$ of $G$, and a maximum matching for $G$ solves the problem of filling as many vacancies as possible.

A Bipartite Matching Algorithm

Just for refresher’s sake, we define a Bipartite Graph $B = (V, U, E)$ to consist of two disjoint sets of vertices $V, U$ such that every edge connects a vertex in $V$ to one in $U$. Clearly, a search for augmenting paths must begin by finding alternating paths from exposed vertices in $V$ where all incident edges are free. By definition, the augmenting path must begin with an exposed vertex in $V$ and end with an exposed vertex in $U$. From Figure 3, we start with $v_2$, and do a breadth first search for all possible alternating paths, looking first at adjacent vertices $u_2$ and $u_6$. In the spirit of alternation, we then only consider those edges emanating from $u_2$ and $u_6$ that are matched, of which there should be only one. Since $u_2$ and $u_6$ are not exposed, we have not yet found an augmenting path, and must continue growing alternating paths from mates $v_3$ and $v_5$. Since the node $u_3$ is reached from $v_3$ before its reached from $v_5$, we ignore the edge $[v_5, u_3]$ in order to avoid consideration of overlapping ultimately redundant alternating paths. Also, observe that we do not have to traverse the vertices at odd levels of the graph (where $v_2$ is at level 0), since we know in this case that the next relevant vertex is always the mate of the current vertex. The problem thus reduces to searching a digraph $(V, A)$ where $(v_1, v_2) \in A$ if $v_1$ is adjacent to the mate of $v_2$.

![Figure 3: A bipartite graph B with exposed nodes in green and current matching M in red](image)

![Figure 4: An auxiliary digraph A of B such that exposed[v_1] \neq 0](image)

In the algorithm, array mate stores the matching, array label is used for searching the digraph (and specifically tracking the parent of the current node), and exposed[v] stores a node of $U$ that is exposed and adjacent to $v$. While BFS-traversing the digraph $A$, we have found a matching when exposed[v] \neq 0. At this point we recursively augment the path moving up the digraph until we reach the root; that is, for the current node $x$, label[x] = 0. At this point, if there are no more augmenting paths, the matching is maximum.
Figure 5: A bipartite graph $B$ with no exposed nodes and matching $M'$ in red, where $M' = |V|/2$. From 5, we first call $\text{Augment}(v_1)$, changing the matching of $v_1$ from $u_4$ to $u_1$. Then, call $\text{Augment}(v_3)$, changing the matching of $v_1$ from $u_6$ to $u_4$. Lastly, call $\text{Augment}(v_2)$, where $v_2$ is the root of $A$. Node $v_2$ is matched to $u_1$, and the algorithm halts.

**Theorem 2.3.** The bipartite matching algorithm computes the maximum matching for $B = (V, U, E)$ in $O(\min(|V|, |U|) \cdot E)$ time.

**Proof** A matching in $B$ can have at most $\min(|V|, |U|)$ edges. As each augmentation increases the matching cardinality by one, we have at most $\min(|V|, |U|)$ stages. Building the auxiliary graph and exposed array takes $O(|E|)$. For each stage, finding the directed path in $A$ takes $O(|A|) = O(|E|)$. Lastly, the augmented matching requires $O(|V|)$ time to compute.

**Algorithm 1: Augment(v)**

```
if label[v] == 0 then
    // label[v] is used for search, is 0 if v is the root, or otherwise contains the parent of v in the auxiliary digraph
    mate[v] = exposed[v] // exposed[v] is a node of B.U that is exposed and adjacent to B.V
    mate[exposed[v]] = v;
else
    exposed[label[v]] = mate[v];
    mate[v] = exposed[v];
    mate[exposed[v]] = v;
    Augment(label[v]);
```
### Algorithm 2: Bipartite Matching Algorithm($B$)

**Result:** The maximum matching of $B$ contained in the array $mate$

1. for $\forall v \in B.V \cup B.U$ do $mate[v]=0$
2. (stage begin);
3. for $\forall v \in B.V$ do $mate[v]=0$
4. $A = \text{Null}$;

