Algorithms and Data Structures
2015-2016
Claudio Sacerdoti Coen
--
Graph Algorithms

DAGs as Dependency Graphs

Tasks to be completed can be arranged into DAGs
(Direct Acyclic Graphs)

1) vertices are tasks
2) an edge (v,w) means that v must be completed before w
   (or is required by w or ...)

Question : can we use a general directed graph instead?

Exercise : draw the graph to cook your favourite meal

Topological Sorting

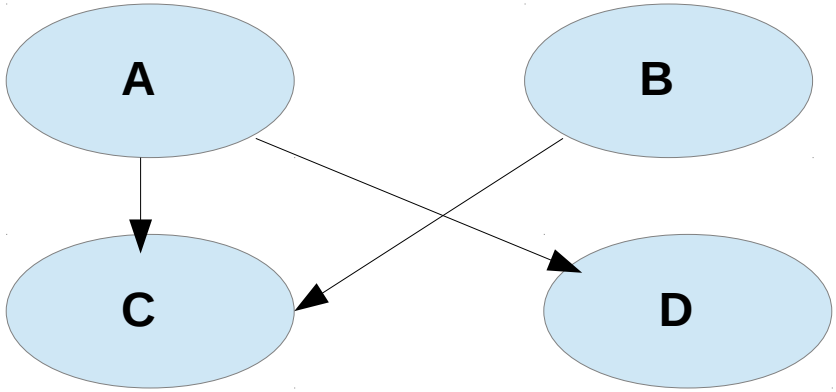
Given a DAG (V,E), (v1, …, vn) is a topological sorting of the nodes in V iff

1) {v1, …, vn} = V
2) for each i < j, there is no path from vj to vi

Tasks can be executed in topological order without violating the dependency graph.

Question : can a DAG be topologically sorted in two different ways?

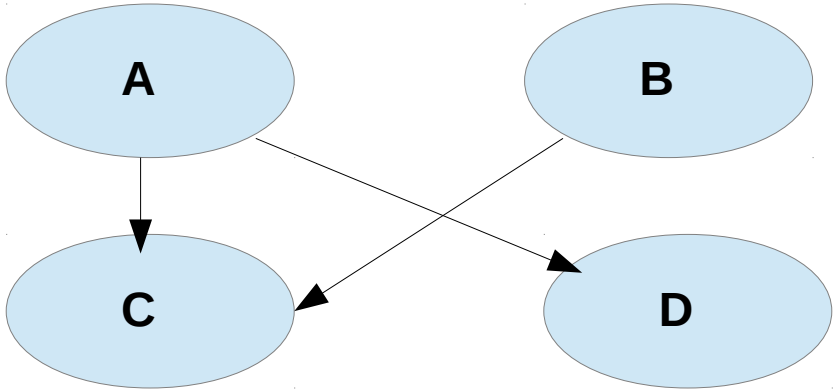
Topological Sorting



{ A, B, C, D }, { B, A, D, C }, { A, D, B, C } are all valid topological orders

Question: are there more?

Topological Sorting



Topological sorting algorith – ideas :

1. nodes are added one by one in front of the result list
2. a node is to be added only after adding all nodes that are reachable from it

Question: how to know what nodes are reachable from a given one?

Topological Sorting

```
TopoSort(G) =
  L = empty_list()
  for u in vertices(G)
    modified_dfs(G,u,L)
  return L

  modified_dfs(G,u,L) =
    mark(u)
    for v in AdjSet(G,u)
      If not marked(v)
          modified_dfs(G,v,L)
      add(L,v)  // v is added to L after all nodes
                // reachable from v
```

Topological Sorting

Exercise : run the previous algorithm on the graph of Slide 5 multiple times, changing only the order in which vertices(G) returns the node.