U.C. Berkeley
CS294-5: Great Algorithms
Professor Richard Karp

February 1, 2006
Scribe: Brian Green

Edmonds's Non-Bipartite Matching Algorithm

Definition 1 (matching) *A* **matching** *in a graph is a set of edges, no two of which meet at a common vertex. A* **maximum matching** *is a matching of maximum cardinality.*

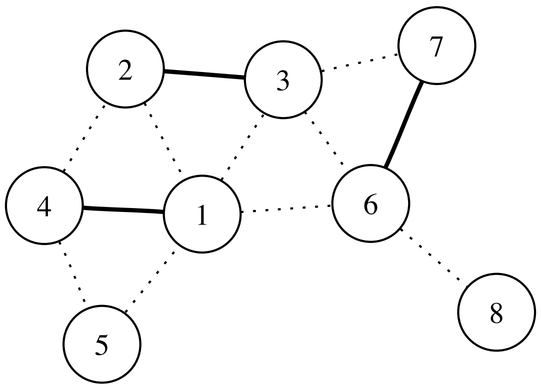


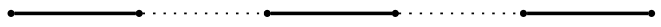
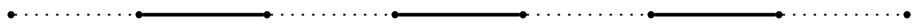
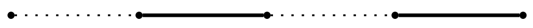
Figure 1. The three solid edges form a matching.

Definition 2 (free vertex) *A vertex is* **free** *with respect to a matching M if it is not incident with any edge in M.*

In the example above, vertices 5 and 8 are free vertices.

Definition 3 (alternating path) *A path is* **alternating** *with respect to a matching M if its edges are alternately in M and not in M.*

There are three types of alternating paths:

1. An alternating path between two matched vertices:

2. An alternating path between two free vertices:

3. An alternating path between a matched vertex and a free vertex:


Definition 4 (augmenting path) *An* **augmenting path** *is a simple alternating path between free vertices.*

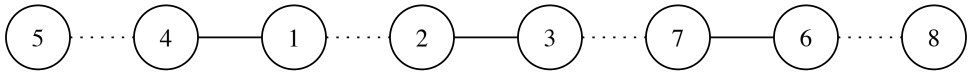


Figure 2. An augmenting path from the above graph.

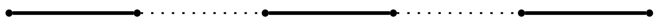
Theorem 5 *M is* **not** *a maximum matching if and only if there exists an augmenting path with respect to M.*

Proof:

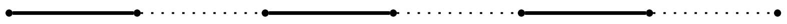
($\Leftarrow$) If P is an augmenting path with respect to the matching M, then the symmetric difference $M \oplus P$ [1] is a matching, and $|M \oplus P| = |M| + 1$.

($\Rightarrow$) Let M be a non-maximum matching and N be a maximum matching. Consider the symmetric difference of the matchings, $M \oplus N$. Since no two edges in a matching meet at a common vertex, every vertex is incident with at most one vertex in M and one vertex in N. Representing N by dotted edges and M by solid edges, each connected component of $M \oplus N$ is one of the following types:

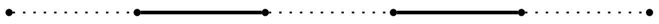
1. A path that begins and ends with a solid edge:



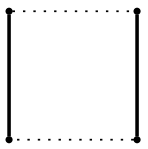
2. A path that begins with a solid edge and ends with a dotted edge:



3. A path that begins and ends with a dotted edge:



4. An alternating cycle such as:



Since N is a maximum matching, |N| > |M|, and there must be a component with more dotted edges than solid ones. Also, any alternating cycle must have an even length because every vertex is incident with at most one edge from N and one edge from M. Thus some connected component must be of type 3 – an augmenting path. $\square$

As a result of this theorem, we now give an algorithm to find a maximum matching.

[1]The symmetric difference $M \oplus P$ is the set of edges $\{e : e \in M \text{ or } e \in P \text{ and } e \notin M \cap P\}$.

```
Maximum-Matching ($G$)
  $M \leftarrow \emptyset$
  **repeat**
    **if** there is an augmenting path with respect to $M$
      **then**
        Let $P$ be an augmenting path with respect to $M$
        $M \leftarrow M \oplus P$
      **else**
        **return** $M$ and **halt**.
```

A special case: Bipartite Graphs

Definition 6 (bipartite graph) *A graph $G = (V, E)$ is* **bipartite** *if V can be partitioned into two disjoint sets, V_1 and V_2, such that every edge of G joins a vertex in V_1 and a vertex in V_2. Equivalently, G is bipartite if and only if every cycle in G is of even length.*

Terminology We use the convention that the vertices in V_1 are called *boys*, and the vertices in V_2 are called *girls*.

