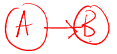
Topological Sort

(an application of DFS)

1

Topological sort



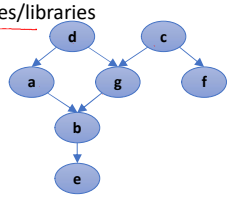
- We have a **set of tasks** and a **set of dependencies (precedence constraints)** of form "task A must be done before task B"
- **Topological sort**: An ordering of the tasks that conforms with the given dependencies
- **Goal**: Find a topological sort of the tasks or decide that there is no such ordering

2

2

Examples

- **Scheduling**: When scheduling *task graphs* in distributed systems, usually we first need to sort the tasks topologically
  ...and then assign them to resources (the most efficient scheduling is an NP-complete problem)
- Or during compilation to order modules/libraries



Tasks d and c must be completed before a, g and f

Tasks a and g must be completed before b

Tasks b must be completed before e

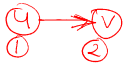
- **Resolving dependencies**: *apt-get* uses topological sorting to obtain the admissible sequence in which a set of Debian packages can be installed/removed

3

3

Topological sort more formally

- Suppose that in a **directed** graph **G = (V, E)** vertices **V** represent tasks, and each edge (**u**, **v**)∈**E** means that task **u** must be done before task **v**



- What is an ordering of vertices 1, ..., |**V**| such that for every edge (**u**, **v**), **u** appears before **v** in the ordering?
- Such an ordering is called a **topological sort of G**
- Note: there can be multiple topological sorts of G

4

4

Topological sort more formally

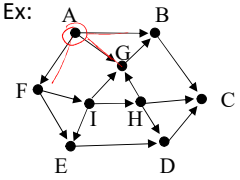
- Is it possible to execute all the tasks in **G** in an order that respects all the precedence requirements given by the graph edges?
- The answer is "**yes**" *if and only if* the directed graph **G** has **no cycle**! (otherwise we have a **deadlock**)
- Such a **G** is called a Directed Acyclic Graph, or just a **DAG**

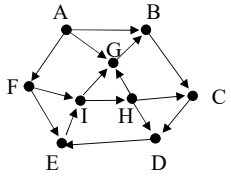
5

5

Topological Sort

- Directed graph G.
- Rule: if there is an edge u → v, then u must come before v.
- Ex:



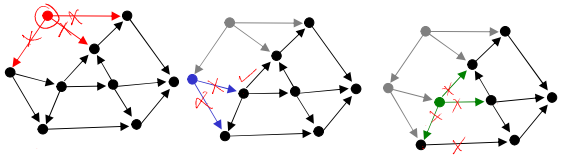


6

6

Intuition

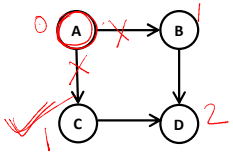
- Cycles make topological sort impossible.
- Select any node with no in-edges
  - print it
  - delete it
  - and delete all the edges leaving it
- Repeat
- What if there are some nodes left over?
- Implementation? Efficiency?

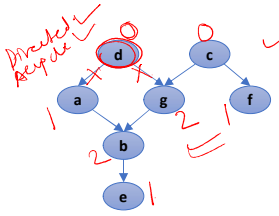


7

7

Example





A topological sort would give A, B, C, D.

Topological Sort- d,c,a,g,f,b,e
Or c,d,f,a,g,b,e

8

8

Topological Sort Code

```
top_sort(Adjacency Matrix adj, Array ts)
{
     n = adj.last
     k = n  //assume k is global
     for i=1 to n
         visit[i] = false
     for i=1 to n
         if (!visit[i])
            top_sort_rec(adj, i, ts)
}
```

```
top_sort_rec(Adjacency Matrix adj,  Vertex start, Array ts)
{
   visit[start] = true
   trav = adj[start] //trav points to a LL

   while (trav != null)
   {
      v = trav.ver
      if (!visit[v])
         top_sort_recurs(adj, v, ts);
      trav = trav.next
    }

   ts[k] = start, k=k-1
}
```

9

9

Thank You !!!

10

10