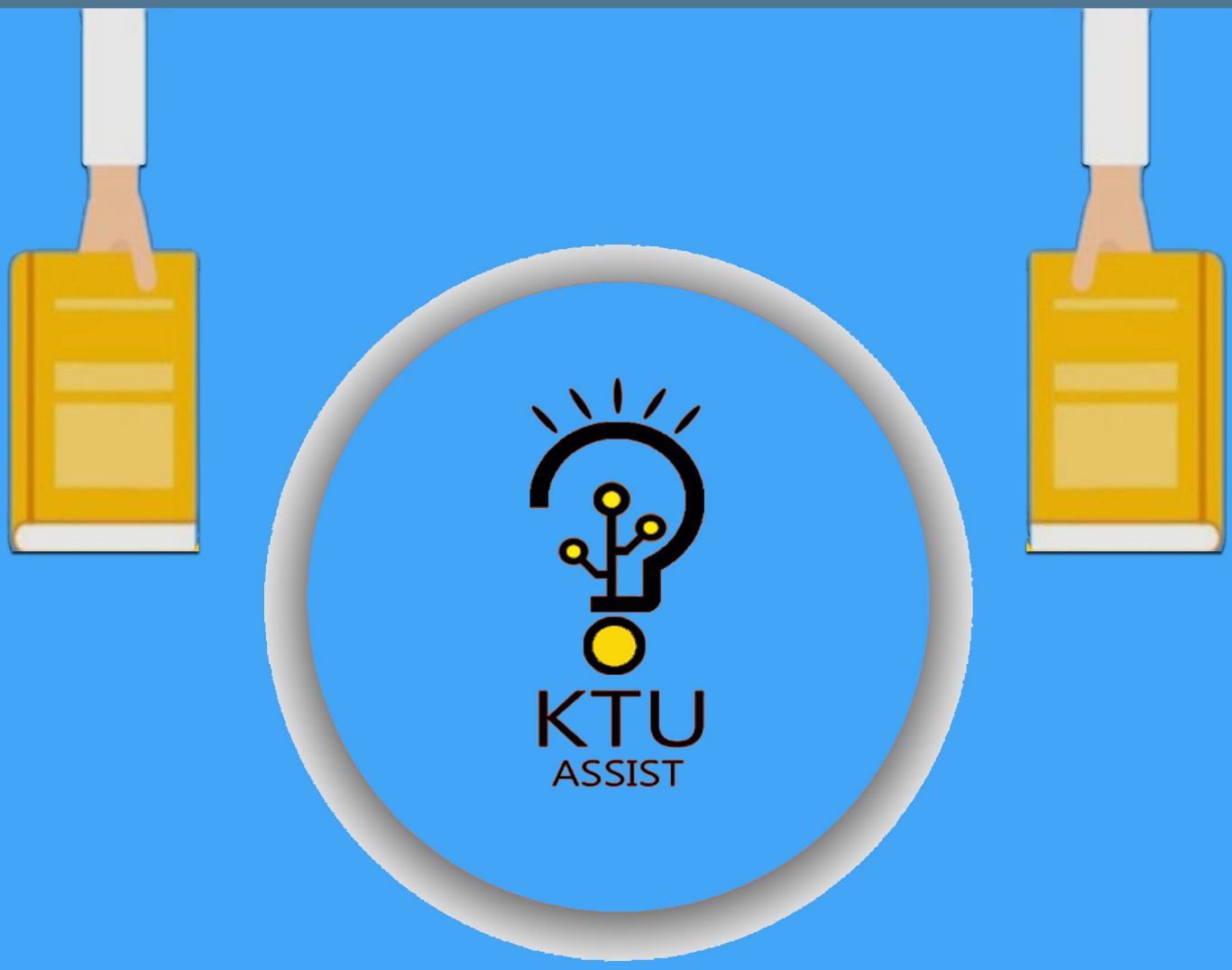
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