The above theorem has simplified the problem of finding a maximum matching to finding an augmenting path. In a bipartite graph, we search for an augmenting path by performing a breadth first search starting from the free boys that is modified to only follow simple alternating paths. In order to ensure that the breadth first search produces a simple alternating path, we enforce the following rule: if the set of vertices encountered in the ith step of the search, L_i, consists of boys then L_{i+1} is the set of girls adjacent to the boys in L_i who do not appear earlier in the search. If L_i consists of girls, then L_{i+i} is the set of boys matched with girls in L_i.

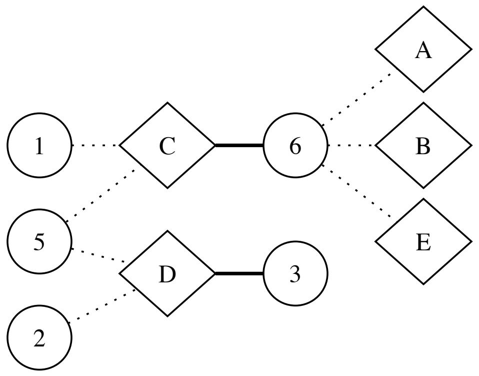
Theorem 7 *If a free girl is reached, then an augmenting path ending at that girl can be constructed. If no free girl is reached, then the current matching is maximum.*

In the example of this algorithm, we represent a bipartite graph by a matrix in which the rows correspond to the boys, the columns correspond to the girls, and an edge between boy i and girl j is represented by a 1 in the (i, j) position. A matching between boy i and girl j is indicated by circling the (i, j) entry in the matrix.

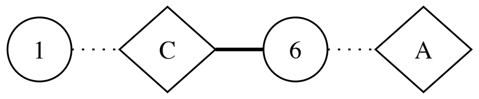
	A	B	C	D	E
1			1		
2				1	
3			1	①	
4	1	1		1	①
5			1	1	
6	1	1	①	1	1

Figure 3. Our initial matching.

We begin the algorithm by running our modified breadth first search from the free boys, $\{1, 2, 5\}$. This search produces the following graph:



Vertex 1 is a free boy, and vertex A is a free girl, so we have found the augmenting path:

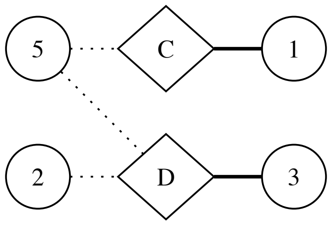


We invert the matchings in the path, matching 1 to C and 6 to A, and now have the following set of matchings:

	A	B	C	D	E
1			①		
2				1	
3			1	①	
4	1	1		1	①
5			1	1	
6	①	1	1	1	1

Figure 4. Set of matchings after one iteration
of the Maximum-Match algorithm

Running the modified breadth first search again from the set of free boys $\{2, 5\}$, we produce the graph:



The search has failed to find a free girl, and so there is no augmenting path remaining in the graph. Thus we have found a maximum matching. It is interesting to note that four of the vertices in the matching, $\{4, 6, C, D\}$, form a minimal vertex cover of this graph. In fact, the size of the maximum matching and minimum vertex cover are related.

Definition 8 (vertex cover) *A set of vertices X in a graph G form a vertex cover if every edge in G is incident with a vertex in X.*

Remark *If X is a vertex cover and M is a matching, then $|M| \leq |X|$.*

PROOF: Each edge of M is incident with a vertex in X, and each vertex in X is incident with at most one edge in M. □

Theorem 9 (*König – Egerváry*) *In a bipartite graph G, the maximum size of a matching is equal to the minimum size of a vertex cover.*

PROOF: Let M be a maximum matching. It suffices to exhibit a vertex cover X such that $|M| \geq |X|$. Partition the vertices into three parts:

1. Vertices occurring in breadth-first search from the free boys, as described above.
2. Vertices occurring in breadth-first search from the free girls, similarly defined.
3. The remaining vertices.

The vertices in parts 1 and 2 are disjoint, otherwise there would exist an augmenting path from a free boy in part 1 to a free girl in part 2. Also, the vertices in part 3 must all be matched, and they must all be matched with other vertices in part 3. If this were not the case, then they would appear in searches from the free boys or free girls.

