L06.02
Depth-First-Search (DFS), Topological sort

50.004 Introduction to Algorithms

Ioannis Panageas(ioannis@sutd.edu.sg)

CLRS Ch 22.3 – 22.4

ISTD, SUTD

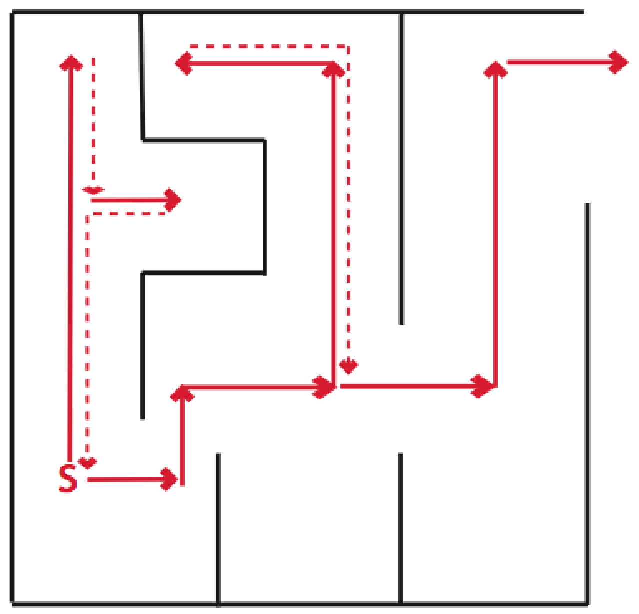
Based on slides by Dr. Simon LUI

1. DFS

Depth-first search

Idea: Similar to exploring a maze

- follow path until you get stuck
- backtrack along breadcrumbs
- recursively explore



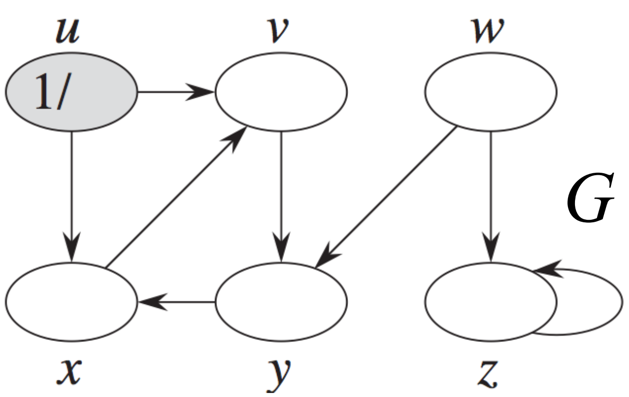
The DFS Algorithm

```
parent  = {s: None}

DFS-visit (V, Adj, s):
    for v in Adj [s]:
        if v not in parent:
            parent [v] = s
            DFS-visit (V, Adj, v)
```

The DFS algorithm (using color and timestamp)

```
DFS($G$)
1  for each vertex $u \in G.V$
2      $u.color = $ WHITE
3      $u.\pi = $ NIL
4  $time = 0$
5  for each vertex $u \in G.V$
6      if $u.color == $ WHITE
7          DFS-VISIT($G, u$)
```

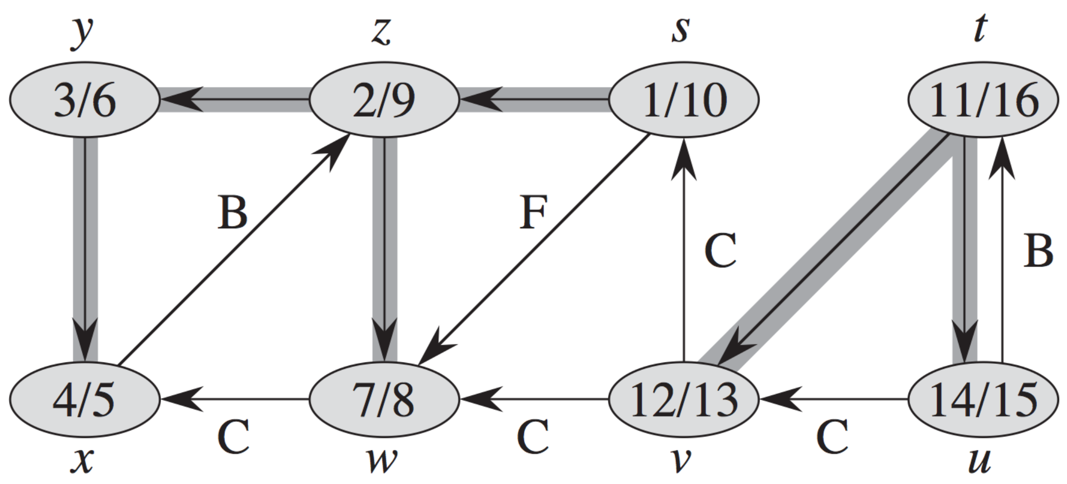


$G = (G.V, G.Adj)$

```
DFS-VISIT($G, u$)
1  $time = time + 1$                 // white vertex $u$ has just been discovered
2  $u.d = time$
3  $u.color = $ GRAY
4  for each $v \in G.Adj[u]$         // explore edge $(u, v)$
5      if $v.color == $ WHITE
6          $v.\pi = u$
7          DFS-VISIT($G, v$)
8  $u.color = $ BLACK                // blacken $u$; it is finished
9  $time = time + 1$
10 $u.f = time$
```

