IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, June 2022
Impact Factor: 6.252

Topological Sorting

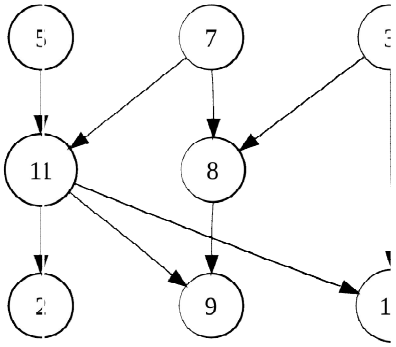
Aibel Shajan[1], Ashin Santhosh[2], Sigma Sathyan[3]
Student, Computer Science, Santhigiri College Vazhithala, Thodupuzha, India[1,2]
Assistant Professor, Computer Science, Santhigiri College Vazhithala, Thodupuzha, India[3]

Abstract: *Here in this paper, we are conducting a study on Topological Sorting, the different types and methods used for topological sorting and its applications. We discuss about the order in which a given graph is sorted. The topic also discusses about the kinds of methods used in order to provide the topological order a proper algorithm. Here we also aim at the exploring of various applications of topological ordering and its time and space complexity.*

Keywords: Sorting, Order, Vertex, Edge, DFS, Time Complexity, etc.

I. INTRODUCTION

In computer science, a topological sort or topological ordering of a directed graph is a linear ordering of its vertices such that for every directed edge uv from vertex u to vertex v, u comes before v in the ordering. For example, the vertices of the graph may represent tasks to be performed, and the edges may represent constraints that one task must be performed before another; in this application, a topological ordering is just a valid sequence for the tasks. Precisely, a topological sort is a graph traversal in which each node v is visited only after all its dependencies are visited. A topological ordering is possible if and only if the graph has no directed cycles, that is, if it is a directed acyclic graph (DAG). Any DAG has at least one topological ordering, and algorithms are known for constructing a topological ordering of any DAG in linear time. Topological sorting has many applications especially in ranking problems such as feedback arc set.



Topological Graph

II. TOPOLOGICAL SORTING

The canonical application of topological sorting is in scheduling a sequence of jobs or tasks based on their dependencies. The jobs are represented by vertices, and there is an edge from x to y if job x must be completed before job y can be started. Then, a topological sort gives an order in which to perform the jobs.

1. History of Topological Sorting

A closely related application of topological sorting algorithms was first studied in the early 1960s in the context of the PERT technique for scheduling in project management. In this application, the vertices of a graph represent the milestones of a project, and the edges represent tasks that must be performed between one milestoneandanother.Topological sorting forms the basis of linear-time algorithms for finding the critical path of the project, a sequence of milestones and tasks that controls the length of the overall project schedule.

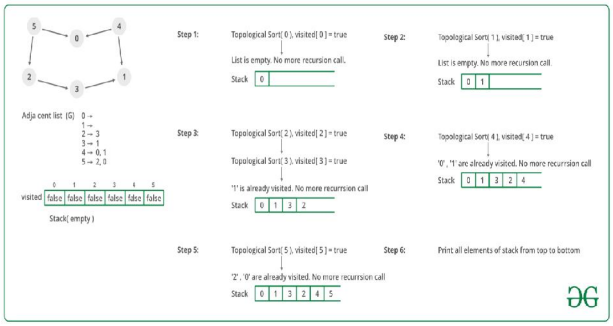
IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, June 2022
Impact Factor: 6.252

Algorithms

The usual algorithms for topological sorting have running time linear in the number of nodes plus the number of edges, asymptotically,O(|V|+|E|)

2. Algorithm to find Topological Sorting

We recommend to first see the implementation of DFS. We can modify DFS to find Topological Sorting of a graph. In DFS, we start from a vertex, we first print it and then recursively call DFS for its adjacent vertices. In topological sorting, we use a temporary stack. We don t print the vertex immediately, we first recursively call topological sorting for all its adjacent vertices, then push it to a stack. Finally, print contents of the stack. Note that a vertex is pushed to stack only when all of its adjacent vertices (and their adjacent vertices and so on) are already in the stack.