   // begin construction of the helper digraph $(V,A)$
5. for $\forall (v,u) \in E$ do
6.   if $mate[u]=0$ then
7.     $\text{exposed}[v] = u$
8.   else
9.     if $mate[u] \neq v$ then $A = A \cup \{v, mate[u]\}$
10. $Q = \text{Null}$;
11. for $\forall v \in V$ do
12.   if $mate[v]=0$ then
13.     $Q = Q \cup \{v\}$
14.     $\text{label}[v] = 0$
15. while $Q \neq \text{Null}$ do
16.   $v = Q\text{.first}$;
17.   $Q = Q - \{v\}$;
18.   if $\text{exposed}[v] \neq 0$ then
19.     $\text{AUGMENT}(v)$;
20.     jump to stage
21. else
22.   for $\forall$ unlabeled $v'$ such that $(v, v') \in A$ do
23.     $\text{label}[v'] = v$
24.     $Q = Q \cup \{v'\}$

### Nonbipartite Matching

There are two key assumptions that no longer hold when we try to extend the bipartite matching solution to general graphs. Unlike the bipartite case...

1. It is possible for vertices of $U$ to be outer (i.e., to have an odd rank on an alternating path that starts with an exposed vertex). As a result, our auxiliary graph can no longer just include vertices of $V$, but in fact all vertices.

2. There may be paths in the auxiliary graph leading from exposed nodes to target nodes that do not correspond to augmenting paths relative to the current matching.
The only missing feature in bipartite graphs is the existence of odd circuits (edge disjoint cycles). In particular, the culprit circuits are those that contain the most possible matched edges - that is, circuits with $2k + 1$ nodes and $k$ matched edges. These are called blossoms. We know we have found a blossom when all of its vertices are either outer or mates of outer vertices. A flower with respect to a matching $M$ is composed of a stem, which is an alternating path of even length from an exposed vertex $u$ to a blossom.

**Lemma 2.4.** If we discover a blossom $b$ while searching for an augmenting path from an exposed node $u$ in a graph $G$ with respect to a matching $M$, then there is an alternating path from $u$ to any node of $b$ ending with a matched edge.

**Proof** Since $b$ was discovered from $u$, there is an alternating path from $u$ to the basis of the blossom $u_0$. Therefore, $u_0$ is outer, and when $b$ is shrunk the resulting vertex $v_b$ is outer as well. Further, when $b$ is shrunk, all outgoing edges from the blossom on an augmenting path are edges coming off of outer vertices. Thus, vertices connected to the root (initial exposed vertex of the path) through the blossom all undergo an even-length reduction in their distance from the root.

**Theorem 2.6.** Suppose that, while searching for an augmenting path from a node $u$ in an undirected graph $G$ with respect to a matching $M$, we discover a blossom $b$. Then there is an augmenting path from $u$ in $G$ with respect to $M$ if there is one from $u$ in $G/b$ with respect to $M/b$.

**Proof** Let $p$ be an augmenting path in $G/b$ wrt $M/b$. If $p$ does not pass through $b$, then $p$ itself is an augmenting path in $G$. Otherwise there are two cases:

**Case 1:** $p$ (or ends) begins at $b$

Say edge $(b, w')$ is the first edge along $p$. We infer from the notation that node $w'$ is outside of blossom $b$, and thus edge $(b, w) \notin M/b$. Moreover, there is a vertex $u_i \in b$ such that $(u_i, w')$ is an unmatched edge of $G$. In $b$, we should be able to find an even length alternating path $p'$ from basis node $u_0$ to $u_i$ ending with a matched edge. Where $p''$ is is the continuation of path $p$ after blossom $b$, the path $p_G = [p', u_i, w', p'']$ is an augmenting path in $G$ wrt $M$. 

**Figure 6:** $q' = (v_1, v_8, v_6, v_5, v_7, v_9)$ in auxillary graph $A$ of $G$ leads from an exposed node to a target node

**Figure 7:** $q = [v_1, v_9, v_8, v_7, v_6, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}]$ (in blue) is neither an augmenting path, nor even a path. As well, $M \oplus Q$, where $Q$ contains edges of walk $q$, is not a matching.
Case 2: \( b \) is not the first or last vertex on \( p \)

Let edge \((w, b) \in M/b\) and \((b, w') \notin M/b\) be in \( p \). Let \( p' \) be the subpath up to \( w \) and \( p'' \) be the subpath after \( w' \). The edge \((w, u_0)\) must be a matched edge in \( G \), as the only edge that enters a blossom does so at the basis vertex. There is a vertex \( u_i \in b \) such that \((u_i, w')\) is an unmatched edge of \( G \). Again, we find an even length alternating path \( p'' \) from \( u_0 \) to \( u_i \). The path

\[
G = [p', w, p'', w', p''']
\]

is an augmenting path in \( G \) wrt \( M \).