timestamps

Properties of DFS



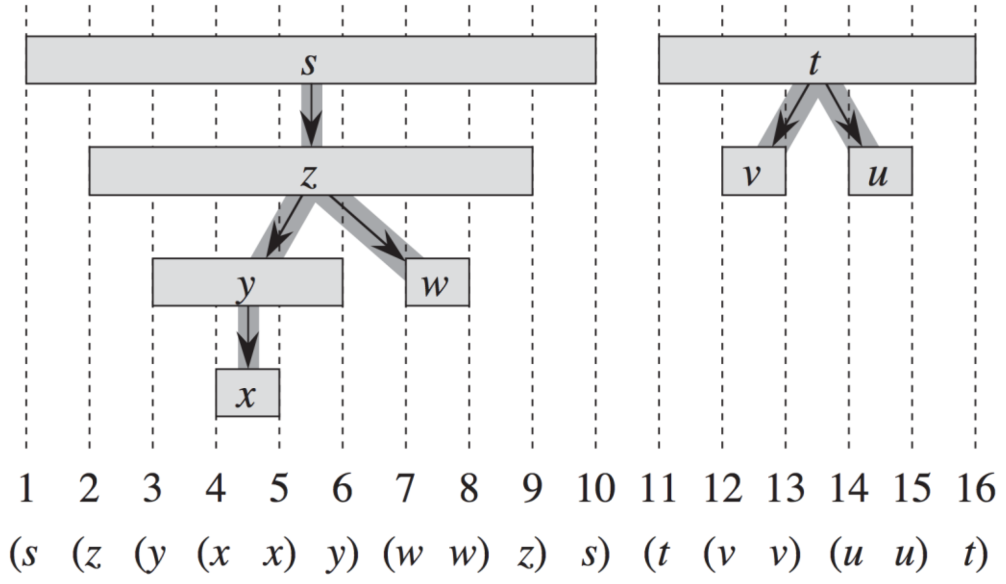
Type of edge $u \to v$

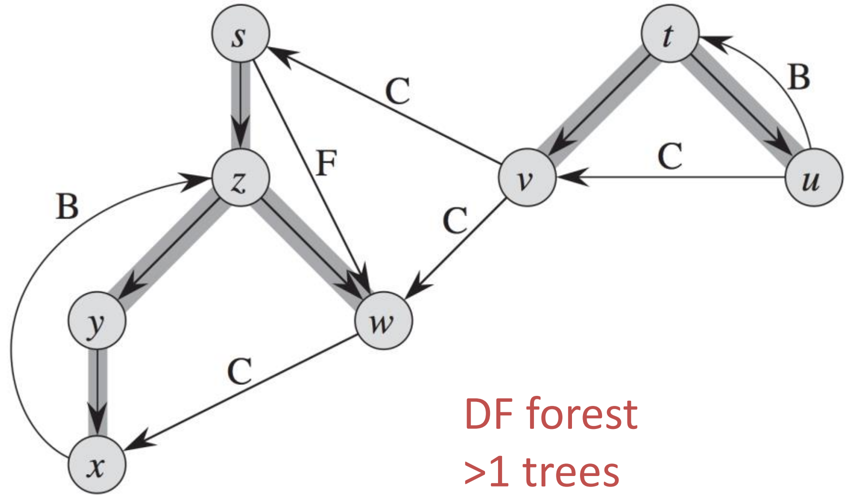
back: $v.d < u.d < u.f < v.f$

forward: $u.d < v.d < v.f < u.f$

cross: $v.d < v.f < u.d < u.f$

tree: first time v is visited





DF forest
>1 trees

Cycle detection

- Graph G has a cycle iff DFS **has a back edge**

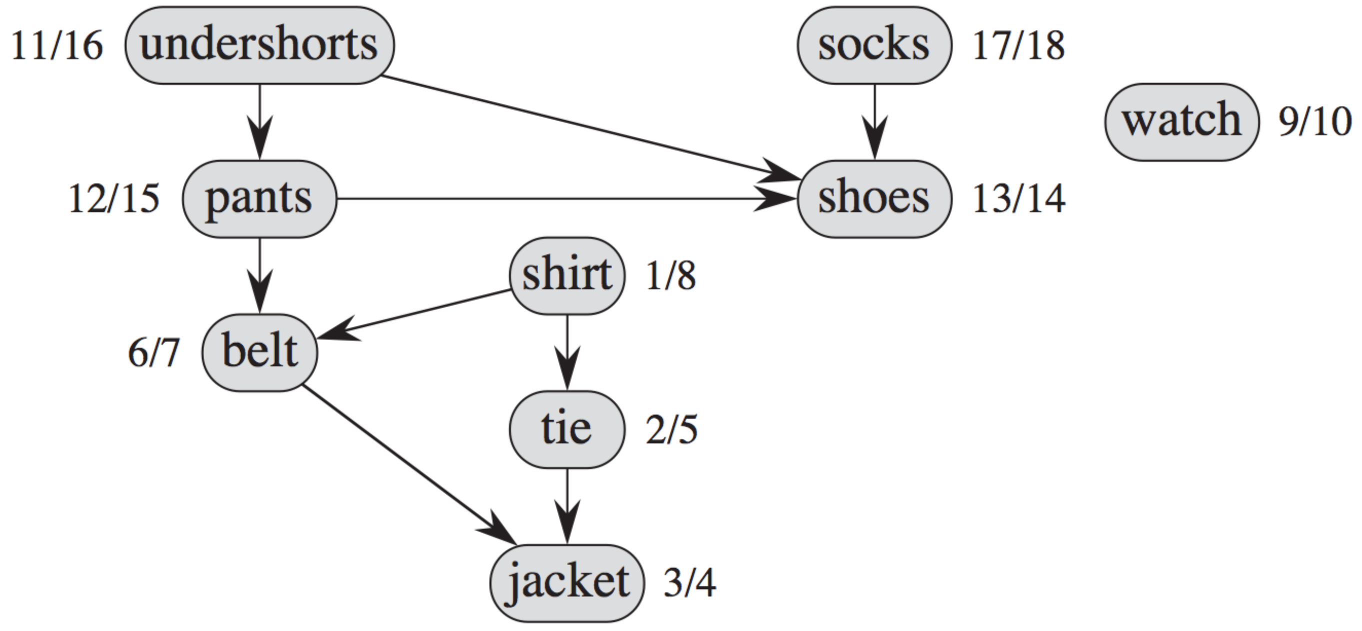
Directed Acyclic Graph = DAG

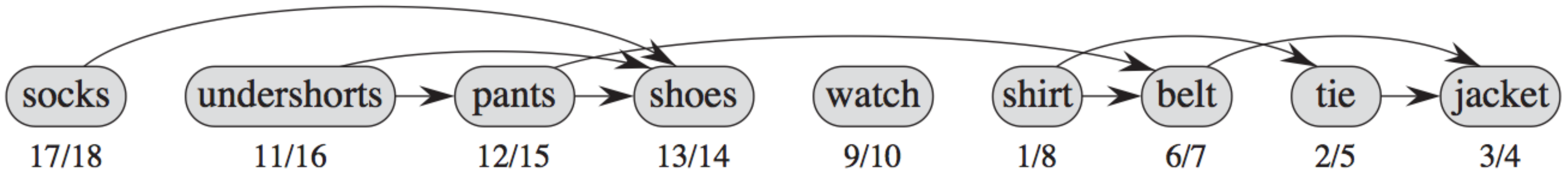
Topological sort

Topological sort of a DAG $G=(V,E)$

1. Run DFS(G), compute finishing times of nodes
2. Output the nodes in decreasing order of finishing times

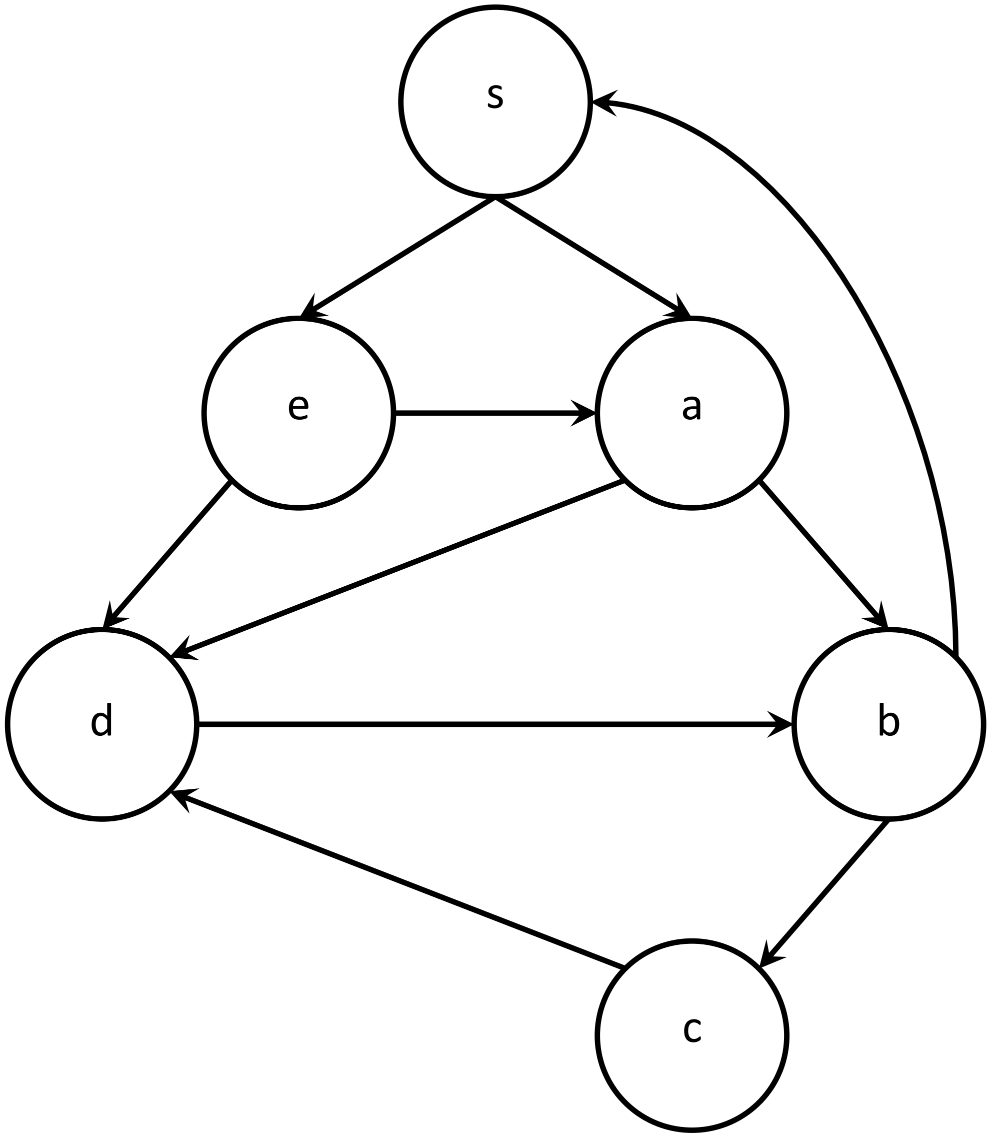
The Graph – relationship between clothing procedures