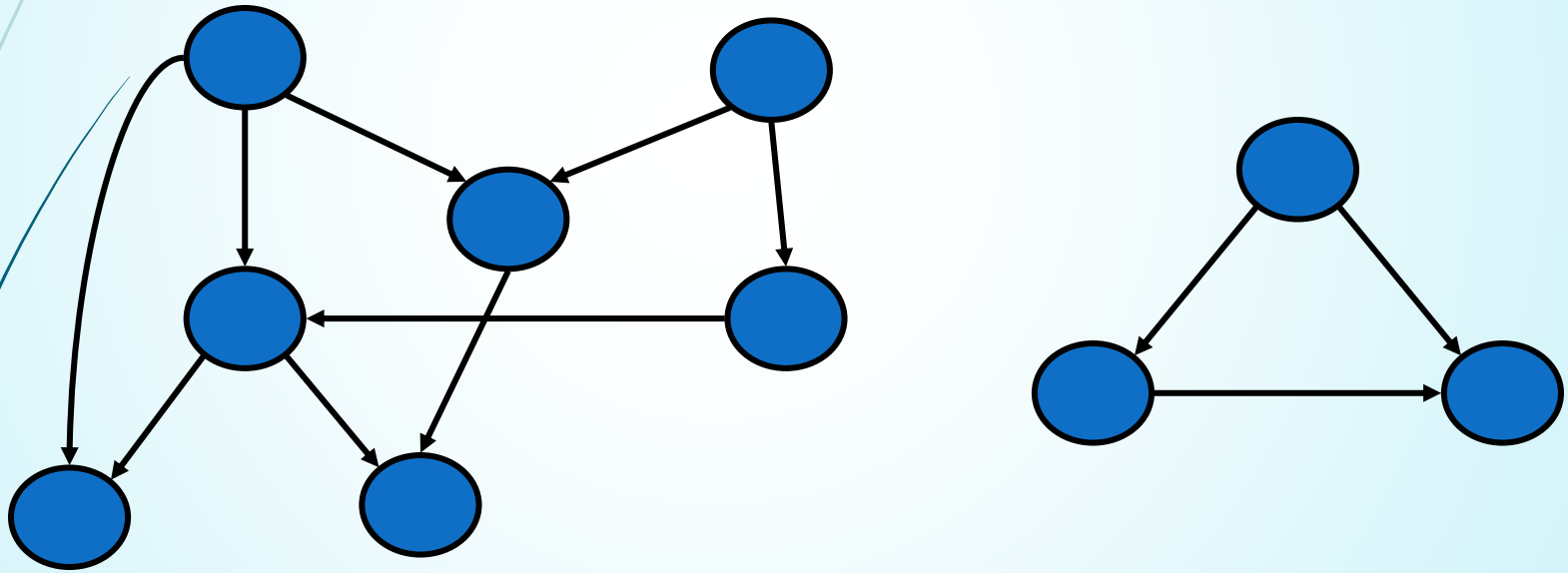
STUDY MATERIALS



Topological Sort

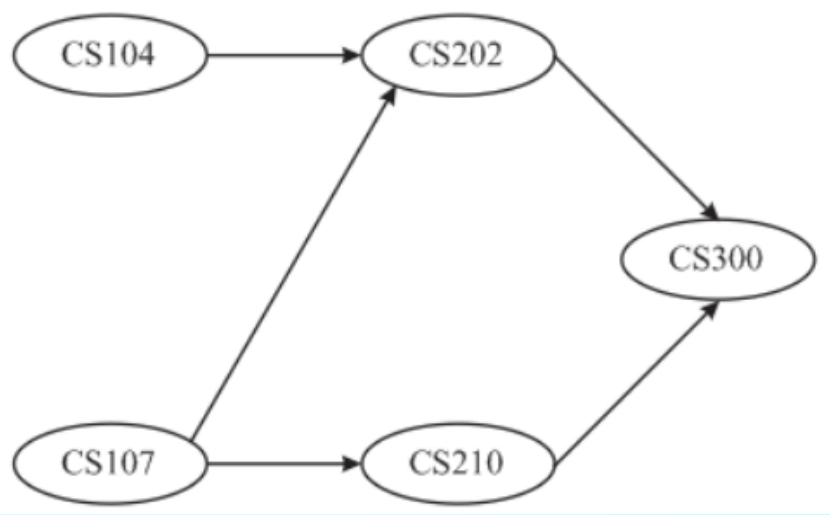
Directed Acyclic Graphs

- A *directed acyclic graph* or *DAG* is a directed graph with no directed cycles:

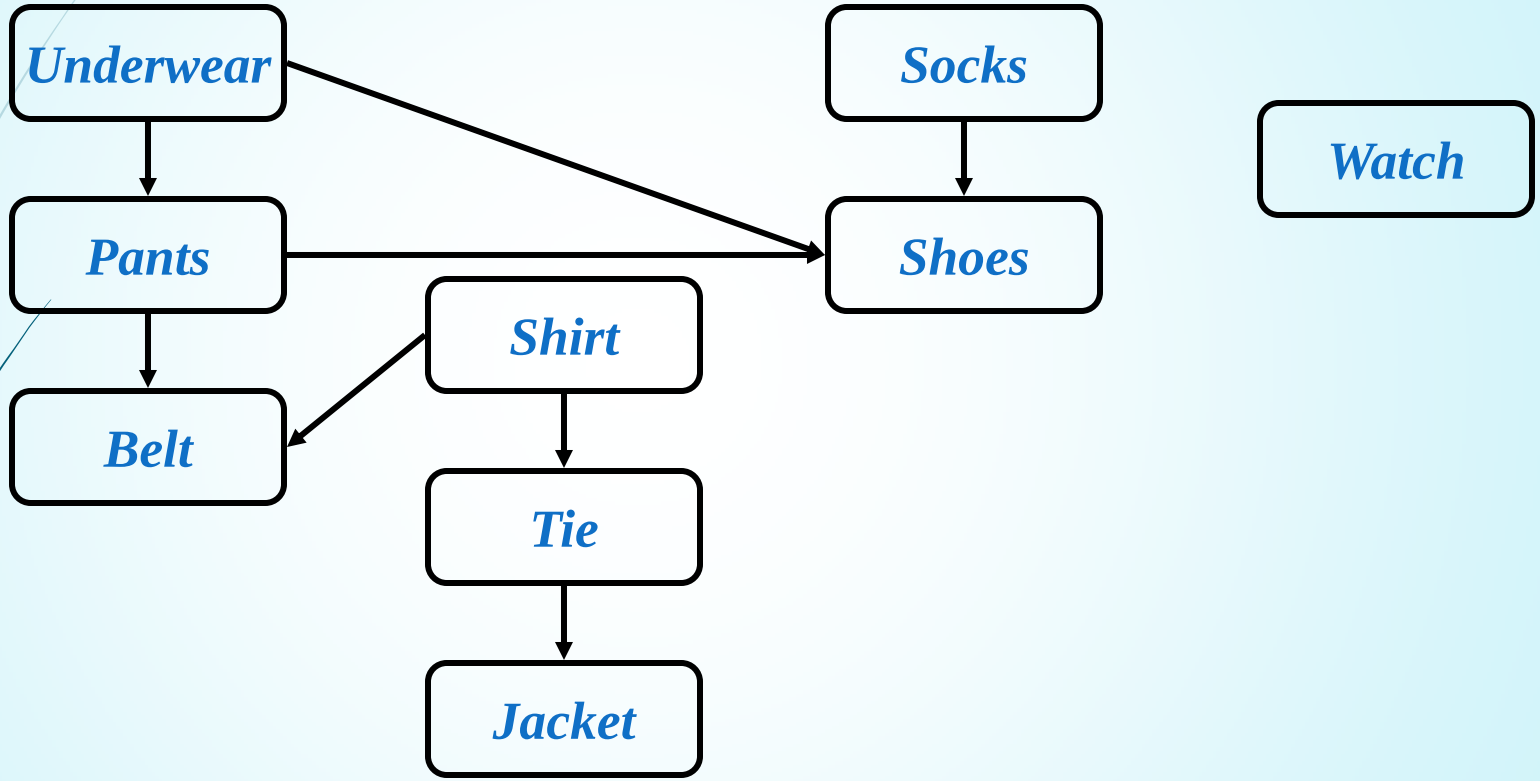


Topological Sort

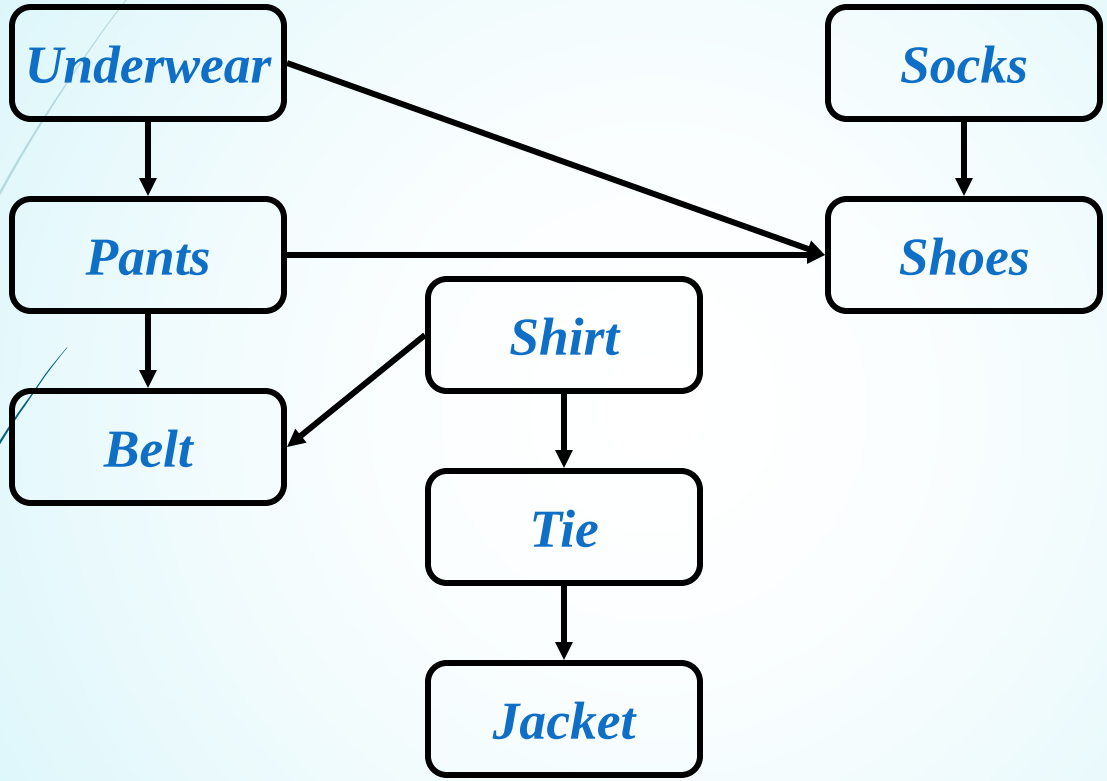
- *Topological sort* of a DAG:
  - Linear ordering of all vertices in graph G such that vertex *u* comes before vertex *v* if edge $(u, v) \in G$
  - Topological Sorting for a graph is not possible if the graph is not a DAG



Getting Dressed



Getting Dressed



Topological Sort Algorithm

```
Topological-Sort()
{
    Run DFS
    When a vertex is finished, output it
    Output vertices in reverse topological
      order
}
```

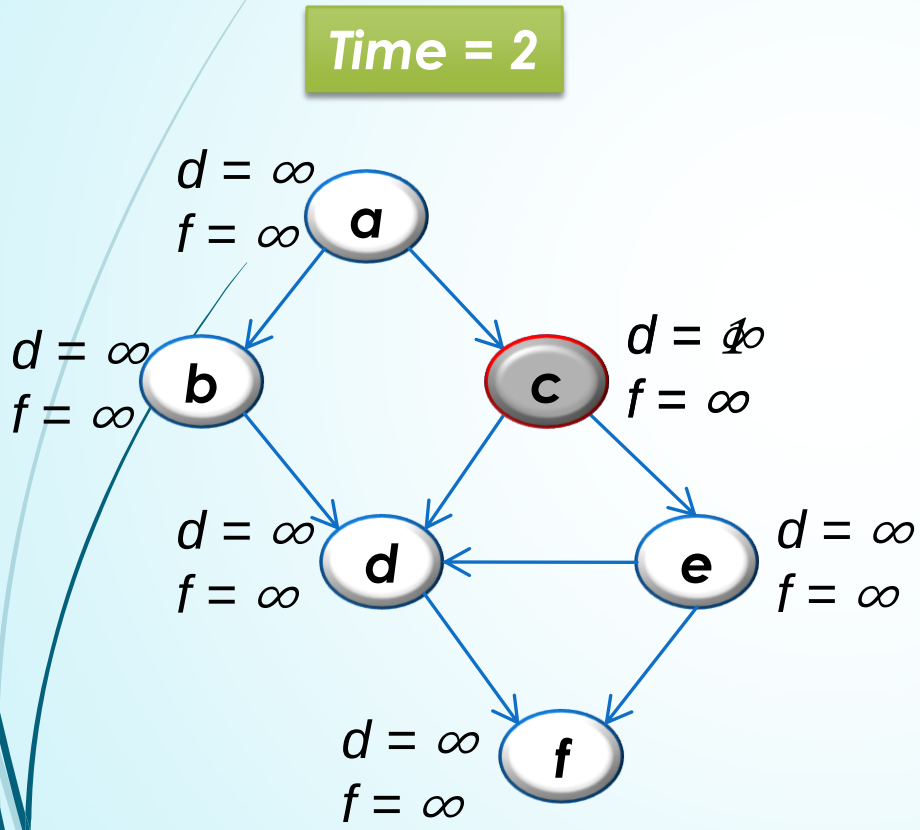
- Time: O(V+E)