**Only If** Let \( u_0 \) be the basis of the blossom \( b \). There are again two cases

Case 1: An unmatched edge enters basis \( u_0 \)

Let \( p \) be an augmenting path in \( G \) wrt \( M \). We consider when \( p \) passes through \( b \), as otherwise \( p \) is an augmenting path wrt \( M/b \) in \( G/b \). Since \( p \) begins and ends at exposed vertices, and only one vertex of \( b \) is exposed, we know \( p \) does not start on \( b \). Let \( p' \) be the subpath of \( p \) before its encounter with blossom \( b \). Say \( p' = [p'', w, u_0, ..., u_i] \) is an augmenting path with respect to \( M/b \) in \( G/b \).

Case 2: A matched edge enters basis \( u_0 \)

Let \( M' = M \oplus S \) where \( S \) is a stem of the flower with blossom \( b \). Since \( S \) by definition is an even length alternating path starting at an exposed vertex, we know that \(|M'| = |M|\). The symmetric difference operation ensures that \( b \) is a blossom wrt \( M' \) and the edge entering basis \( u_0 \) is unmatched wrt \( M' \). Furthermore, \(|M'/b| = |M/b|\). Because \( M' \) and \( M \) have the same cardinality and there is an augmenting path wrt \( M \), then there must be an augmenting path wrt \( M' \). Since the edge entering \( u_0 \) is unmatched in \( M' \), Case 1 asserts that there is an augmenting path wrt \( M'/b \) in \( G/b \), and as \( M/b \) and \( M'/b \) are again of the same cardinality, there must also be an augmenting path wrt \( M/b \) in \( G/b \).

**Theorem 2.7.** If at some point for a node \( u \) in a Graph \( G \) there is no augmenting path with respect to a matching \( M \), then there will never be an augmenting path from \( u \).v (Proof left as exercise).

Finding augmenting paths in the nonbipartite case can thus be achieved as follows: Start by constructing alternating paths from the exposed vertices of \( G \). Upon discovering a blossom \( b \), shrink \( b \). According to Theorem 2.6 and Lemma 2.4, this is a legitimate operation because it preserves the existence of augmenting paths from the node being considered. Meanwhile, from Lemma 2.5, we know that the shrunk vertex \( v_b \) is outer, so we now traverse all edges emanating from inner vertices inside \( b \).
The algorithm terminates either upon finding an augmenting path, or when all edges touching exposed vertices are explored.

Algorithm 3: Nonbipartite Matching Algorithm\( (B) \)

Result: The maximum matching of \( B \) contained in the array \( \text{mate} \)

\[
\begin{array}{ll}
\text{for } \forall v \in V \text{ do} & \text{mate}[v] = 0; \\
\text{considered}[v] = 0; & \text{while } \exists u \in V \text{ with considered}[u] = 0 \text{ and mate}[u] = 0 \text{ do} \\
\text{considered}[u] = 1; & A = \text{Null} \\
\text{forall } v \in V \text{ do } \text{exposed}[v] = 0; & \text{// construct auxiliary digraph} \\
\text{forall } [v, w] \in E \text{ do} & \text{if mate}[w] = 0 \text{ and } w \neq u \text{ then} \\
\text{exposed}[v] = w & \text{if mate}[w] \neq v \text{ and mate}[w] \neq 0 \text{ then} \\
\text{else} & A = A \cup \{(v, \text{mate}[w])\}; \\
\text{forall } v \in V \text{ do } \text{seen}[v] = 0; & \text{forall } v \in V \text{ such that } (v, w) \in A \text{ do} \\
\text{seen}[\text{mate}[w]] = 1; & \text{if exposed}[w] \neq 0 \text{ then } \text{AUGMENT}(u) \text{ jump to stage;} \\
\text{if seen}[w] = 1 \text{ then } \text{BLOSSOM}(w); & \text{while } Q \neq \text{Null} \text{ do} \\
& v = Q.\text{first}; \\
& Q = Q - \{v\}; \\
& \text{forall unlabelled nodes } w \in V \text{ such that } (v, w) \in A \text{ do} \\
& Q = Q \cup \{w\}; \\
& \text{label}[w] = v; \\
& \text{seen}[\text{mate}[w]] = 1; \\
& \text{if exposed}[w] \neq 0 \text{ then } \text{AUGMENT}(w) \text{ jump to stage;} \\
& \text{if seen}[w] = 1 \text{ then } \text{BLOSSOM}(w); \\
\end{array}
\]