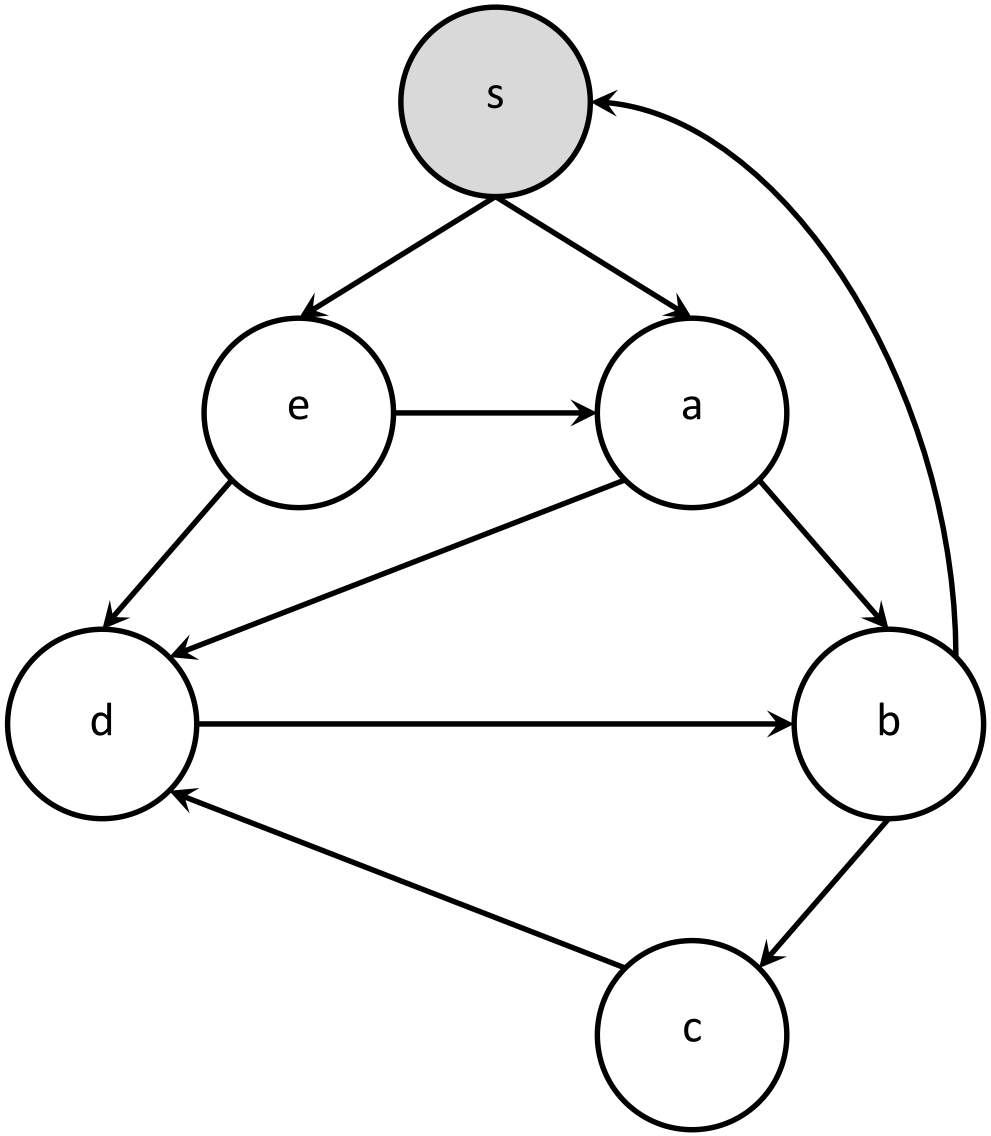


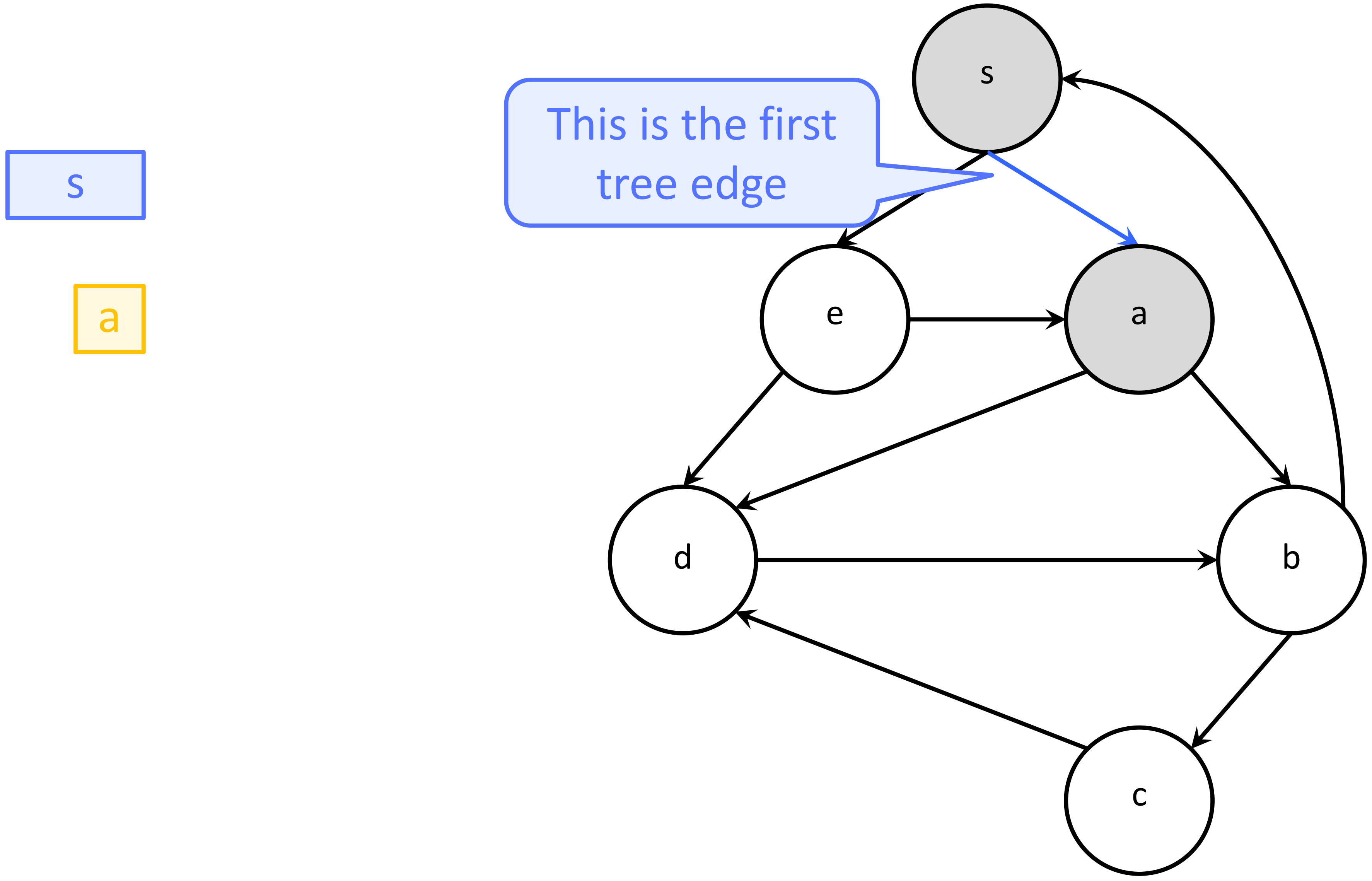
The Topological sort – a workable sequence of clothing