Topological sort

1) *Call DFS(**G**) to compute the finishing times **f**[**v**]*

*Let's say we start the DFS from the vertex **c***

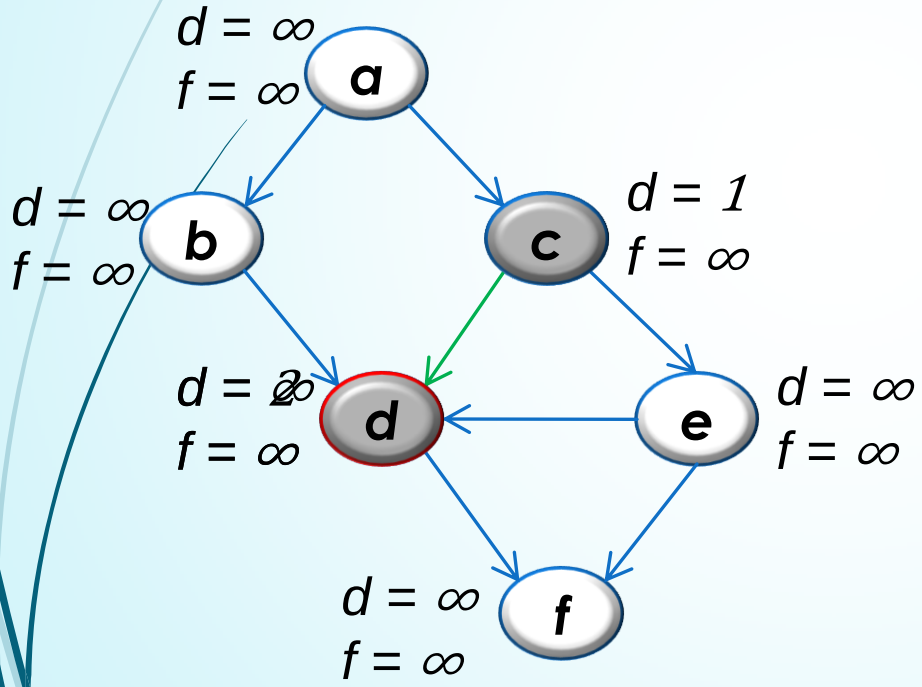
*Next we discover the vertex **d***



Topological sort

1) *Call DFS(**G**) to compute the finishing times **f**[**v**]*

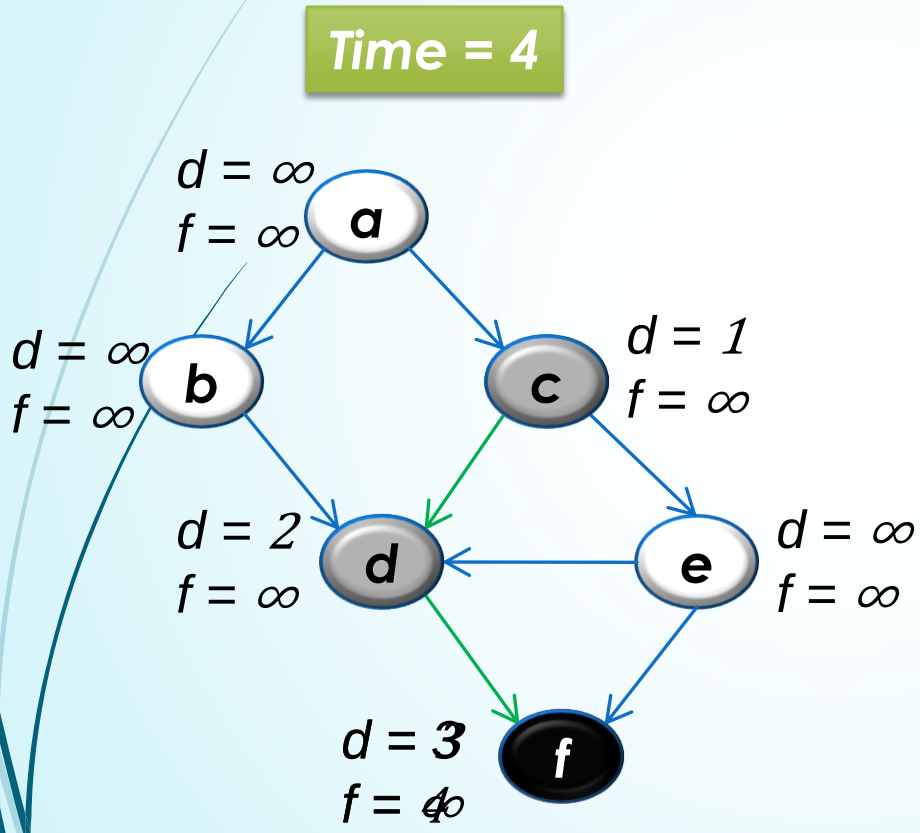
Time = 3



*Let's say we start the DFS from the vertex **c***

*Next we discover the vertex **d***

Topological sort



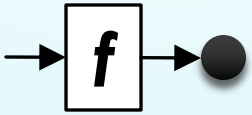
1) *Call DFS(**G**) to compute the finishing times **f**[**v**]*

