

# 1 Topological Sort

In this section, we will be discussing only directed graphs.

If  $G = (V, E)$  is a directed graph, a **topological sort** of  $G$  is a one to one correspondence

$$f : V \rightarrow \{1, 2, \dots, |V|\}$$

such that  $f(v) < f(w)$  for all  $(v, w) \in E$ .

It is important to note that not all graphs have a topological sort. In fact, a graph has a topological sort if and only if the graph has no non-trivial cycles (the graph is acyclic).

Graphs that do have a topological sort may have a unique topological sort, or several valid topological sorts.

If we know the graph  $G$  is acyclic, the following algorithm will find a topological sort of  $G$ :

## **TopologicalSort1(G)**

Call DFS(G) to compute the finishing time for each vertex

Put the vertices in a list in reverse sorted order of the finishing time

Return the list

We can prove the correctness of this algorithm in the following way:

*Proof.* Let  $G = (V, E)$  be a directed, acyclic graph.

Let  $(v, w) \in E$ .

Need to show that after running DFS,  $v.f > w.f$

Case 1:  $w$  is black when  $(v, w)$  is explored by DFS.

So  $w.f$  is set and  $v.f$  is not, so when the algorithm finishes  $v.f > w.f$ .

Case 2:  $w$  is gray when  $(v, w)$  is explored by DFS.

So  $(v, w)$  is a back edge, and so  $G$  has a cycle. But  $G$  is acyclic, so this cannot happen.

Case 3:  $w$  is white when  $(v, w)$  is explored by DFS.

So  $v.f$  cannot be set until after  $w.f$  is set. So when the algorithm finished,  $v.f > w.f$ . □

We can rewrite our algorithm so that if  $G$  has a cycle, the algorithm correctly detects the cycle and reports that.

### **TopologicalSort2**( $G$ )

```
cycle = false
count = 1
for each  $v \in V$ 
     $v.color = \text{white}$ 
for each  $v \in V$ 
    if  $v.color = \text{white}$  and  $\neg \text{cycle}$ 
        Visit( $G, v$ )
if cycle
    output true
```

### **Visit**( $G, v$ )

```
 $v.color = \text{gray}$ 
for each  $w$  s.t.  $(v, w) \in E$ 
    if  $w.color = \text{gray}$ 
        cycle = true
    else if  $w.color = \text{white}$  and  $\neg \text{cycle}$ 
        Visit( $G, w$ )
 $v.color = \text{black}$ 
 $v.count = \text{count}$ 
count = count + 1
```