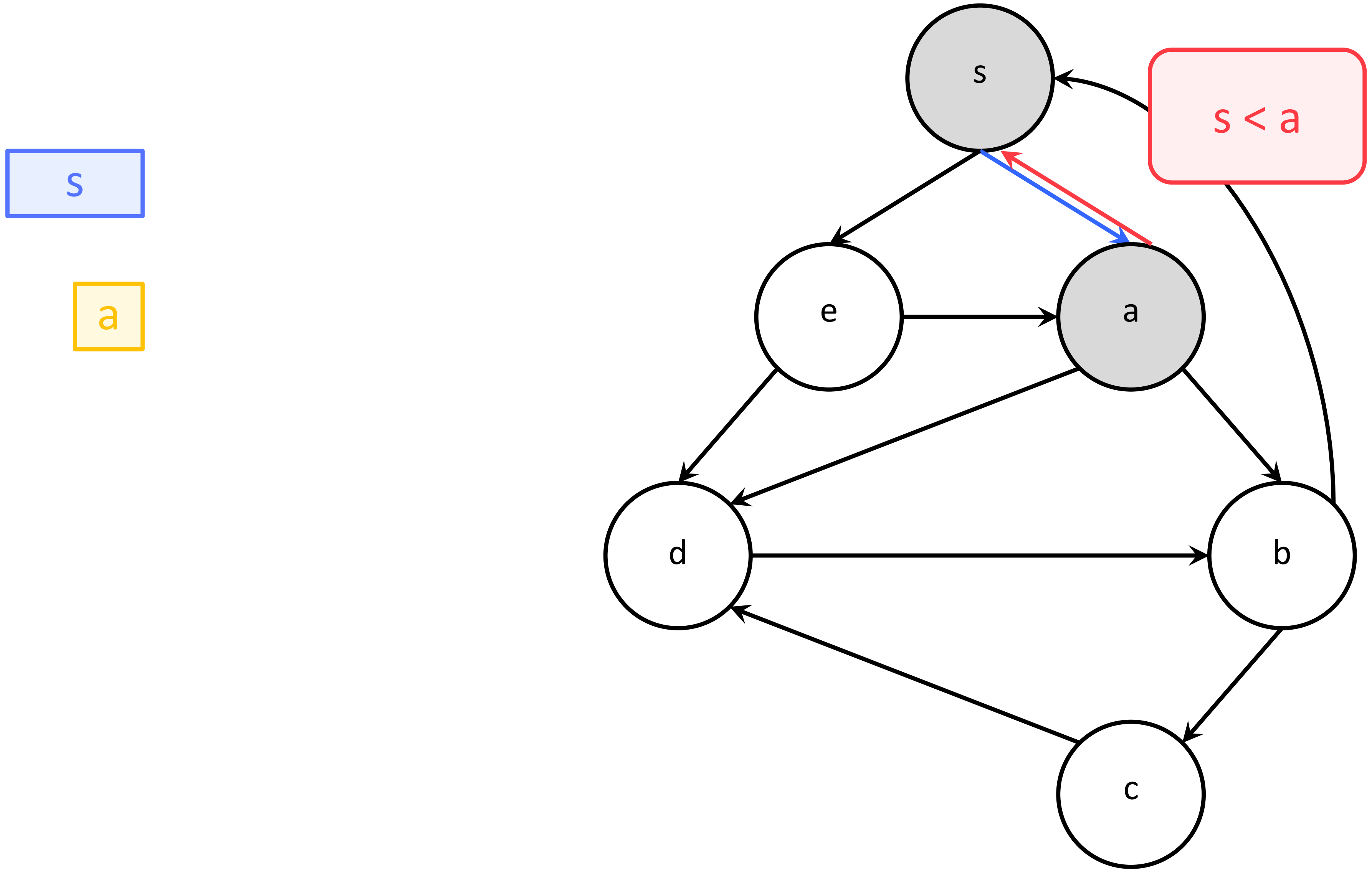
TOPOLOGICAL SORT

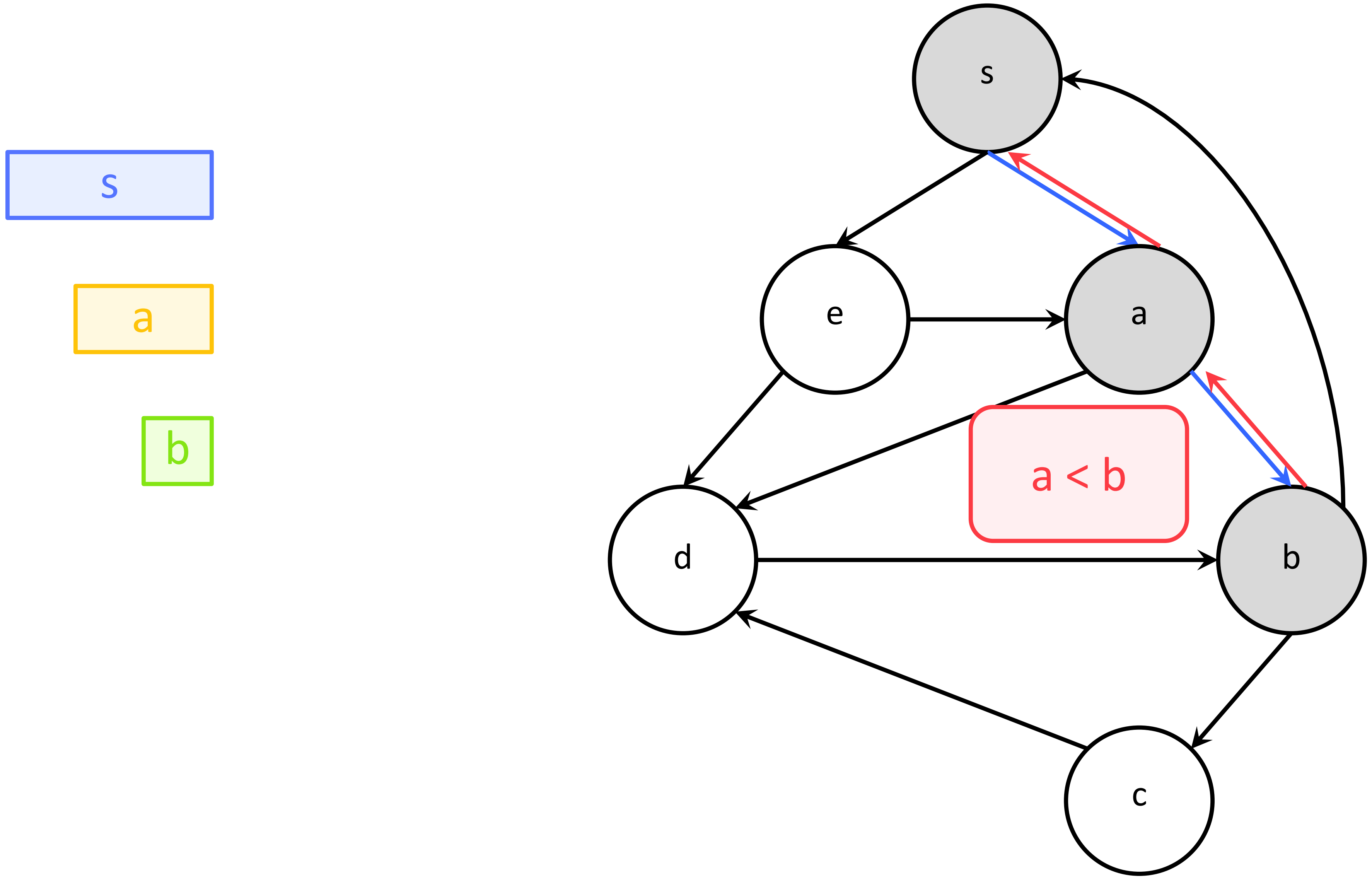


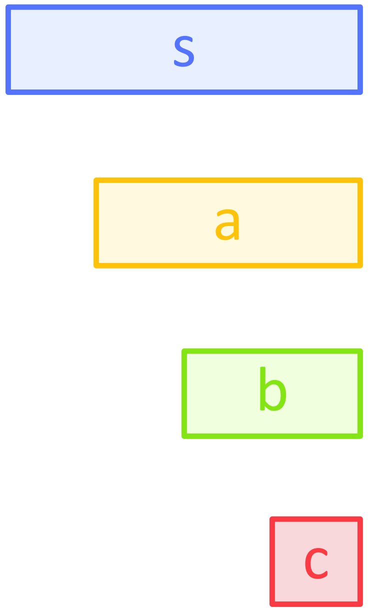
s

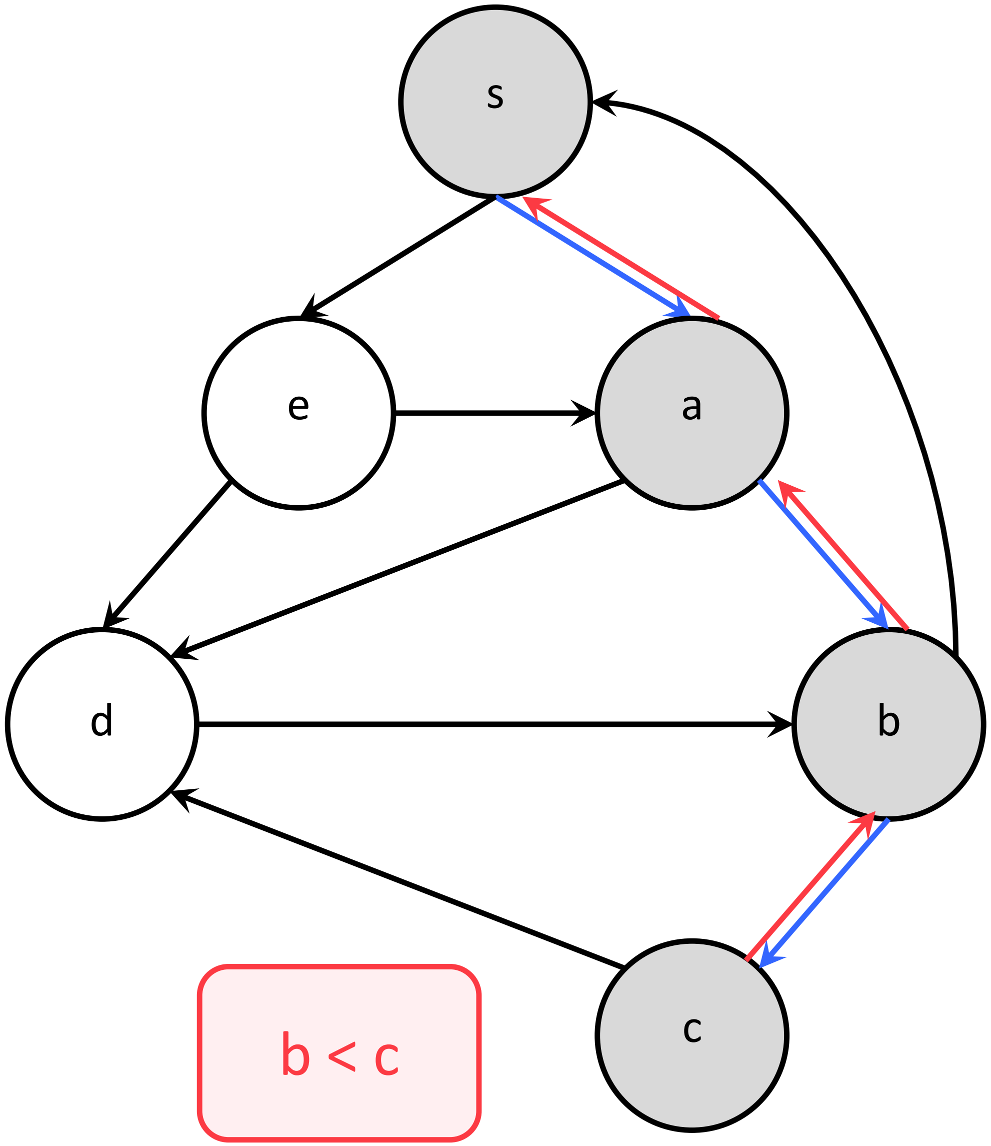