2) *as each vertex is finished, insert it onto the **front** of a linked list*

*Next we discover the vertex **d***

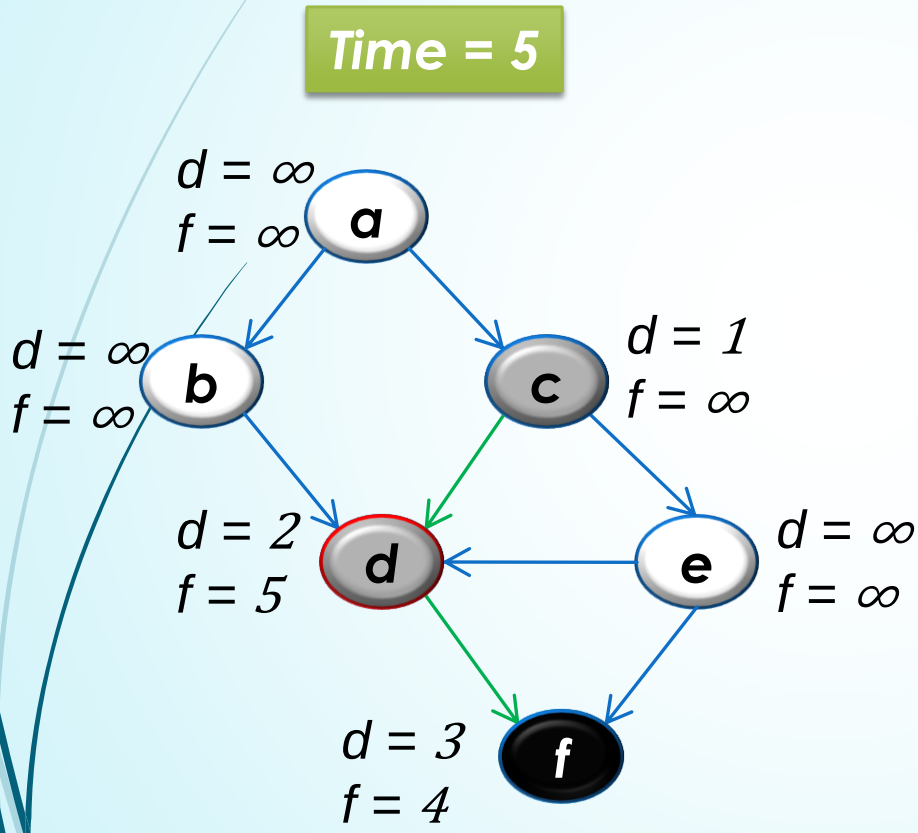
*Next we discover the vertex **f***

f** is done, move back to **d



Topological sort

1) *Call DFS(**G**) to compute the finishing times **f**[**v**]*



*Let's say we start the DFS from the vertex **c***

*Next we discover the vertex **d***

*Next we discover the vertex **f***

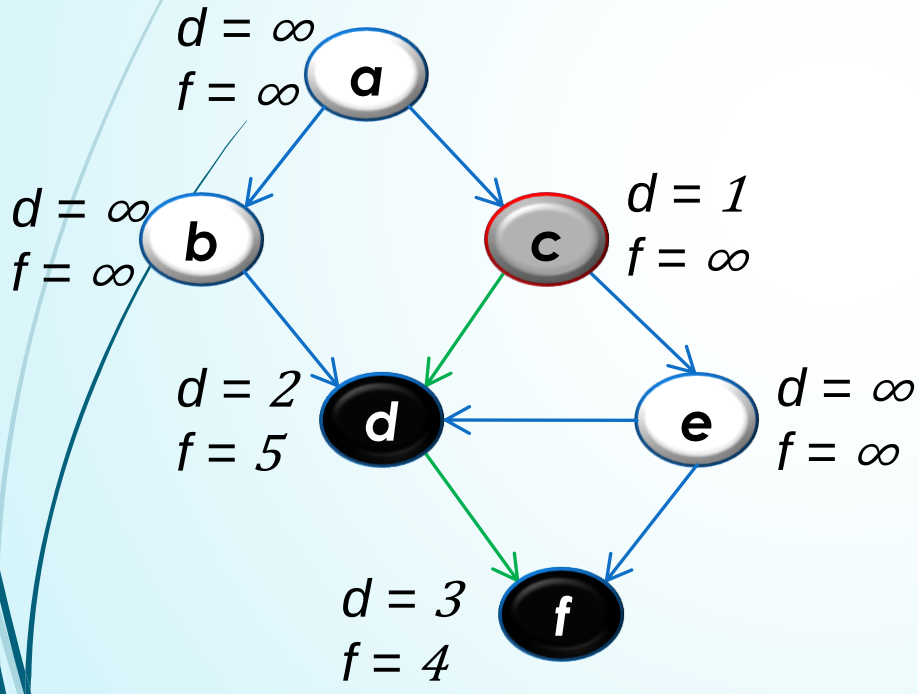
f** is done, move back to **d

d** is done, move back to **c

Topological sort

1) *Call DFS(**G**) to compute the finishing times **f**[**v**]*

Time = 6



*Let's say we start the DFS from the vertex **c***

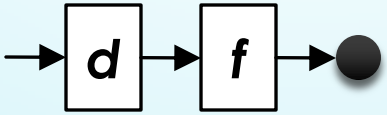
*Next we discover the vertex **d***

*Next we discover the vertex **f***

