An efficient algorithm for the *MPQ*-tree from an interval representation

Toshiki Saitoh (510040)

School of Information Science,
Japan Advanced Institute of Science and Technology

February, 8 ,2007

Keywords: interval graph, interval representation, *MPQ*-tree, graph algorithm.

Interval graphs were introduced in the 1950's by Hajös and Benzer independently. An undirected graph $G = (V, E)$ is an interval graph if and only if there is a one-to-one correspondence between its vertices and a set of intervals $\mathcal{I}$ of a linearly ordered set, such that two vertices are connected by an edge of G if and only if their corresponding intervals have nonempty intersection. Then $\mathcal{I}$ is called an interval representation of G. Interval graphs have a number of applications such as model the topological structure of the DNA molecule and scheduling. In the applications, data are often given in the form of interval representations, and the size of them are quite huge in the area of bioinfomatics.

From graph theoretical point of view, Interval graphs are subclass of chordal graphs. Many NP-hard problems on general graphs can be solved efficiently if we restrict the graph class to chordal graphs or to interval graphs. Hence problems on chordal graphs and interval graphs, such as recognition and isomorphism, have been widely investigated.

Booth and Lueker introduced a data structure called *PQ*-tree for recognition of interval graphs in 1976. Korte and Möhring simplified their recognition algorithm by introducing *MPQ*-tree in 1989. *MPQ*-trees can be viewed as a canonical form of an interval graph; that is, given two interval graphs are isomorphic if and only if their corresponding *MPQ*-trees are isomorphic. Every interval representation of an interval graph G can be obtained from the *MPQ*-tree which is corresponding to G. An *MPQ*-tree is a compact structure since if requires $O(|V|)$ space. Thus, *MPQ*-tree is a useful data sturucture of an interval graph. Therefore, efficient algorithm constructing an *MPQ*-tree from an interval representation is important in practical use.

There are two known ways to construct an *MPQ*-tree from an interval representation.

The first one is as follows. The algorithm constructs a graph representation from an input interval representation. Then, the graph representation is translated into a *PQ*-tree. Finally, the algorithm constructs an *MPQ*-tree using the *PQ*-tree. There are so many conditional branching on the process of translation of graph representations to *PQ*-trees, that an implementation of the algorithm gets complicated. In addition, since the algorithm uses a graph representation with $O(|V| + |E|)$ space, the algorithm must take $O(|V| + |E|)$ time and $O(|V| + |E|)$ space.

The second one is as follows. The algorithm constructs a graph representation from an input interval representation. Then, the algorithm constructs an *MPQ*-tree directly without constructing *PQ*-tree. Since this algorithm omits the construction of *PQ*-trees, it is simpler than the first one. However, since there are many conditional branches in the algorithm, too, an implementation is still complicated. The algorithm takes $O(|V| + |E|)$ time and $O(|V| + |E|)$ space as well as the first one.

In this paper, we introduce an algorithm that constructs the *MPQ*-tree directly from an input interval representation. The algorithm does not construct graph representation and runs in $O(|V|)$ time and $O(|V|)$ space. The algorithm uses stacks, and the implementation is simple.

An input interval representation is redundant. Treating the redundant interval representations is complicated. To make it simple, the algorithm constructs from an input interval representation a compact interval representation that is not redundant. Additionally, the algorithm reorder the intervals for simplicity. Then, the algorithm sweeps the interval representation from the left side and constructs the *MPQ*-tree. As the result, the algorithm takes $O(|V|)$ time and $O(|V|)$ space to construct the *MPQ*-tree.