

## Chapter 10

# Strongly Connected Components and Topological Sort.

### 10.1 Topological Sort

**Definition 10.1.1.** (connectivity)

1. Let  $G = (V, E)$  be a non-directed graph. A **connected component** of  $G$  is a subset  $U \subseteq V$  of maximal size in which there exists a path between every two vertices.
2. A non-directed graph  $G$  is said to be a **connected** graph if it only has one connected component.
3. Let  $G = (V, E)$  be a directed graph. A **strongly connected component** of  $G$  is a subset  $U \subseteq V$  of maximal size in which for any pair of vertices  $u, v \in U$  there exist both directed path from  $u$  to  $v$  and a directed path from  $v$  to  $u$ .

#### 10.1.1 Depth First Search (DFS)

As its name implies, depth-first search searches "deeper" in the graph whenever possible. Depth-first search explores edges out of the most recently discovered vertex  $v$  that still has unexplored edges leaving it. Once all of  $v$ 's edges have been explored, the search "backtracks" to explore edges leaving the vertex from which  $v$  was discovered. This process continues until all vertices that are reachable from the original source vertex have been discovered. If any undiscovered vertices remain, then depth-first search selects one of them as a new source, repeating the search from that source. The algorithm repeats this entire process until it has discovered every vertex.

```

1 DFS(  $G$  ):
2   for  $v \in V$  do
3      $v.\text{visited} \leftarrow \text{False}$ 
4   end
5    $\text{time} \leftarrow 1$ 
6   for  $v \in V$  do
7     if not  $v.\text{visited}$  then
8        $\pi(v) \leftarrow \text{null}$ 
9       Explore(  $G, v$  )
10    end
11  end

1 Explore( $G, v$ ):
2   Previsit( $v$ )
3   for  $(v, u) \in E$  do
4     if not  $u.\text{visited}$  then
5        $\pi(u) \leftarrow v$ 
6       Explore(  $G, u$  )
7     end
8   end
9   Postvisit( $v$ )

1 Previsit( $v$ ):
2    $\text{pre}(v) \leftarrow \text{time}$ 
3    $\text{time} \leftarrow \text{time} + 1$ 

1 Postvisit( $v$ ):
2    $\text{post}(v) \leftarrow \text{time}$ 
3    $\text{time} \leftarrow \text{time} + 1$ 

```

**Properties of depth-first search.** Depth-first search yields valuable information about the structure of a graph. Perhaps the most basic property of depth-first search is that the predecessor subgraph  $G_\pi$  does indeed form a forest of trees since the structure of the depth-first trees exactly mirrors the structure of recursive calls of explore-function. That is,  $u = \pi(v)$  if and only if  $\text{explore}(G, v)$  was called during a search of  $u$ 's adjacency list.

Additionally, vertex  $v$  is a descendant of vertex  $u$  in the depth-first forest if and only if  $v$  is discovered during the time in which  $u$  is gray. Another important property of depth-first search is that discovery and finish times have a parenthesis structure. If the explore procedure were to print a left parenthesis " $(u$ " when it discovers vertex  $u$  and to print a right parenthesis " $)u$ " when it finishes  $u$ , then the printed expression would be well-formed in the sense that the parentheses are properly nested.

The following theorem provides another way to characterize the parenthesis structure.

**Theorem 10.1.1** (Parenthesis theorem). *In any depth-first search of a (directed or undirected) graph  $G = (V, E)$ , for any two vertices  $u$  and  $v$ , exactly one of the following three conditions holds:*

1. *the intervals  $[\text{pre}(u), \text{post}(u)]$  and  $[\text{pre}(v), \text{post}(v)]$  are entirely disjoint, and neither  $u$  nor  $v$  is a descendant of the other in the depth-first forest.*
2. *the interval  $[\text{pre}(u), \text{post}(u)]$  is contained entirely within the interval  $[\text{pre}(v), \text{post}(v)]$ , and  $u$  is a descendant of  $v$  in a depth-first tree, or*

3. the interval  $[pre(v), post(v)]$  is contained entirely within the interval  $[pre(u), post(u)]$ , and  $v$  is a descendant of  $u$  in a depth-first tree.

*Proof.* Assume without loss of generality that  $pre(u) < pre(v)$ . Split to the following:

1. Either  $pre(v) < post(u)$ . In that case, we will prove, by induction on  $pre(v) - pre(u)$ , that for any  $u, v$  satisfies the relations, the third case holds.
  - (a) Base.  $pre(v) - pre(u) = 1$ . Then clearly  $\{u, v\}$ , i.e.  $v$  is a direct child of  $u$ . Showing that the value of  $post(v)$  has to be set before  $post(u)$  is left as an exercise.
  - (b) Assumption. Assume correctness for any  $pre(v) - pre(u) < t < post(u)$ .
  - (c) Step. Consider  $t > 1$  such  $pre(v) - pre(u) = t$ . Since  $t > 1$  there is must to be vertex  $w$  for which  $pre(u) < pre(w) < pre(v) = t$ . Splits again:
    - i. Either  $post(w) > pre(v)$ . Observes that:

$$pre(v) - pre(w) < pre(v) - pre(u) = t$$

and also:

$$pre(w) - pre(u) < pre(v) - pre(u) = t$$

Therefore by the induction assumption:

$$[pre(v), post(v)] \subset [pre(w), post(w)] \subset [pre(u), post(u)]$$

and in addition  $w$  is a descendant of  $u$  and  $v$  is a descendant of  $w$ . Hence  $v$  is a descendant of  $u$  and the third case holds.

- ii. Or  $post(w) < pre(v)$  for any  $w$  satisfies  $pre(u) < pre(w) < pre(v)$ . That means that any call for **Explore**( $G, w$ ) at line 6 (over suited  $w$ 's) returned, and at time  $t$  a new child of  $u$  has been discovered<sup>1</sup>, namely  $v$  is a direct child of  $u$  and we back to the base.

2. Or,  $post(u) < pre(v)$ , and by definition,  $pre(u) < post(u) < pre(v) < post(v)$  and thus the intervals  $[pre(u), post(u)]$  and  $[pre(v), post(v)]$  are disjoint.

Showing that  $v$  is not a descendant of  $u$  can be proved in similar manner to the above, by induction on  $pre(v) - post(u)$ . Completing the proof is left as an exercise.

□

*Corollary 10.1.1.* Vertex  $v$  is a proper descendant of vertex  $u$  in the depth-first forest for a (directed or undirected) graph  $G$  if and only if  $pre(u) < pre(v) < post(v) < post(u)$ .

## 10.2 DFS on DAGS

the pre is well calibrated on DAGS.

<sup>1</sup>otherwise we get a contradiction for  $post(u) > t$