3. Kahn's Algorithm

One of these algorithms, first described by Kahn (1962), works by choosing vertices in the same order as the eventual topological sort. First, find a list of "start nodes" which have no incoming edges and insert them into a set S; at least one such node must exist in a non-empty acyclic graph. Then:

```
L ← Empty list that will contain the sorted elements
S ← Set of all nodes with no incoming edge

while S is not empty do
    remove a node n from S
    add n to L
    for each node m with an edge e from n to m do
        remove edge e from the graph
        if m has no other incoming edges then
            insert m into S

if graph has edges then
    return error   (graph has at least one cycle)
else
    return L   (a topologically sorted order)
```

if the graph is a DAG; a solution will be contained in the list L (the solution is not necessarily unique). Otherwise, the graph must have at least one cycle and therefore a topological sort is impossible.Reflecting the non-uniqueness of the resulting sort, the structure S can be simply a set or a queue or a stack. Depending on the order that nodes n **is** removed from set S, a different solution is created. A variation of Kahn's algorithm that breaks ties lexicographically forms a key component of the Coffman Graham algorithm for parallel scheduling and layered graph drawing.

4. Depth-first Search

An alternative algorithm for topological sorting is based on depth-first search. The algorithm loops through each node of the graph, in an arbitrary order, initiating a depth-first search that terminates when it hits any node that has already been visited since the beginning of the topological sort or the node has no outgoing edges.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, June 2022
Impact Factor: 6.252

```
L ← Empty list that will contain the sorted nodes
while exists nodes without a permanent mark do
    select an unmarked node n
    visit(n)

function visit(node n)
    if n has a permanent mark then
        return
    if n has a temporary mark then
        stop   (not a DAG)

    mark n with a temporary mark

    for each node m with an edge from n to m do
        visit(m)

    remove temporary mark from n
    mark n with a permanent mark
    add n to head of L
```

Each node n gets prepended to the output list L only after considering all other nodes which depend on n (all descendants of n in the graph). Specifically, when the algorithm adds node n, we are guaranteed that all nodes which depend on n are already in the output list L: they were added to L either by the recursive call to visit () which ended before the call to visit n, or by a call to visit () which started even before the call to visit n. Since each edge and node is visited once, the algorithm runs in linear time. This depth-first-search-based algorithm is the one described by Cormen et al. (2001) it seems to have been first described in print by Tarjan in 1976.

5. Parallel Algorithms

On a parallel random-access machine, a topological ordering can be constructed in O(log2 n) time using a polynomial number of processors, putting the problem into the complexity class NC2. One method for doing this is to repeatedly square the adjacency matrix of the given graph, logarithmically many times, using min-plus matrix multiplication with maximization in place of minimization. The resulting matrix describes the longest path distances in the graph. Sorting the vertices by the lengths of their longest incoming paths produces a topological ordering.

```
p processing elements with IDs from 0 to p-1
Input: G = (V, E) DAG, distributed to PEs, PE index j = 0, ..., p - 1
Output: topological sorting of G

function traverseDAGDistributed
    δ incoming degree of local vertices V
    Q = {v ∈ V | δ[v] = 0}                    // All vertices with incegree 0
    nrOfVerticesProcessed = 0

    do
        global build prefix sum over size of Q     // get offsets and total amount of vertices in this step
        offset = nrOfVerticesProcessed + sum(Q_i, i = 0 to j - 1)          // j is the processor index
        foreach u in Q
            localOrder[u] = index++;
            foreach (u,v) in E do post message (u, v) to PE owning vertex v
        nrOfVerticesProcessed += sum(|Q_i|, i = 0 to p - 1)
        deliver all messages to neighbors of vertices in Q
        receive messages for local vertices V
        remove all vertices in Q
        foreach message (u, v) received:
            if --δ[v] = 0
                add v to Q
    while global size of Q > 0

    return localOrder
```

The communication cost depends heavily on the given graph partition. As for runtime, on a CRCW-PRAM model that allows fetch-and-decrement in constant time.