s
a
This is the first tree edge
s
e
a
d
b
c

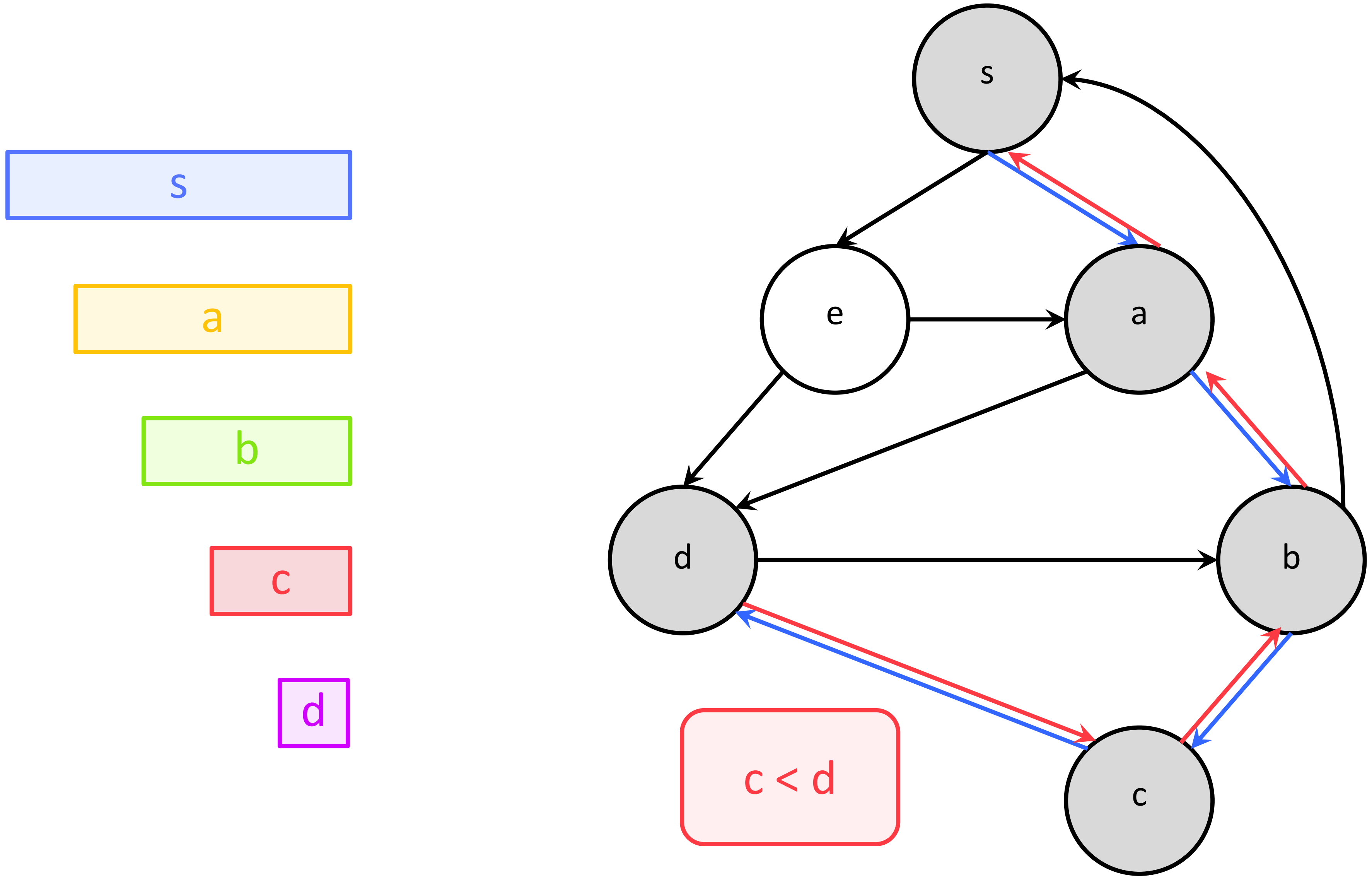

s
a
s
e
a
d
b
c
s < a


s
a
b
s
e
a
d
a < b
b
c





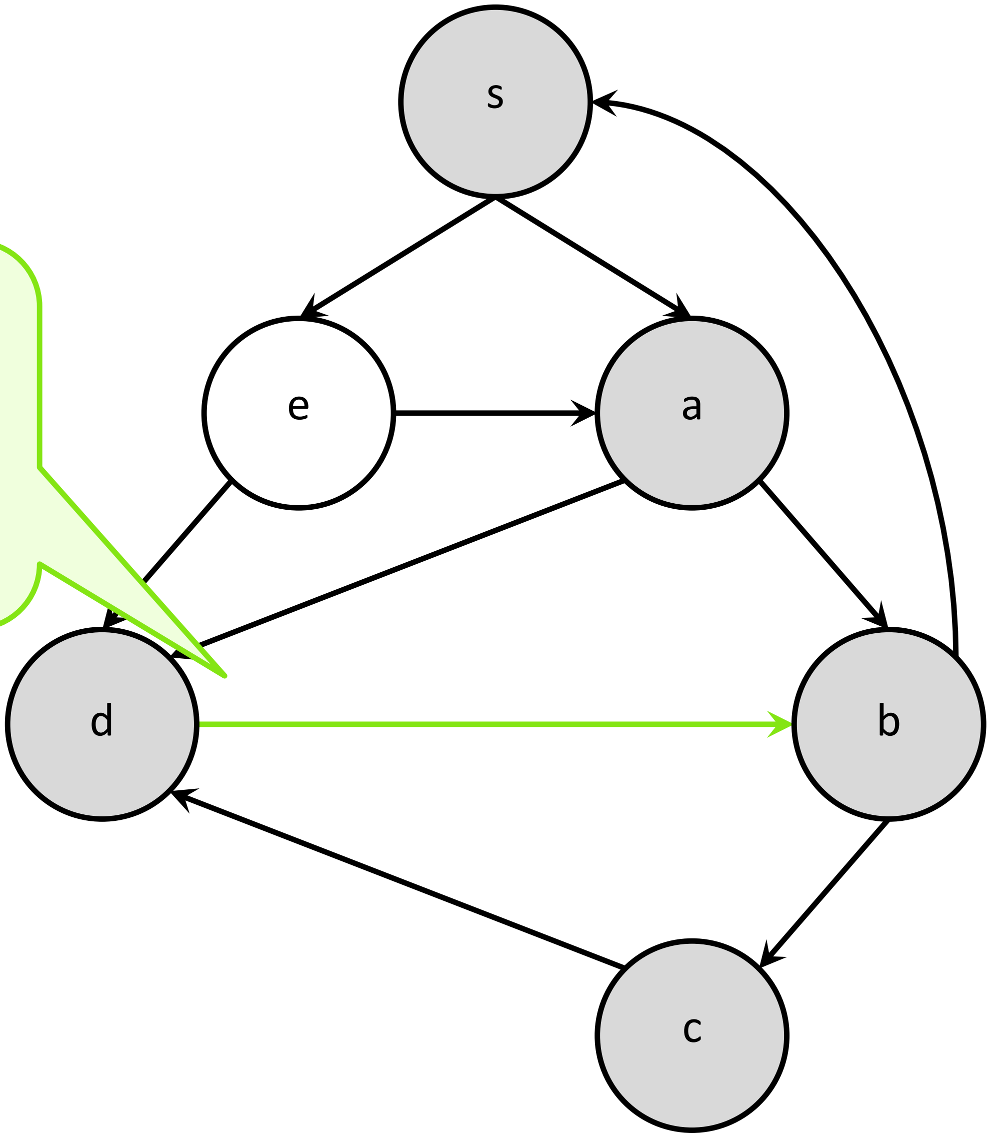