**BLOSSOM**\( (b \in V) \) works as follows...

1. Backtrace the label array until we find the first common node in the divergent paths - known as the basis of \( b \). The nodes of \( b \) consist of all nodes on these two paths (aka the odd circuit) including matched edges. Array \( \text{blossom}[v] = b \) keeps track of the vertices \( v \in V \) that belong to \( b \). As well, we store the basis of \( b \) and the exact cyclic order of \( b \)'s vertices. Upon discovering \( b \), we also look for an exposed vertex \( u_j \in b \). At that point we have to augment the matching from \( v_b \), as detailed in the example below

2. To shrink \( b \), we replace any pointer to a vertex \( v \in b \) in the auxiliary graph \( A \), the BFS queue \( Q \) and the array \( \text{label} \) by the new node \( v_b \).

**Nonbipartite Example**

The first four iterations of the algorithm are trivial, as augmenting paths consisting of single edges are found immediately: \([v_2, v_3][v_4, v_5][v_6, v_7][v_8, v_9]\). In the next stage, we pick an exposed and unconsidered node \( v_{10} \), and construct the auxiliary digraph with respect to the current matching. The only node \( v \) with \( \text{exposed}[v] \neq 0 \) is \( v_2 \). Execute the following steps:

1. Remove \( v_{10} \) from \( Q \), insert \( v_8 \) into \( Q \), while the seen entry for \( v_9 \) is set to 1, because the mate of \( v_9 \) has been labeled. \( Q = \{v_8\} \)

2. Remove \( v_8 \) from \( Q \), insert \( v_6 \) into \( Q \), and set the seen entry for \( v_7 \) to 1, \( Q = \{v_6\} \)
3. Remove $v_6$ from $Q$, insert nodes $\{v_4, v_5\}$ into $Q$, and set the seen entry for $v_5$ to 1, as $v_5$ is the mate of $v_4$ $Q = \{v_4, v_5\}$.

4. At this point we examine $v_5$ and realize it was already seen in the previous step, ie $\text{seen}[v_5] = 1$. Thus we have found a blossom. To find all the vertices of the blossom, we trace the array label back from $v_5$ and its mate $v_4$. The first common node in these backwards paths is $v_6$, and so $v_6$ becomes the basis of the blossom. We give to this blossom a new label $v_b$, shrinking all nodes in the backtracing and their mates into $v_b$. All references to nodes of the blossom in auxiliary graph $A$ and queue $Q$ are replaced by $v_b$.

5. Within the BLOSSOM procedure we discover that $v_b$ is also part of a blossom. Because of reference replacement, $v_b$ will now be next up in $Q$. Remove $v_b$ from $Q$, insert nodes $\{v_2, v_3, v_7\}$. Then at this point $\text{seen}([\text{mate}(v_2)]) = \text{seen}[v_3] = 1$. At this point in the same for loop we realize that we have seen $v_3$ already, and we have another blossom with basis $v_b$. Shrink all nodes of this second blossom into node $v_{b1}$, and update relevant references to blossom nodes in $A$ and $Q$ as in the previous step.

6. Now we find the node $u_j$ of blossom $v_{b1}$ such that $\text{exposed}[u_j] \neq 0$. This is node $v_2$. Thus we augment the matching from $v_{b1}$.
Figure 15: From $p = [v_{10}, v_9, v_8, v_7, v_{b1}, v_1]$ we must find a unique alternating subpath that enters $v_{b1}$ through a matched edge and exits on a free edge.

Figure 16: The unique path in $v_{b1}$ is $\{v_6, v_3, v_2\}$. Substituting into $p$, we get $[v_{10}, v_9, v_8, v_7, v_6, v_3, v_2, v_1]$. Then, we must find the unique alternating subpath in blossom $v_b$ that enters through a matched edge and exits through a free edge.

Figure 17: The unique path in $v_b$ is $\{v_6, v_5, v_4\}$. Our final augmenting path is $p = [v_{10}, v_9, v_8, v_7, v_6, v_5, v_4, v_3, v_2, v_1]$.

