Topological Sort with smallest available vertex first

I am trying to solve a problem based on TopSort. Since there will be a number of valid topological sorted orderings, I need to output the one which is lexicographically the smallest, i.e. Smallest-numbered available vertex first.

I will explain the method I have adopted, then post the code. I have maintained an array which contains the number of in-edges for each node of the graph. An adjacency matrix is also present. I have a boolean array 'visited' which keeps a count of visited nodes, and a minimum priority queue to pop out the smallest available vertex at that moment. Here is my code:

```
void dfs(int u){

    visited[u] = true;
    cout<<u<<" ";

    list<int>::iterator i;
    for(i = adj[u].begin(); i != adj[u].end(); ++i){

        if(!visited[*i]){

            inedge[*i]--;
            if(!inedge[*i]){
                pq.push(*i);
            }

            if(!pq.empty()){
                int temp = pq.top();
                pq.pop();
                dfs(temp);
            }
        }
    }
}
```

Now, at the first call of this function, the priority queue contains only those nodes for which inedge[i]=0 (number of in-edges is zero). I pop out the minimum node from that priority queue `u = pq.top()` , and call `dfs(u)` .

But my code is giving wrong outputs. Can anybody help me out if the logic I used here is wrong?

My input consists of N incomplete sequences (with missing numbers in each of the N). Input:

```
2
1 3
2 3 4
```

Output:

```
1 2 3 4
```

This is the correct expected output, and my code does generate this. But for an input given in this image Input:

```
6
7 8 9
7 11 9
5 11 2
3 8 9
11 10
3 10
```

expected output is :

```
3 5 7 8 11 2 9 10
```

My code outputs :

```
3 5 7 8 11 9 2 10
```

My code outputs wrong results for a few other test cases as well, but I do not know how to solve this issue.

c++ priority-queue depth-first-search topological-sort

edited Nov 20 '14 at 8:46

asked Nov 20 '14 at 8:31
user2774555
69 1 10

define "wrong outputs" and "expected outputs" and give examples. – Ashalynd Nov 20 '14 at 8:35

what is the criterion for your priority queue? it's not present in this code, so it's difficult to say anything. – Ashalynd Nov 20 '14 at 8:37

It is a minimum priority queue – user2774555 Nov 20 '14 at 8:47

I would be interested in which context you need this. Scheduling? – SebastianK Nov 20 '14 at 8:48

@Aleksandar - yes.. Can you tell me whats wrong in my approach? – user2774555 Nov 20 '14 at 8:53

1 Answer

The problem seems to be that `pq.top` is called before all outgoing edges of the current node have been checked.

Consider the following graph: Nodes: A, B, C; Edges A->B, A-C. Let C have a smaller priority value, i.e. it should come first. During dfs(A), you check B before C. Since it is immediately taken from the queue again, B is processed first (but should not be). So, insert all adjacent nodes before querying the queue again.

edited Nov 20 '14 at 9:41

answered Nov 20 '14 at 8:44
SebastianK
2,611 1 18 44

Oh yes, I realized this fact, just that I do not know how to solve this issue. It is due to the same reason that 9 comes before 2 in my input-output – user2774555 Nov 20 '14 at 8:50

Try it nonrecursively: `while (!pq.empty) { process pq.top /*...*/ }` – SebastianK Nov 20 '14 at 8:52

Thanks! it seems to be working now. – user2774555 Nov 20 '14 at 9:04