$b < c$


s
a
b
c
d
s
e
a
d
b
c
c < d

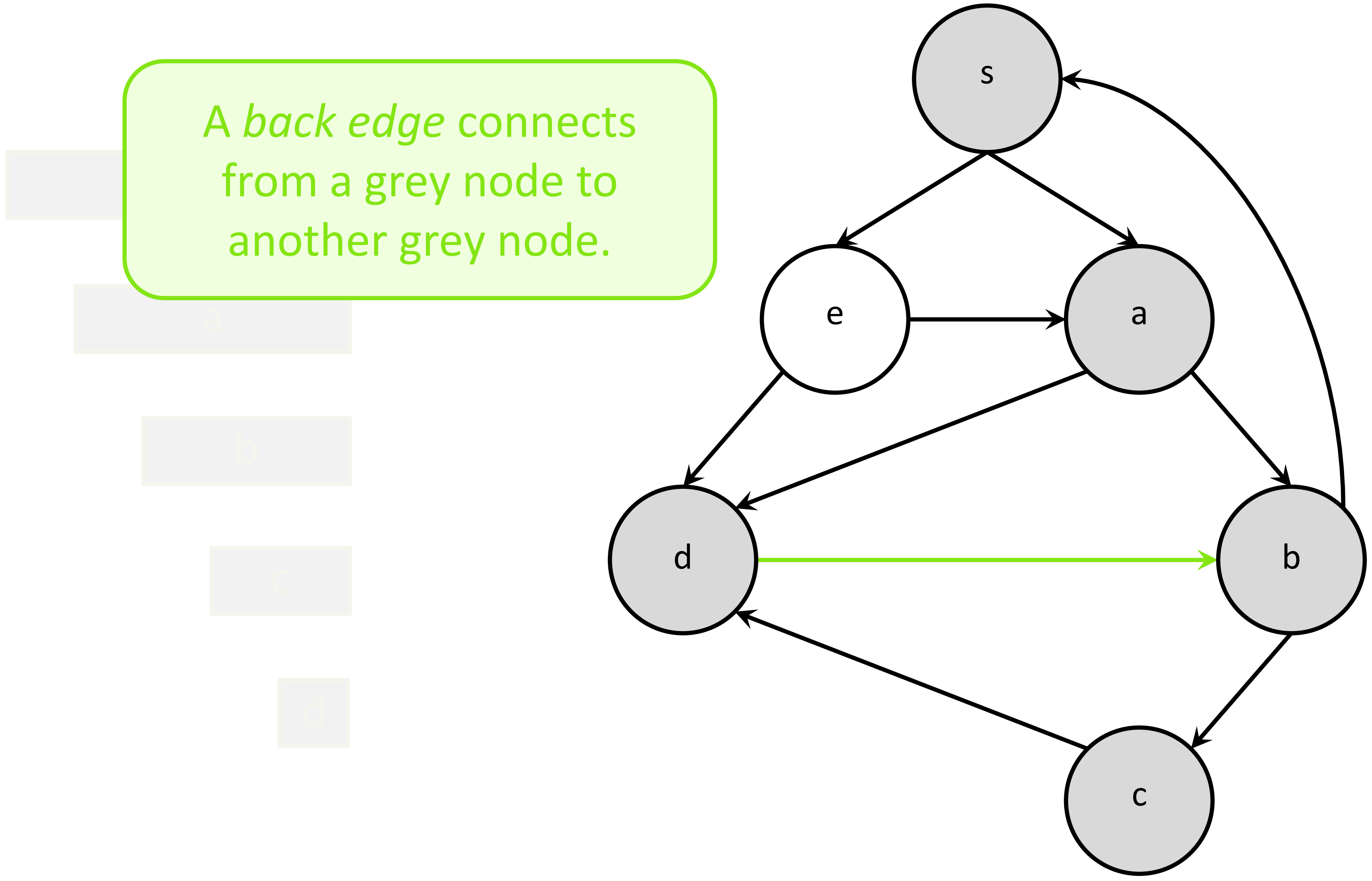
This is a *back edge*. We don't follow it because the grey node **b** is on the stack.