f** is done, move back to **d

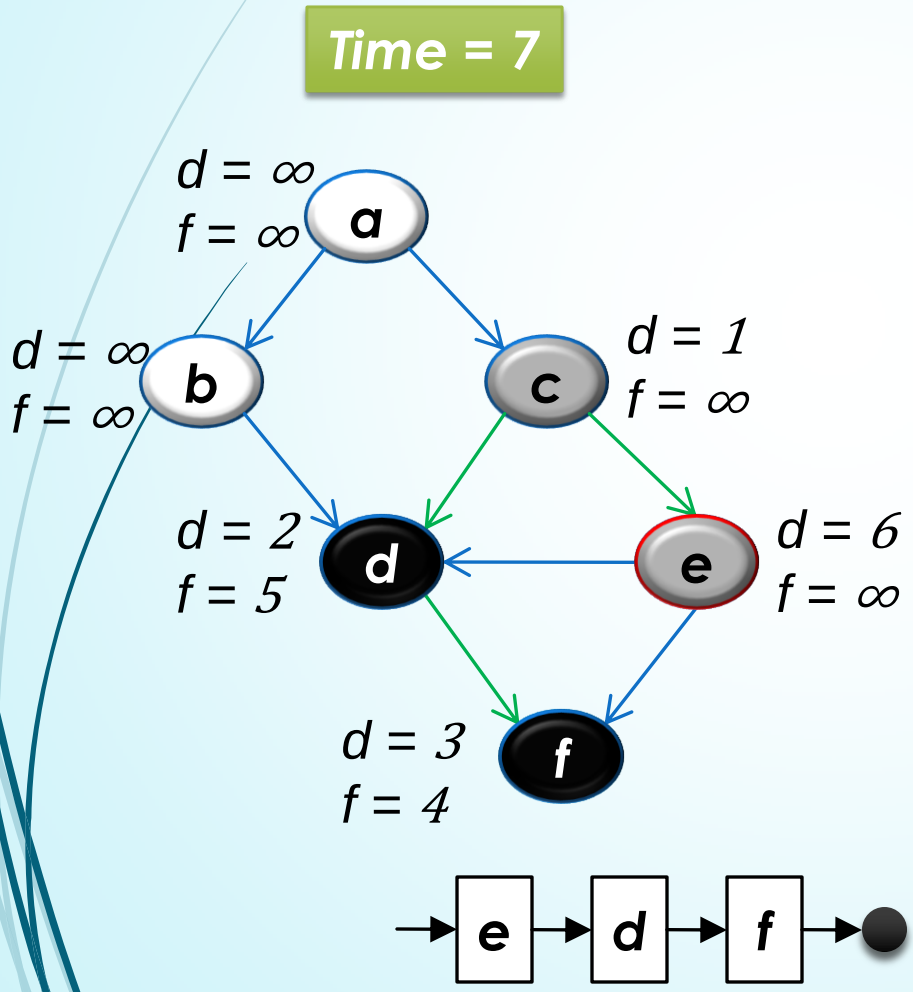
d** is done, move back to **c

*Next we discover the vertex **e***



Topological sort

1) *Call DFS(**G**) to compute the finishing times **f**[**v**]*



*Let's say we start the DFS from the vertex **c***

*Next we discover the vertex **d***

*Next we discover the vertex **f***

*Both edges from **e** are **cross edges***

d** is done, move back to **c

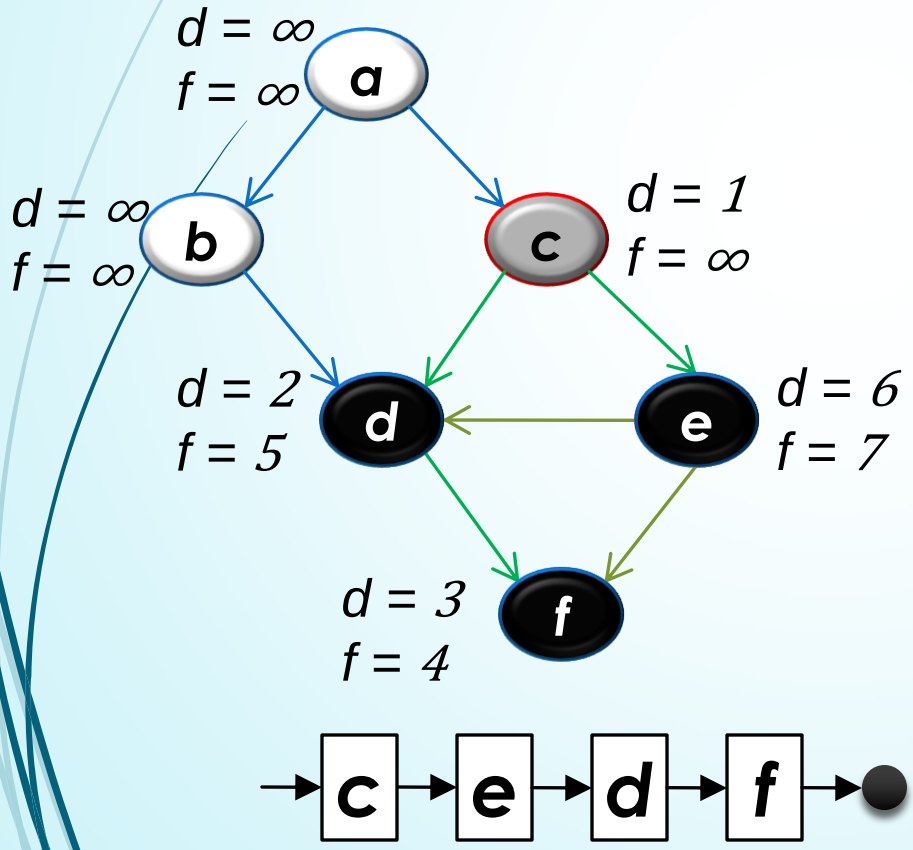
*Next we discover the vertex **e***

e** is done, move back to **c

Topological sort

1) *Call DFS(**G**) to compute the finishing times **f**[**v**]*

Time = 8



*Let's say we start the DFS from the vertex **c***

*Just a note: If there was (**c**,**f**) edge in the graph, it would be classified as a **forward edge** (in this particular DFS run)*