Figure 18: $M' = M \oplus P$ is a maximum matching of cardinality $|V|/2$. There are no more augmenting paths from exposed vertices, and thus we are done.

Performance Analysis

Theorem 2.8. The above algorithm correctly finds a maximum matching in a graph $G$ in $O(|V|^4)$

Proof: The algorithm halts when it fails to find augmenting paths from exposed nodes of $G$, and thus matching $M$ is maximum. For the time bound . . .

1) Loop stage is executed at most $|V|$ times (once for each vertex), and each stage can be executed in $O(|V|^3)$ time.

2) Construction of the auxiliary digraph and the exposed array can be done in $O(|V|^2)$.

3) The Blossom procedure will be executed at most $|V|$ times, as each execution decreases the number of vertices by at least 2. Each call to blossom takes at most $O(|V|^2)$ time.

   (a) Updating A can be done in $O(|A|) = O(|E|) = O(|V|^2)$ time. The label array and $Q$ can be updated in $O(|V|)$ time.

4) Augment is executed once for each of the $|V|$ searches, and takes $O(|V|^3)$ each time

   (a) We may have to make at most $O(|V|)$ blossom expansions, as there are at most that many blossoms

   (b) Each expansion amounts to first finding the node $u_j$ (where exposed$[u_j] = 0$) of the blossom that is connected to a node $w$ in the original graph, by looking at all points shrunk into $b$- and $w$, if it is also a blossom. Secondly, we find a combination of nodes that is an edge in $E$. 
i. Note that \( \text{blossom}[v] = b \) if \( v \) belongs to blossom \( b \). Find \((u_j, w)\) in \( O(|V|^2) \) by starting with each node \( v \) and finding \( \text{blossom}[v] \) and \( \text{blossom}[\text{blossom}[v]] \) until it returns \( b \) or \( w \).

ii. Part 2 can be done in \( O(|E|) \) by checking for each edge \([v, u] \in E\) whether \( v \) is essentially shrunk into \( b \) and \( u \) is eventually shrunk into \( w \). The rest of AUGMENT can be done in \( O(|V|) \) time for each blossom expansion.

Exercises

1) Prove Theorem 2.7: If at some point for a node \( u \) in a Graph \( G \) there is no augmenting path with respect to a matching \( M \), then there will never be an augmenting path from \( u, v \) (Proof left as exercise).

2) Show that in a bipartite graph, the cardinality of the maximum matching equals the cardinality of the smallest set of nodes that covers all edges (that is, every edge is incident upon a node in the set).

3) Show that a bipartite graph \( B = (V, U, E) \) with \(|V| = |U| = n\) has a matching of cardinality \( n \) iff for all \( S \subseteq V \) we have \(|\{u \in U : [v, u] \in E \text{ for some } v \in S\}| \geq |S|\).

4) Show by an example that if \( G = (V, E) \) is a graph, \( M \) is a matching of \( G \), and \( b \) a blossom (a circuit with \( 2m + 1 \) edges, \( m \) of which are in \( M \)), then the following may fail to be true: there is an augmenting path in \( G \) wrt \( M \) iff there is one in \( G/b \) wrt \( M/b \). Compare with Theorem 2.6.

5) Let \( M \) be a perfect matching in a graph \( G \), all of whose vertices are of odd degree. Show that \( M \) includes every cut edge of \( G \).

6) Let \( M \) and \( N \) be matchings of a graph \( G \), where \(|M| > |N|\). Show that there are disjoint matchings \( M' \) and \( N' \) of \( G \) such that \(|M'| = |M| - 1\), \(|N'| = |N| + 1\) and \( M' \cup N' = M \cup N \).

7) A factory has \( n \) jobs 1, 2, ..., \( n \) to be processed, each requiring one day of processing time. There are two machines available. One can handle one job at a time and process it in one day, whereas the other can process two jobs simultaneously and complete them both in one day. The jobs are subject to precedence constraints represented by a \(<\) symbol, where \( i < j \) signifies that job \( i \) must be completed before job \( j \) is started. The objective is to complete all the jobs while minimizing \( d_1 + d_2 \), where \( d_i \) is the number of days during which which machine \( i \) is in use. Formulate this problem as one of finding a maximum matching in a suitably defined graph.
References