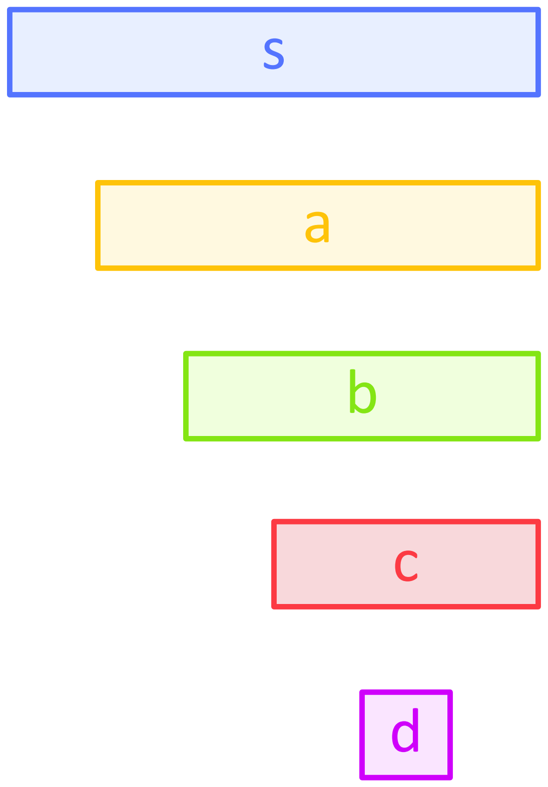
c

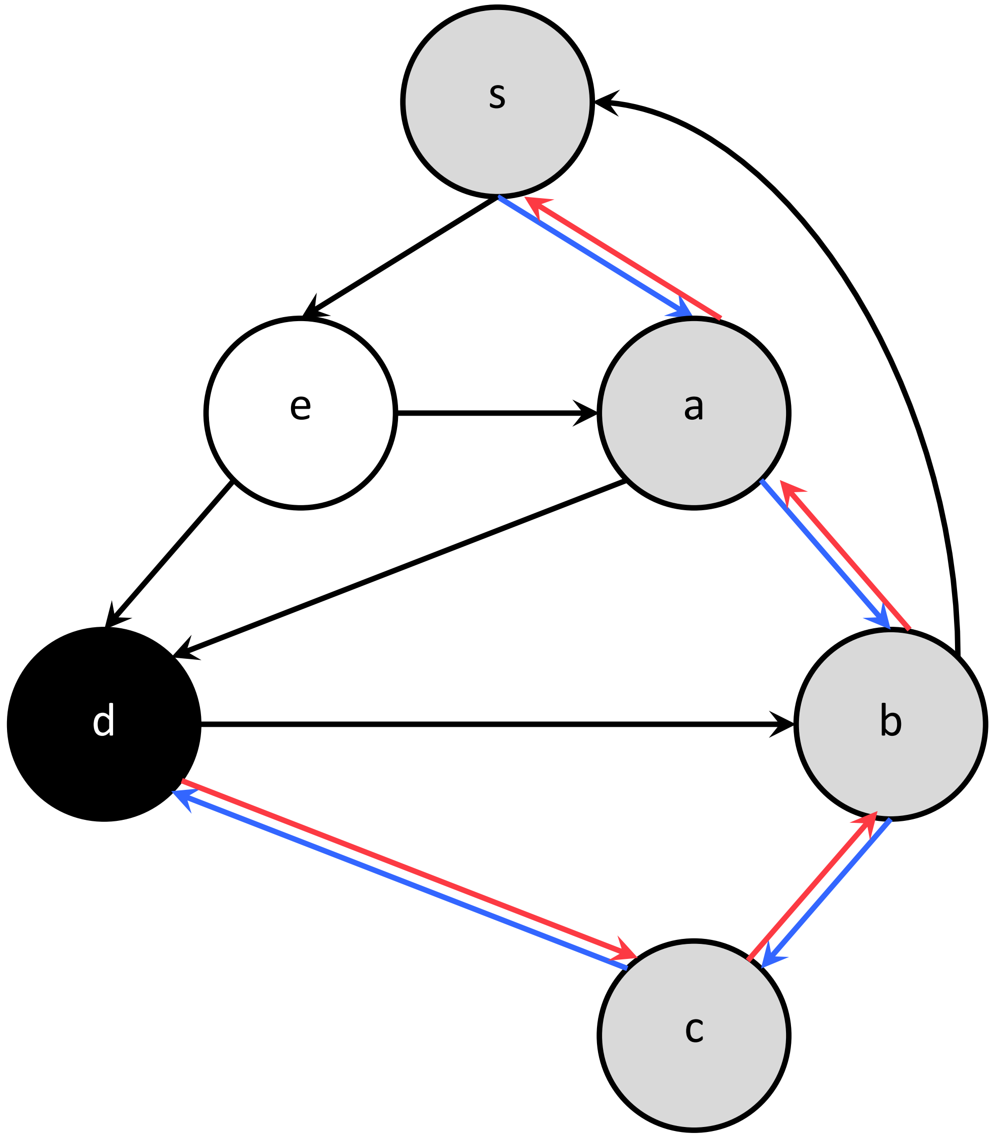
d

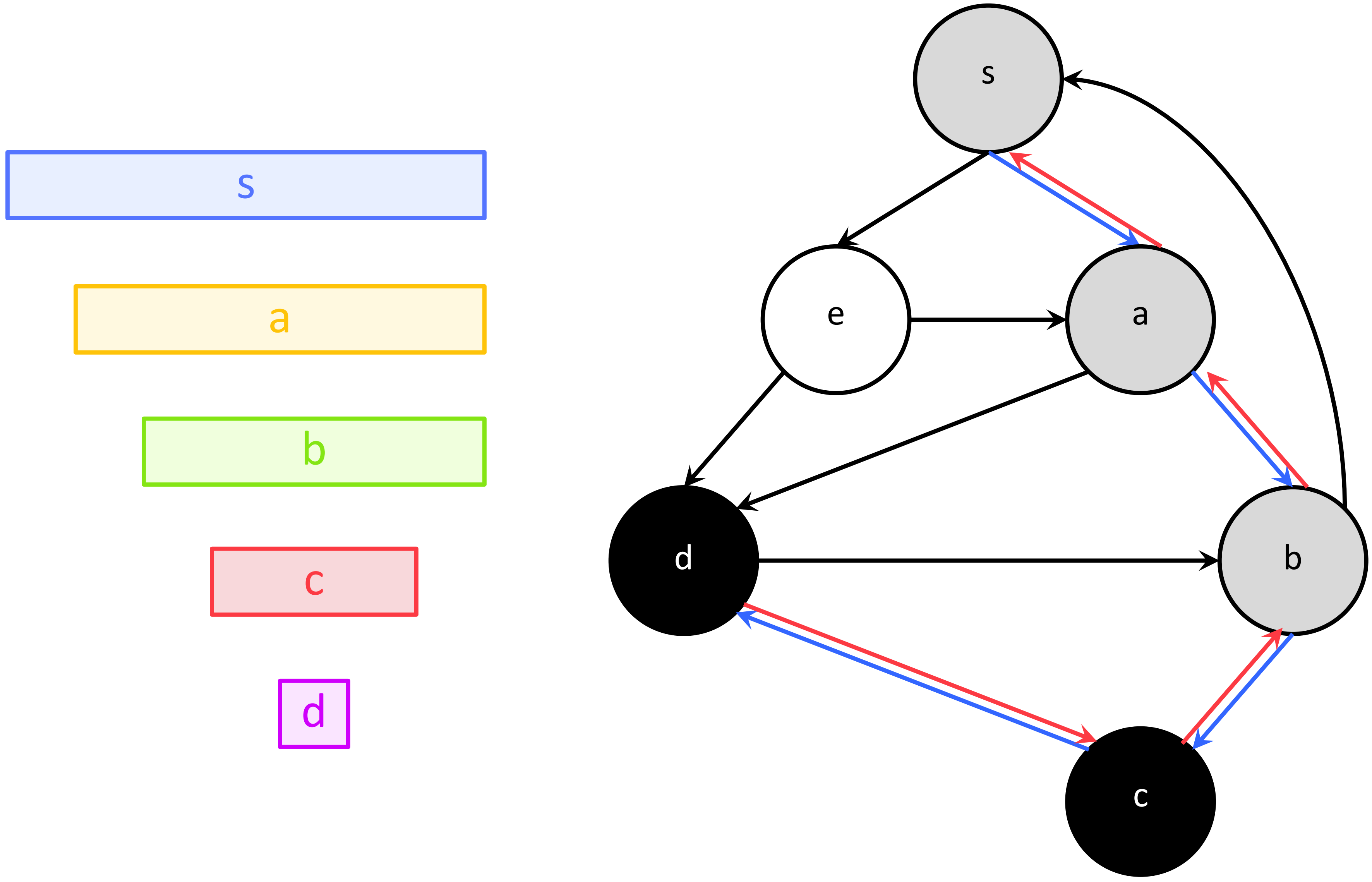


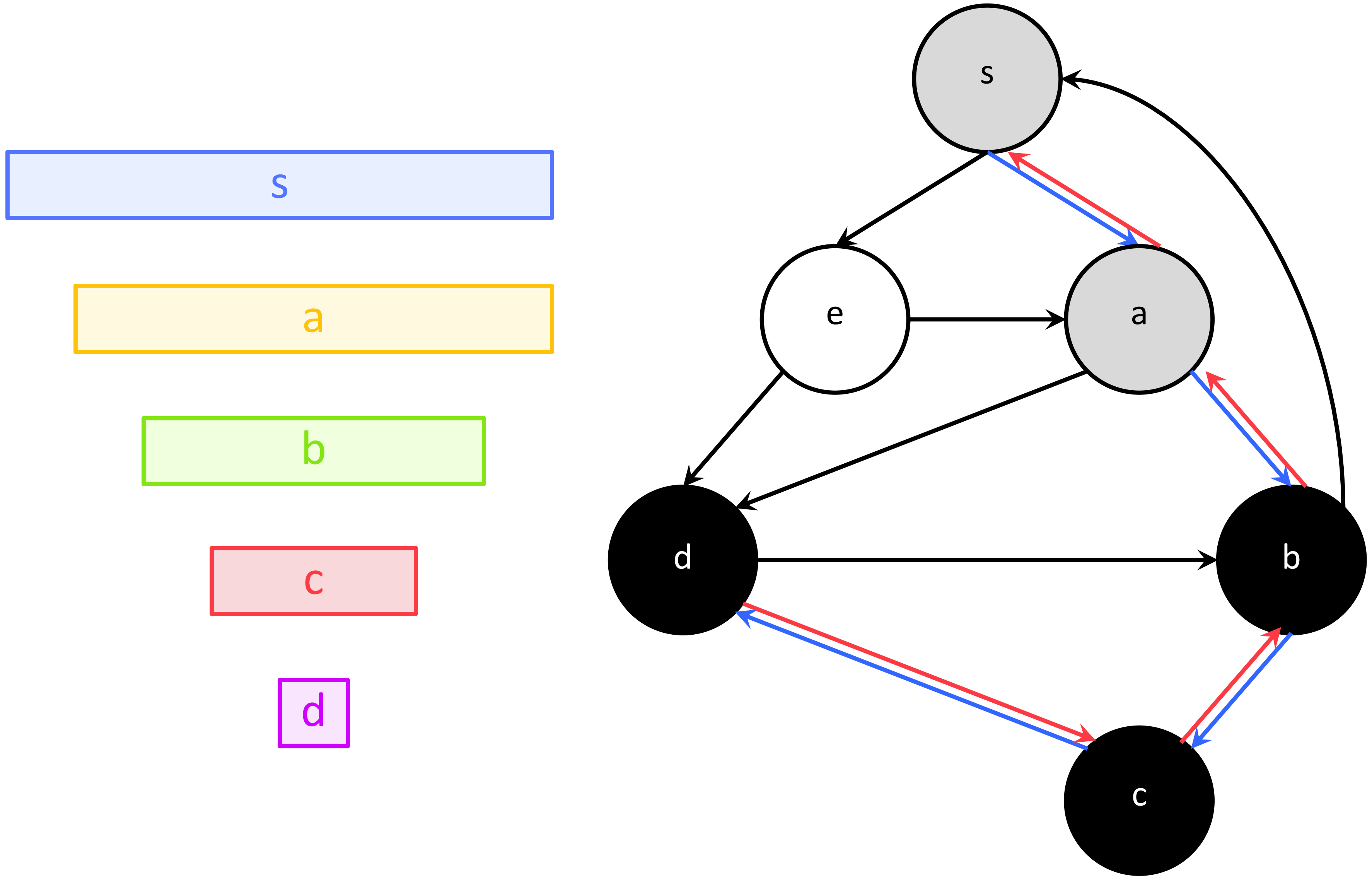
A *back edge* connects from a grey node to another grey node.