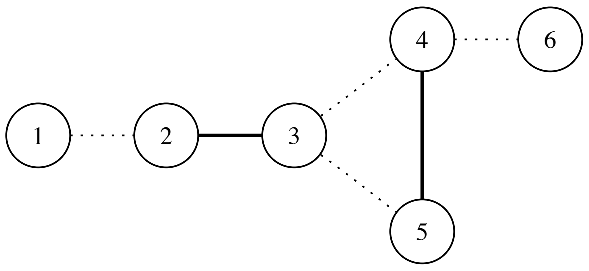
The girls in part 1 together with the boys in parts 2 and 3 form a vertex cover X, and each vertex in X is incident with a different edge of M, so $|M| \geq |X|$. □

Non-Bipartite Matching

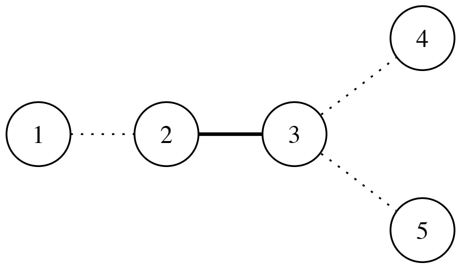
The modified breadth first search that we used for the previous graph depended on the fact that bipartite graphs do not have odd length cycles. This property ensures that we can safely ignore any cycles encountered during the search without excluding any augmenting paths. In a non-bipartite graph, we cannot ignore cycles in the graph, as the next example demonstrates.

Example 1

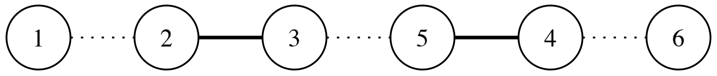
Consider this example graph.



Starting a BFS from vertex 1, we run into a problem when we encounter the edge between vertices 4 and 5. If we follow the edge, vertices 4 and 5 would appear at odd and even levels in the search.

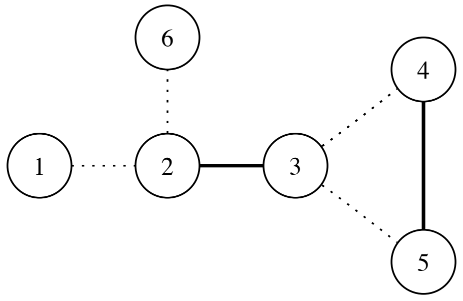


However, unless we allow a vertex to appear at both odd and even levels, the search stalls and we fail to find the augmenting path:

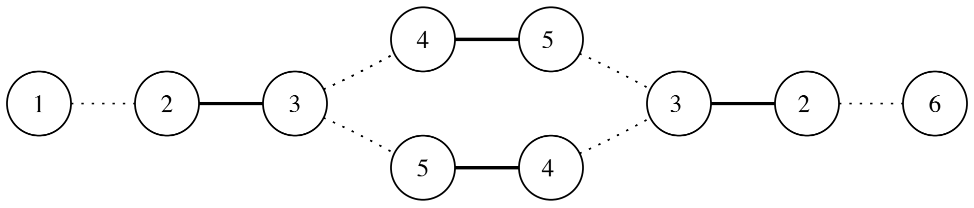


Example 2

The last example showed that in order to find every augmenting path, we must allow nodes to appear at even and odd levels in the search. However, this causes a new problem, which is seen in the following graph.



Running a BFS that allows cycles from vertex 1, we arrive at the graph:



We appear to have found an augmenting path from 1 to 6, but it is not a simple path - and, in fact, no augmenting path exists.

Paths, Trees, and Flowers

In a famous paper called *Paths, Trees, and Flowers*[2], Jack Edmonds resolved this dilemma, as we shall prove in the next lecture.

Call an edge in the matching *solid* and an edge not in the matching *dotted*. To search for an augmenting path from a given free vertex, Edmonds's algorithm builds a tree of alternating paths. The root, and all vertices at an even distance from the root, are called *outer vertices*. Vertices at an odd distance from the root are called *inner vertices*. If the search reaches a free inner vertex, then an augmenting path to that vertex can be constructed, as we show later.

The general tree-building step involves scanning an outer vertex v. Each solid edge (v, w), where w is not already in the tree, is added to the tree. The vertex w is designated *inner*, and the unique solid edge (w, x) that is incident with w is added to the tree, and x is designated *outer*.