III. ADVANTAGES OF TOPOLOGICAL SORTING

- Topological Sorting is mostly used to schedule jobs based on their dependencies. Instruction scheduling, ordering formula cell evaluation when recomputing formula values in spreadsheets, logic synthesis, determining the order of compilation tasks to perform in make files, data serialization, and resolving symbol dependencies in linker are all examples of applications of this type in computer science.
- Finding cycle in a graph: Only directed acyclic graphs may be ordered topologically (DAG). It is

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, June 2022
Impact Factor: 6.252

impossible to arrange a circular graph topologically.

- Operation System deadlock detection: A deadlock occurs when one process is waiting while another holds the requested resource.
- Dependency resolution: Topological Sorting has been proved to be very helpful in Dependency resolution.
- Critical Path Analysis: A project management approach known as critical route analysis. It's used to figure out how long a project should take and how dependent each action is on the others. There may be some preceding actions before an activity. Before beginning a new activity, all previous actions must be completed.
- Course Schedule problem: Topological Sorting has been proved to be very helpful in solving the Course Schedule problem.
- Other applications like manufacturing workflows, data serialization, and context-free grammar.

IV. TIME AND SPACE COMPLEXITY

Determine the indegree for each node. This is O(M) time (where M is the number of edges), since this involves looking at each directed edge in the graph once.

1. Find nodes with no incoming edges.
2. This is a simple loop through all the nodes with somenumber ofconstant-time appends.
3. O(N) time (where N is the number of nodes).
4. Add nodes until we run out of nodes with no incoming edges. This loop could run once for every node O(N) times. In the body, we:
5. Do two constant-time operations on an array to add a node to the topological ordering.
6. Decrement the indegree for each neighbor of the node we added. Over the entire algorithm, we'll end up doing exactly one decrement for each edge, making this step O(M) time overall.
7. Check if we included all nodes or found a cycle. This is a fast O (1) comparison.
8. Altogether, the time complexity is O(M+N).
9. That's the fastest time we can expect, since we'll have to look at all the nodes and edges at least once.
10. What about space complexity? Here are the data structures we created:
11. indegrees this has one entry for each node in the graph, so it's O(N)space.
12. nodesWithNoIncomingEdges in a graph with no edges, this would start out containing every node, so it's O(N) space in the worst case.
13. topologicalOrdering in a graph with no cycles, this will eventually have every node. O(N) space.
14. All in all, we have three structures and they're all O(N) space. Overall space complexity: O(N).
15. This is the best space complexity we can expect, since we must allocate a return array which costs O(N) space itself.

V. CONCLUSION

Topological Sort for a DAG is a linear ordering of vertices such that every directed edge(u,v),vertex u comes before v in ordering. It is used in building systems, advanced packaging tools task scheduling and so on. It is an efficient method which is implied in programing where one process requires the result of another process and by applying topological sort, we can ensure the smooth working of the program.

REFERENCES

[1] https://www.youtube.com/watch?v=Q9PIxaNGnig
[2] https://www.youtube.com/watch?v=eL-KzMXSXXIK
[3] https://www.geeksforgeeks.org/topological-sorting/
[4] https://www.scaler.com/topics/data-structures/topological-sort-algorithm/
[5] https://en.wikipedia.org/wiki/Topological_sorting#:~:text=In%20computer%20science%2C%20a%20topological,before%20v%20in%20the%20ordering.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, June 2022
Impact Factor: 6.252

[6] https://www.hackerearth.com/practice/algorithms/graphs/topological-sort/tutorial/
[7] https://www.youtube.com/watch?v=Yh5o_PSK9to
[8] https://www.javatpoint.com/topological-sorting
[9] https://www.educative.io/edpresso/what-is-topological-sort