The Western Journal of Graduate Research 2001, Vol. 10 (1), 10-14

Maximum Matching and Minimum Vertex Covers of Trees

Tieling Chen

Ph.D.
Department of Computer Science
The University of Western Ontario
tielingc@usca.edu

Abstract

This paper considers the maximum matching problem and the minimum vertex cover problem on trees. A new proof of the correctness of a linear time algorithm is given.

Introduction

The maximum matching problem and the minimum vertex cover problem are classic ones in Graph theory. Given an undirected graph, a matching is a set of edges of the graph such that no two edges in the set have a vertex in common. For a given graph, one basic problem is to find a maximum matching which is one with the maximum number of edges. Matching in general graphs is difficult but good results have been obtained, especially on some specific graphs such as bipartite graphs (Alt *et al*., 1991; Hopcroft & Karp, 1973; Tarjan, 1983). An undirected graph $G = (V, E, U)$ is called a bipartite graph if V and U are two disjoint sets of vertices and E is a set of edges connecting vertices between V and U. Every edge in E is incident to two vertices belonging to different sets.

A vertex cover of an undirected graph is a set of vertices such that every edge in the graph is incident to at least one of these vertices. The minimum vertex cover problem is to find a vertex cover of a given undirected graph with the minimum number of vertices. Approximation algorithms with rate 2 can be obtained by using the algorithms for the maximum matching problem. Instead of finding an exact minimum vertex cover, these algorithms find a vertex cover with size no more than twice of the minimum size. The problem of finding an exact minimum vertex cover of a general graph is NP hard. The concept of NP hard can be found in Manber (1989) and many other books about algorithms.

Matching and vertex cover problems are also interesting problems outside of Computer Science. There are many problems involving matching and vertex cover. For example, workers or machines may be matched to jobs. Maximum matchings would improve the efficiencies. In very complicated circuits, the sets of nodes that are vertex covers are important for maintenance and examination. Minimum vertex covers would simplify the process and save time.

Many algorithms with low time complexities for the maximum matching problem are on bipartite graphs. The paper by Hopcroft and Karp (1973) gives an algorithm with a time complexity $O(n^{5/2})$ and Alt *et al*. (1991) provides an algorithm with time $O(n^{3/2}\sqrt{m/\log n})$. These algorithms also provide approximation algorithms with rate 2 for minimum vertex cover problem on bipartite graphs.

A tree can be regarded as a special (the simplest) bipartite graph. A linear time algorithm for finding a maximum matching of trees has been known (See Problem 7.95 in Manber, 1989). In this paper, we provide a simple and original proof for the correctness of this algorithm. With a slight modification, the algorithm also finds an exact minimum vertex cover in the tree case.

A fundamental strategy in bipartite matching is using alternating paths. Interested readers can refer to Section 7.10 in Manber (1989). For a given matching *M* in *G*, a vertex that is not incident to any edge in *M* is called unmatched. An alternating path for the matching *M* is a path connecting an unmatched vertex in *V* and an unmatched vertex in *W*, and the edges in the path are alternatively in *E* – *M* and in *M*. It is easy to see that the number of edges belonging to *E* – *M* is larger than the number of edges belonging to *M* by one. If there exists an alternating path for *M*, we can replace all the edges of the path belonging to *M* by the edges belonging to *E* – *M* and get an improved matching. The procedure of finding the maximum matching starts from some matching *M*, then searches for an alternating path and modifies the matching until no more alternating path can be found. The following Alternating Path Theorem (Berge, 1962) guarantees the final matching is a maximum one.

Alternating Path Theorem: A matching in a graph is maximum if and only if it has no alternating paths.

In a tree, if we put all the nodes in odd number layers in a set *V* and put all the nodes in even number layers in a set *U*, then the tree is converted into a bipartite graph. The Alternating Path Theorem is also valid for trees. An alternating path of a matching *M* in a tree is a path with one end node, which is unmatched, in an odd number layer, and the other end node, which is also unmatched, in an even number layer, and the edges in the path are alternatively in *E* – *M* and *M*.

Main Result

In the following, we first introduce the algorithm finding both a maximum matching and a minimum vertex cover of a tree in linear time, and then give a simple proof of the correctness. An example is also provided at the final.

Suppose T is a tree with a root *r*(*T*). Number the nodes by a Breadth First Search (BFS) starting at *r*(*T*). Suppose the size of *T* is *n*. Number the nodes of *T* with numbers from 1 to n. The root is first numbered, which is 1, and then all its children are numbered consequently. Then we go to the next level to number the grandchildren, and so on. For each node we record its parent. We will check all the nodes with the backward order of the numbers. This can be implemented by dealing with a queue of pointers to the nodes by a last in first out order.

We collect edges from *T* and put them in an initially empty set *M*, and collect nodes from *T* and put them in an initially empty set *S*. Check the nodes with the backward order of their numbers. The first node checked has the largest number and has no children. We collect the edge incident to this node and its parent into M, and collect the parent into S, and then mark the parent and consider the next node. Generally, when checking a node v, if it is marked or its parent w is marked then we just skip it and go to the next node, if both of v and w are not marked then we collect the edge (v, w) into M and collect w into S. Then mark w and go to the node next to v.