If, in the process of scanning an outer vertex v, an edge (v, w) is found such that w is outer, then a cycle is formed as indicated in this diagram:

[2]Canadian Journal of Math, 1965

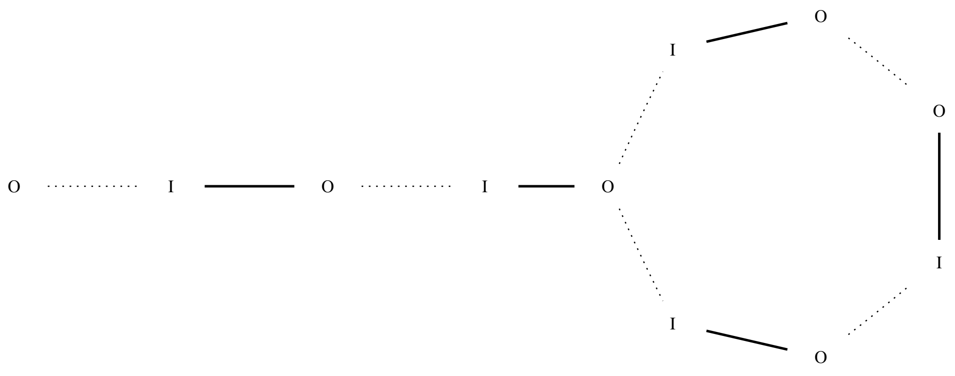


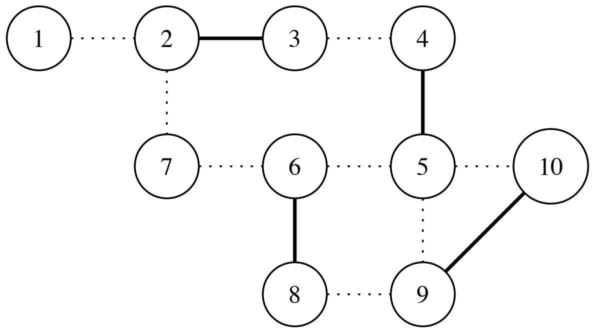
Figure 5. A cycle of inner and outer vertices.

In this case, the cycle is contracted to a single macrovertex, and the process continues. The macrovertex is called a *blossom*.

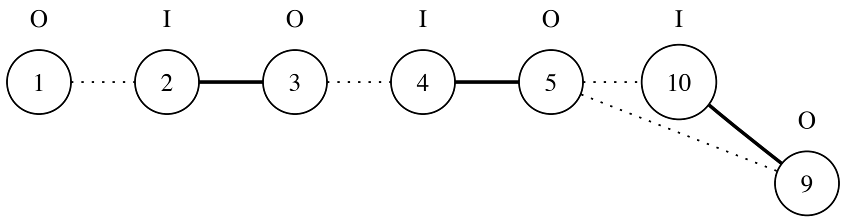
We shall show that, if a free inner vertex is encountered, then an augmenting path can be constructed from the root to that vertex. If the tree-building process ends without finding an augmenting path, then the vertices of the tree may be temporarily deleted until after the next augmentation.

First, an example:

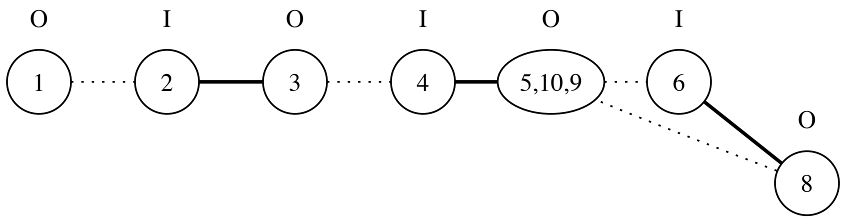
Example



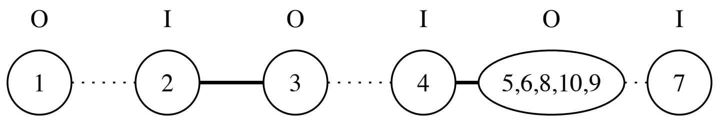
Starting a BFS from vertex 1, we encounter the cycle 5 - 10 - 9 in the graph.



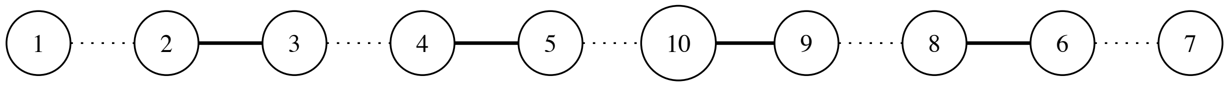
The vertices 5, 10, and 9 form a blossom, so we shrink 5,10,9 into a single macrovertex, and continue the search.



Shrink $(5, 10, 9), 6, 8$ into a single vertex.



An augmenting path in the shrunken graph is found. By unshrinking, we get the following augmenting path in the original graph.



It remains to be shown that it is always possible to unshrink the resulting augmenting path into an augmenting path in the original graph.