Transitive closure of a directed graph

Contents

1 Formulation of the problem

Let $G = (V, E)$ be a directed graph. The sequence $P(u, v)$ of edges $e_1 = (u, w_1)$, $e_2 = (w_1, w_2)$, ..., $e_k = (w_{k-1}, v)$ is called a "path *from the vertex* u *to* v. *We say that* v is *reachable* from u if there exists at least one path $P(u, v)$. Each vertex is reachable from itself.

It is required to construct the **transitive closure** $G^+ = (V, E^+)$ of the graph G; namely, an edge $(v, w) \in E^+$ if and only if the vertex w is reachable in the graph G from the vertex v.

An equivalent formulation is as follows: Given a reflexive binary relation R, construct the minimal (with respect to inclusion) relation R^+ that contains R and has the transitivity property; that is, if aR^+b and bR^+c, then aR^+c. If the original relation R is symmetric, then R^+ is an equivalence relation, and it suffices to find the corresponding equivalence classes.

2 Properties of the problem

If vertices v and w belong to the same strongly connected component of the graph G, then the transitive closure contains the edges (v, w) and (w, v). In an undirected graph, the edge (v, w) belongs to the transitive closure if and only if the vertices v and w belong to the same connected component. Consequently, for an undirected graph, the search for transitive closure is equivalent to finding connected components.

If vertices x and y belong to the same strongly connected component of the graph G, while vertices z and t belong to the other component, then the edges (x, z), (x, t), (y, z), and (y, t) simultaneously belong or not belong to the transitive closure G^+. It follows that the search for the transitive closure of the graph G can be reduced to finding the transitive closure of the acyclic graph obtained from G by merging each strongly connected component into a single vertex. This idea underlies Purdom's algorithm алгоритм Пурдома.

In terms of vertices, the graph G and its transitive closure G^+ have the same strongly connected components.

3 Algorithms for solving the problem

In an **undirected** graph, a vertex w is reachable from a vertex v if and only if both belong to the same connected component. The transitive closure of such graph reduces to finding its connected components and can be constructed by the following algorithms:

- a systematic application of the breadth first search. The time complexity is $O(m)$;
- the use of a system of disjoint sets[1] The time complexity is $O(m\alpha(m, n))$. An effective multithreaded implementation is possible [2];
- the parallel algorithm of Shiloach-Vishkin[3] The time complexity is $O(\ln n)$, provided that $n + 2m$ processors are used.

For a **directed** graph, the transitive closure can be reduced to the search for shortest paths in a graph with unit weights. It can then be found by the following algorithms:

- **Floyd--Warshall algorithm**[4][5]. The complexity is $O(n^3)$. (The first published algorithm for transitive closure [6] is exactly the Floyd--Warshall algorithm for graphs with unit weights);
- a repeated application of the **breadth first search**. The complexity is $O(mn)$;
- **Purdom's algorithm**[7]. The complexity is $O(m + \mu n)$.

Notation: m is the number of edges, n is the number of vertices, and $\mu \leq m$ is the number of edges that connect strongly connected components.

4 Parallelization resource

The problem can be solved by applying the breadth first search to all the vertices of the graph. For different vertices, the calculations are independent and may be performed in parallel.

Since the size of the output data and the complexity of calculations are quadratic, it makes sense to perform a preliminary processing of the data and use various graph decompositions in order to reduce the original problem to the parallel finding of shortest paths on smaller subgraphs [8]:

- selection of connected components;
- selection of subgraphs connected only by bridges (that is, by edges whose removal makes the graph disconnected);
- selection of biconnected components connected only by joints (also called articulation points or cut vertices; these are vertices вершины, whose removal makes the graph disconnected);
- removal of hanging vertices (that is, vertices having at most one adjacent vertex).

Which decompositions are advantageous in a specific situation depends on the structure of the graph and can be determined only by analyzing this structure.

5 References

1. Tarjan, Robert Endre. “Efficiency of a Good but Not Linear Set Union Algorithm.” Journal of the ACM 22, no. 2 (April 1975): 215–25. doi:10.1145/321879.321884.
2. Anderson, Richard J, and Heather Woll. “Wait-Free Parallel Algorithms for the Union-Find Problem,” 370–80, New York, New York, USA: ACM Press, 1991. doi:10.1145/103418.103458.
3. Shiloach, Yossi, and Uzi Vishkin. “An $O(\log n)$ Parallel Connectivity Algorithm.” Journal of Algorithms 3, no. 1 (March 1982): 57–67. doi:10.1016/0196-6774(82)90008-6.
4. Floyd, Robert W. “Algorithm 97: Shortest Path.” Communications of the ACM 5, no. 6 (June 1, 1962): 345. doi:10.1145/367766.368168.
5. Warshall, Stephen. “A Theorem on Boolean Matrices.” Journal of the ACM 9, no. 1 (January 1, 1962): 11–12. doi:10.1145/321105.321107.

6. Roy, Bernard. “Transitivité Et Connexité.” Comptes Rendus De l'Académie Des Sciences 249 (1959): 216–218.
7. Purdom, Paul, Jr. “A Transitive Closure Algorithm.” Bit 10, no. 1 (March 1970): 76–94. doi:10.1007/BF01940892.
8. Banerjee, Dip Sankar, Ashutosh Kumar, Meher Chaitanya, Shashank Sharma, and Kishore Kothapalli. “Work Efficient Parallel Algorithms for Large Graph Exploration on Emerging Heterogeneous Architectures.” Journal of Parallel and Distributed Computing, December 2014. doi:10.1016/j.jpdc.2014.11.006.