d** is done, move back to **c

*Next we discover the vertex **e***

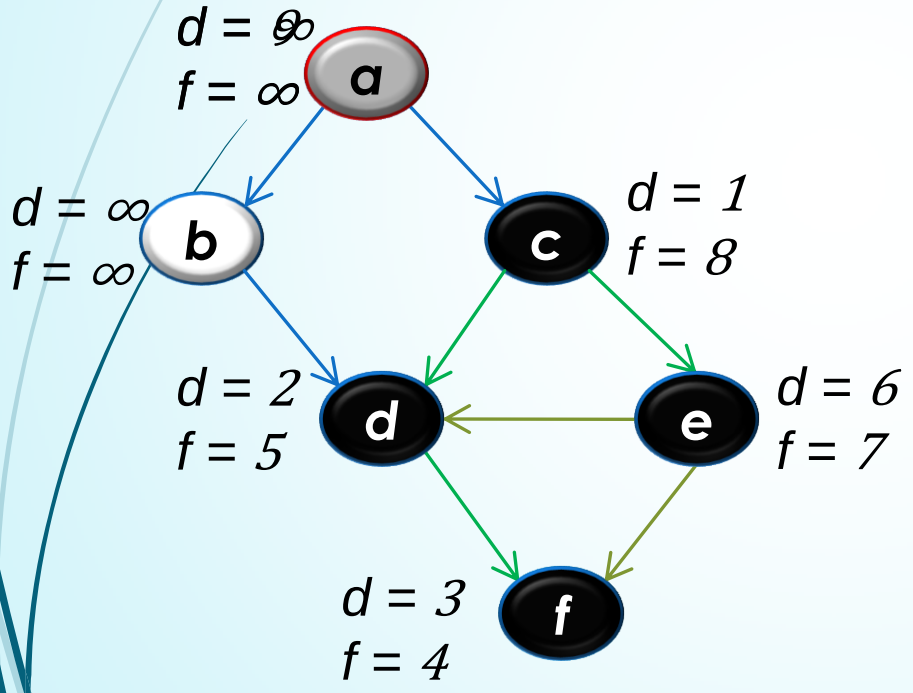
e** is done, move back to **c

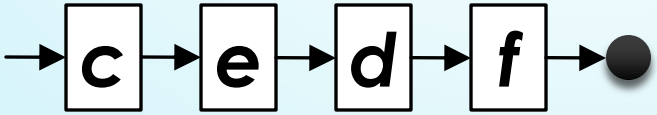
***c** is done as well*

Topological sort

1) *Call DFS(**G**) to compute the finishing times **f**[**v**]*

Time = 10





*Let's now call DFS visit from the vertex **a***

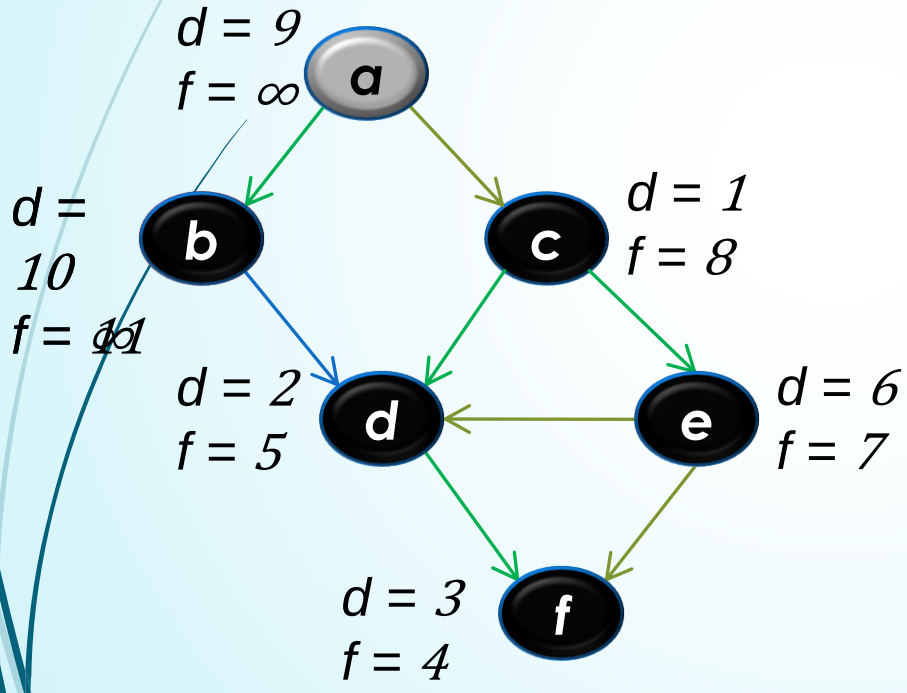
*Next we discover the vertex **c**, but **c** was already processed => (**a**,**c**) is a cross edge*

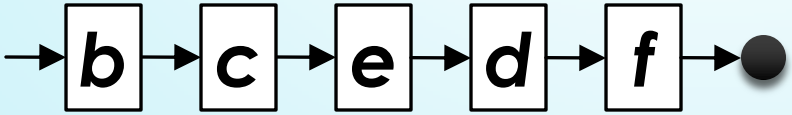
*Next we discover the vertex **b***

Topological sort

1) *Call DFS(**G**) to compute the finishing times **f**[**v**]*

Time = 11





*Let's now call DFS visit from the vertex **a***

*Next we discover the vertex **c**,*
*but **c** was already processed => (**a**,**c**) is a cross edge*

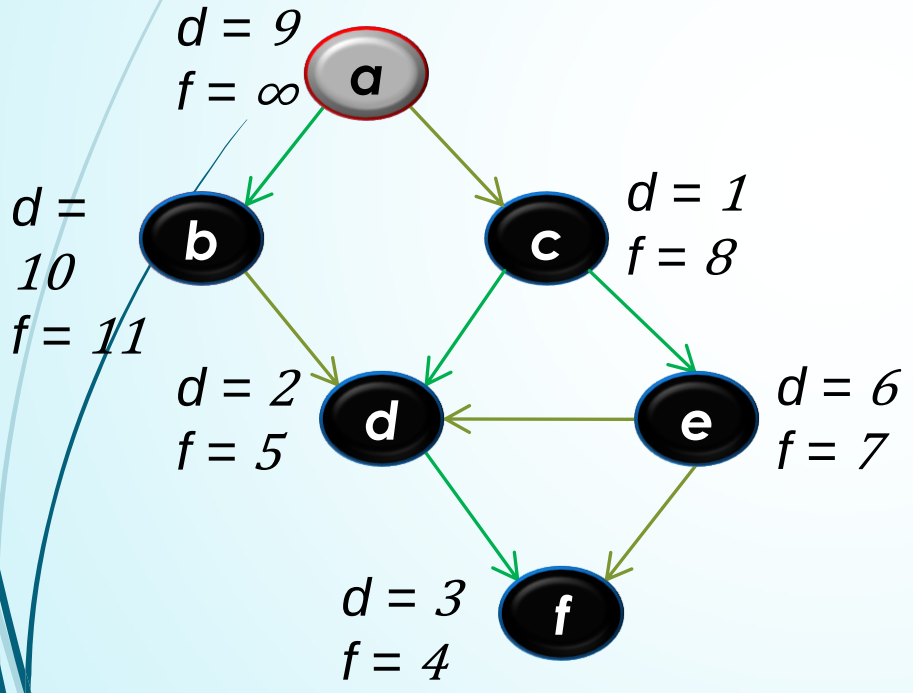
*Next we discover the vertex **b***

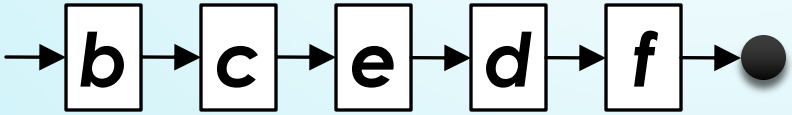
b** is done as (**b**,**d**) is a cross edge => now move back to **c