Lemma: When all the nodes have been checked, the set M is a maximum matching of T and the set S is a minimum vertex cover of T.

Proof: When all the nodes have been checked, each marked node contribute an edge to M together with one of its children, and there is no edge in T with two end nodes unmarked since if one node is not marked then its parent must be marked. It is also obvious that there is no edge with two ends marked in T, and each edge in M has exactly one end node marked and the marked node is the parent of the other end node.

The set M is matching. If two edges (v, w) and (x, y) in M have a common vertex $w = x$, then w must be the parent of both v and y, or the parent of one node, say v, and the child of the other node, say y. In either case, if one edge, say (v, w), is collected first, then $x = w$ is marked and (x, y) cannot be collected. So both cases cannot happen and M is a matching.

To show M is a maximum matching, we only need to show there is no alternating path of M since T is a bipartite graph. Suppose P is an alternating path of M and v and w are the two ends. The nodes v and w cannot be marked since each marked node contributes an edge to M together with one of its children. The number of edges in the path P is odd and we assume this number is $2k+1$ for some non-negative integer k. There are k edges in P belonging to M and each one of them has a marked end node. Thus there are k marked nodes and $k+2$ unmarked nodes in P. This means there are at least two unmarked nodes appear consecutively in P, which implies there is an edge with two unmarked ends in P (and hence in T). But this is impossible. So there does not exist any alternating path of M in T and hence M is a maximum matching.

Then we show that the set S is a minimum vertex cover. For a general graph, the set of end vertices of all the edges in a maximum matching is a vertex cover. Every minimum vertex cover must contain at least one end vertex of each edge in the maximum matching. Hence the vertex cover consisting of all the vertices of edges in a maximum matching has a size no more than twice of the minimum size. This is actually the basic idea of rate 2 approximation algorithms. It is easy to see that the size of a minimum vertex cover is bounded below by the size of a maximum matching. Back to our case, because the size of S equals the size of M, if we can show S is a vertex cover, it must be a minimum one.

From the above algorithm, after all the nodes of T have been checked, the tree does not have an edge with both ends unmarked. This implies that every edge is incident to a node in S and so S is a vertex cover. We give a pseudo code of the algorithm and analyze the time complexity. For example:

Algorithm Maximum_matching_minimum_vertex_cover(*T*)
Input: *T* is a rooted tree with *n* nodes.
Output: A maximum matching *M* and a minimum vertex cover *S* of *T*.

begin
    M ← 0
    S ← 0
    number the nodes of T by a **BFS**
    (check the nodes with the backward order of the numbers)
    for every node **do**
        if the node or its parent is marked **then**
            skip this node
        else
            mark its parent
            add the edge to M
            add the parent to S
end

The BFS costs time of O(*n*) and the loop also costs time of O(*n*), so the time complexity of the above algorithm is O(*n*).

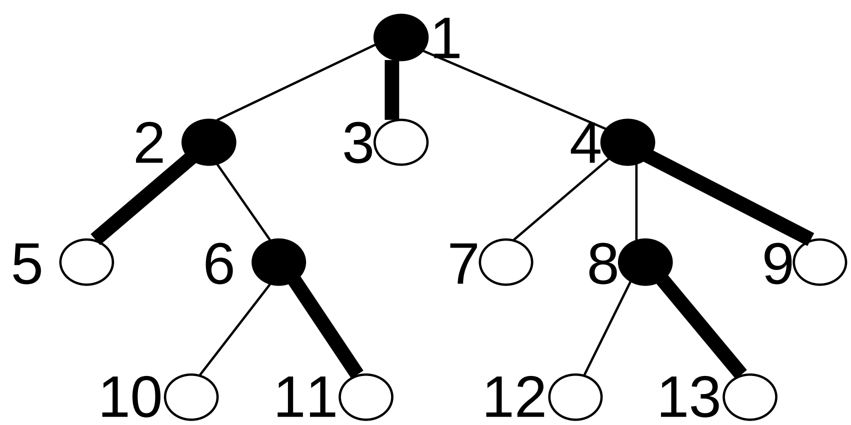

Figure 1. An example of applying the algorithm on a tree with size 13 is shown. The set of the thick edges is a maximum matching, and the set of the black nodes is a minimum vertex cover.

References

Alt H., Blum N. & Melhorn K. (1991) Computing a maximum cardinality matching in a bipartite graph in time $O(n^{1.5}\sqrt{m/\log n})$, *Information Processing Letters*, 37(4):237-240

Berge C. (1962). *The Theory of Graphs and Its Applications*. John Wiley and Sons, New York.

Hopcroft J. & Karp R. (1973) An $O(n^{5/2})$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225-231.

Manber U. (1989). *Introduction to algorithms: a creative approach*. Addison-Wesley, Reading, Mass. and Don Mills, Ont.

Tarjan R. (1983). *Data structures and network algorithms*, SIAM, Philadelphia.