MATH 640 PROJECT 4
DOMINATORS OF DIRECTED GRAPHS

NICK BELOV, JAMES BETTI, AND NURAY KUTLU

1. Defining the Problem

Definition 1.1 (Domination Relation)**.** Let $G = (V, E)$ be a directed graph with some designated source vertex S. It is assumed that every vertex in G is reachable starting from S. For distinct vertices v and w, v is said to be *dominated* by w if every path from S to v contains w.

Definition 1.2 (Immediate Dominator)**.** A vertex w is an *immediate dominator* of v if w dominates v and every other dominator of v dominates w. A vertex w is an *immediate dominator* of v if w dominates v and every other dominator of v dominates w.

Theorem 1.3. *Every vertex except for S has a unique immediate dominator.*

The immediate dominator of a vertex v will be denoted $\text{idom}(v)$.

Definition 1.4 (Dominator Tree)**.** The *dominator tree* of a directed graph G is a directed tree T such that:

- The root of T is the source vertex S.
- Each vertex v in G is a child of its immediate dominator $\text{idom}(v)$.

The dominator tree of a directed graph entirely captures the domination relationships in the graph. In paticular, the dominators of a fixed vertex v are exactly the ancestors of v in the dominator tree!

In our project, we implemented two algorithms for computing the dominator tree of a directed graph. The first being a naïve algorithm that runs in $O(|E| \cdot |V|)$ and the second being the much more efficient Lengauer-Tarjan algorithm that runs in $O(|E| \log |V|)$.

2. Maple Implementation

Our implementation provides two Maple procedures:

`constructDominator`**:** a straightforward enumeration method that follows the definition of dominance and runs in $O(|V|\,|E|)$ time;

`constructDominatorC`**:** an optimized version that mirrors the Lengauer–Tarjan approach.

Both procedures expect a digraph encoded as a triple $[n, E, S]$ where

- n is the number of vertices (labeled $1, \ldots, n$),
- $E = \{[u_1, v_1], \ldots, [u_m, v_m]\}$ is the edge set represented as a Maple *set* of two–element lists $[u, v]$ (edge $u \to v$),
- S is the designated source vertex. Both functions assume that every vertex is reachable from S.

Each function returns a list of length n. The ith entry of this list is $\text{idom}(i)$, the label of the immediate dominator of vertex i. For the root S the entry is 0. In order to ensure correctness of the fast algorithm we tested both functions against small random digraphs to ensure that they produce the same output.

3. A Slow Algorithm

There is an easy $O(|E| \cdot |V|)$ solution using depth-first search (DFS). A DFS is performed to check which vertices are reachable from r. Then for each vertex v:

- remove v from the graph;
- perform DFS starting from r to determine which vertices are now reachable;
- the vertices no longer reachable are dominated by v.

In particular the dominators of each non-root vertex v can be determined, which can be checked to find $\text{idom}(v)$.

4. A Fast Algorithm

A complete description and justification of the Lengauer-Tarjan algorithm, which achieves an $O(|E| \log |V|)$ time complexity, can be found in the original paper. Here, we highlight some of the main aspects of the algorithm.

First, it is helpful to relabel the vertices in DFS order.

Definition 4.1 (Semidominator)**.** Given a vertex w in a directed graph labeled according to DFS order, the *semidominator* of w is defined as:

$$\text{sdom}(w) = \min\{v \mid \text{there is a path } v = v_0, v_1, \ldots, v_k = w \text{ with } v_i > w \text{ for } 1 \leq i < k\}.$$

Lengauer and Tarjan proved the following relationship:

Theorem 4.2. *Let u be the vertex with minimum $\text{sdom}(u)$ among all vertices u such that there is a path $\text{sdom}(w) \to u \rightsquigarrow w$. Then the immediate dominator of w satisfies:*

$$\text{idom}(w) = \begin{cases} \text{sdom}(w), & \text{if } \text{sdom}(w) = \text{sdom}(u), \\ \text{idom}(u), & \text{otherwise.} \end{cases}$$

This theorem enables us to efficiently compute the immediate dominators if we already know the semidominators.

To actually compute the semi-dominators, Lengauer and Tarjan used a union-find data structure with path compression which is the final ingredient in the algorithm.

References

[1] Thomas Lengauer and Robert Endre Tarjan, *A fast algorithm for finding dominators in a flowgraph*, ACM Trans. Program. Lang. Syst. **1** (1979), no. 1, 121–141.