Topological sort

1) *Call DFS(**G**) to compute the finishing times **f**[**v**]*

Time = 12





*Let's now call DFS visit from the vertex **a***

*Next we discover the vertex **c**, but **c** was already processed => (**a**,**c**) is a cross edge*

*Next we discover the vertex **b***

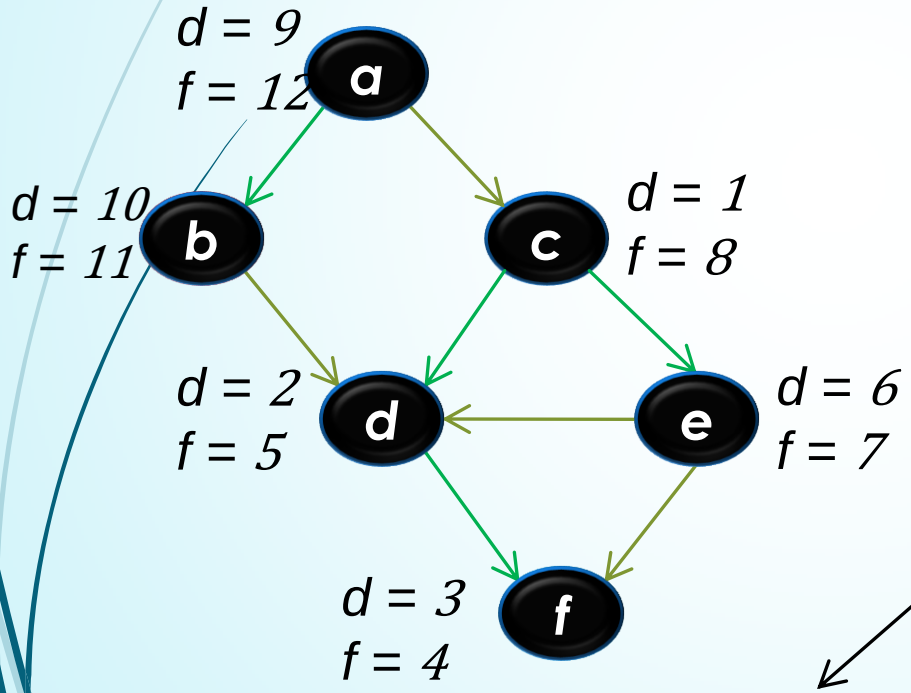
b** is done as (**b**,**d**) is a cross edge => now move back to **c

***a** is done as well*

Topological sort

1) Call DFS(**G**) to compute the finishing times **f**[**v**]

Time = 13



WE HAVE THE RESULT!

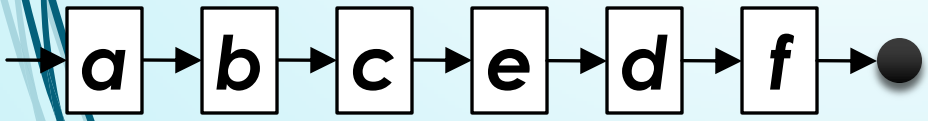
3) return the linked list of vertices

but **c** was already

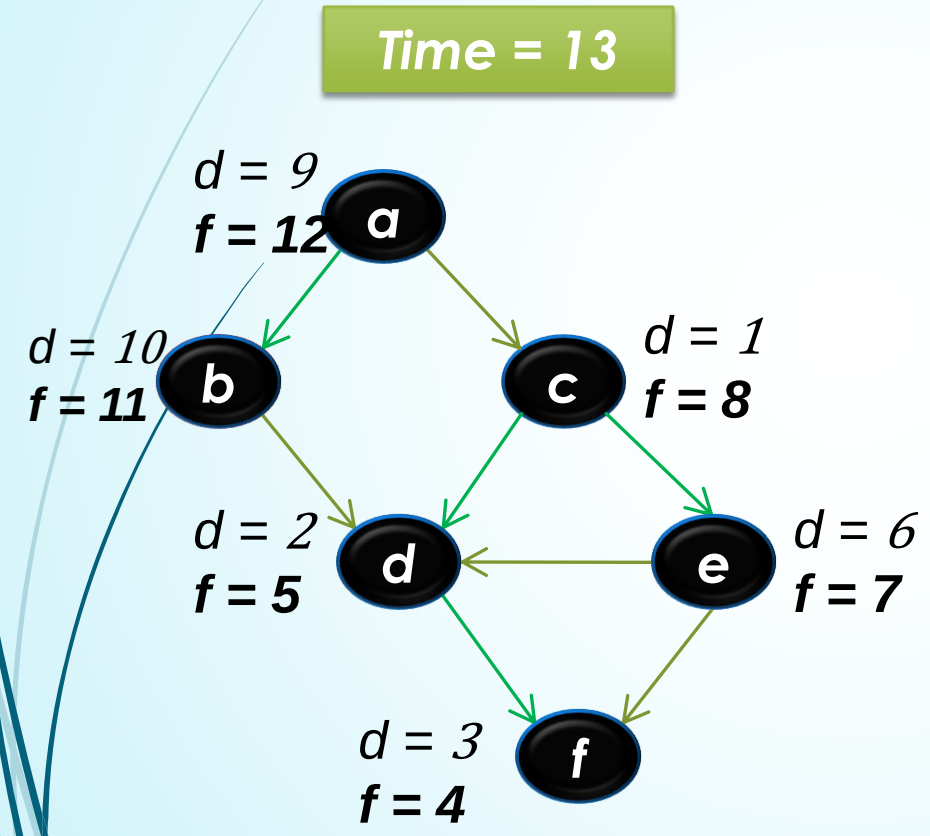
Next we discover the

b is done as (**b**,**d**) is a cross edge => now

a is done as well



Topological sort



*The linked list is sorted in **decreasing** order of finishing times **f**[]*

END