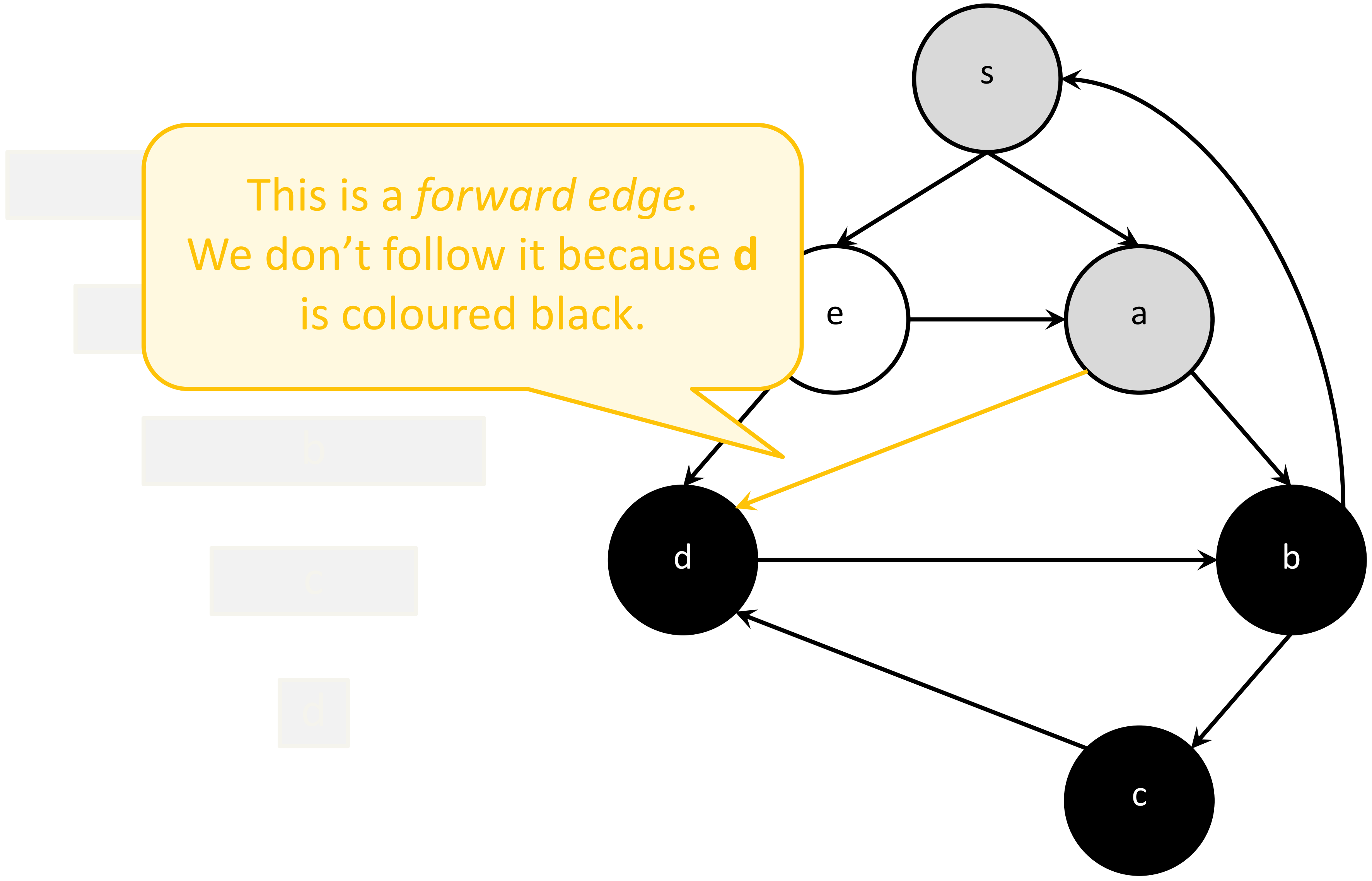

s
a
b
c
d


s
e
a
d
b
c

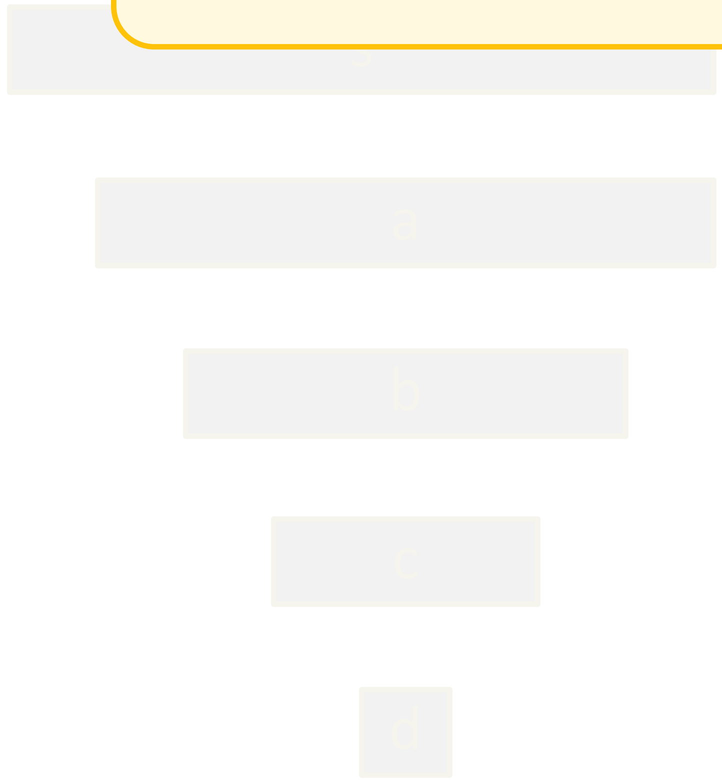

s
a
b
c
d
s
e
a
d
b
c

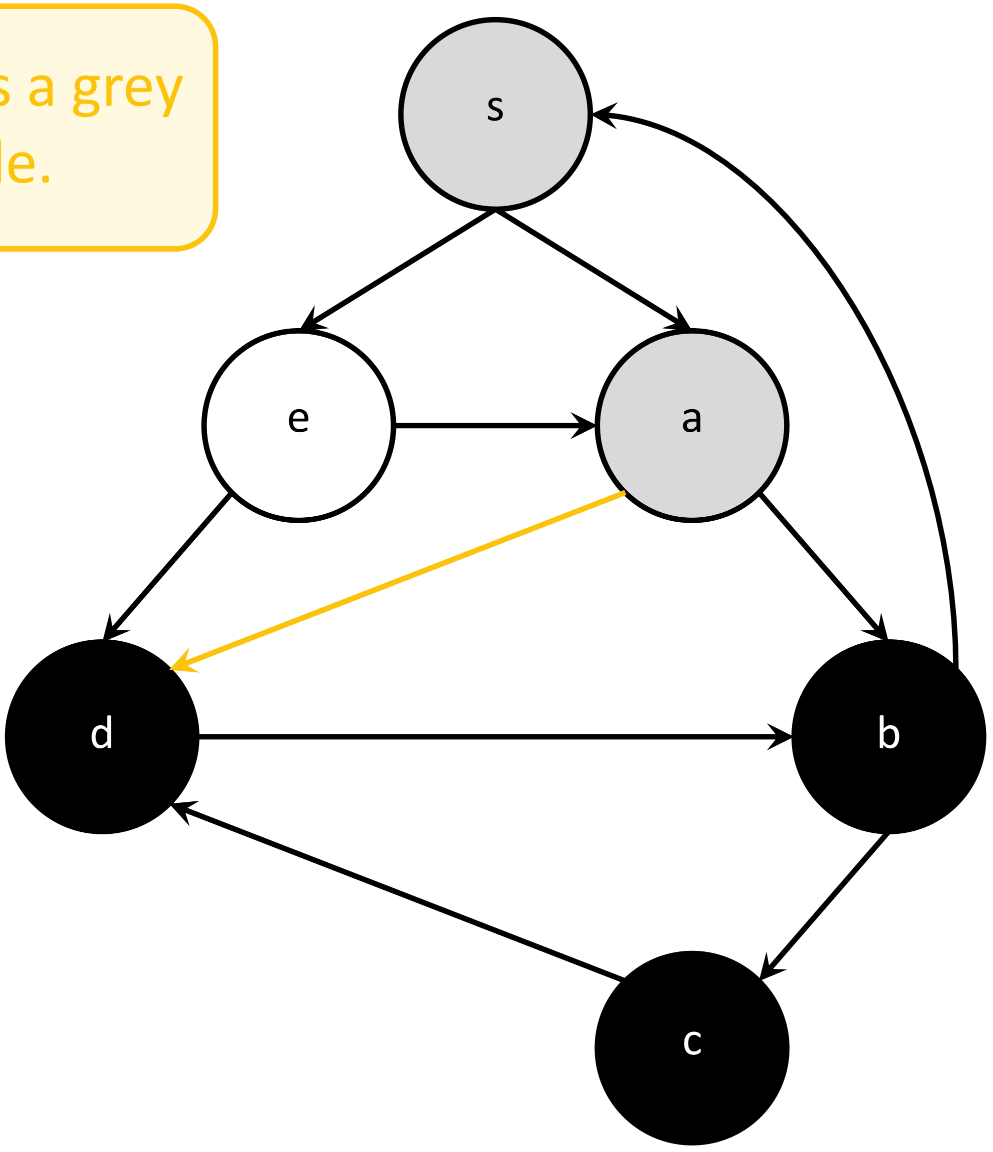

s
a
b
c
d
s
e
a
d
b
c

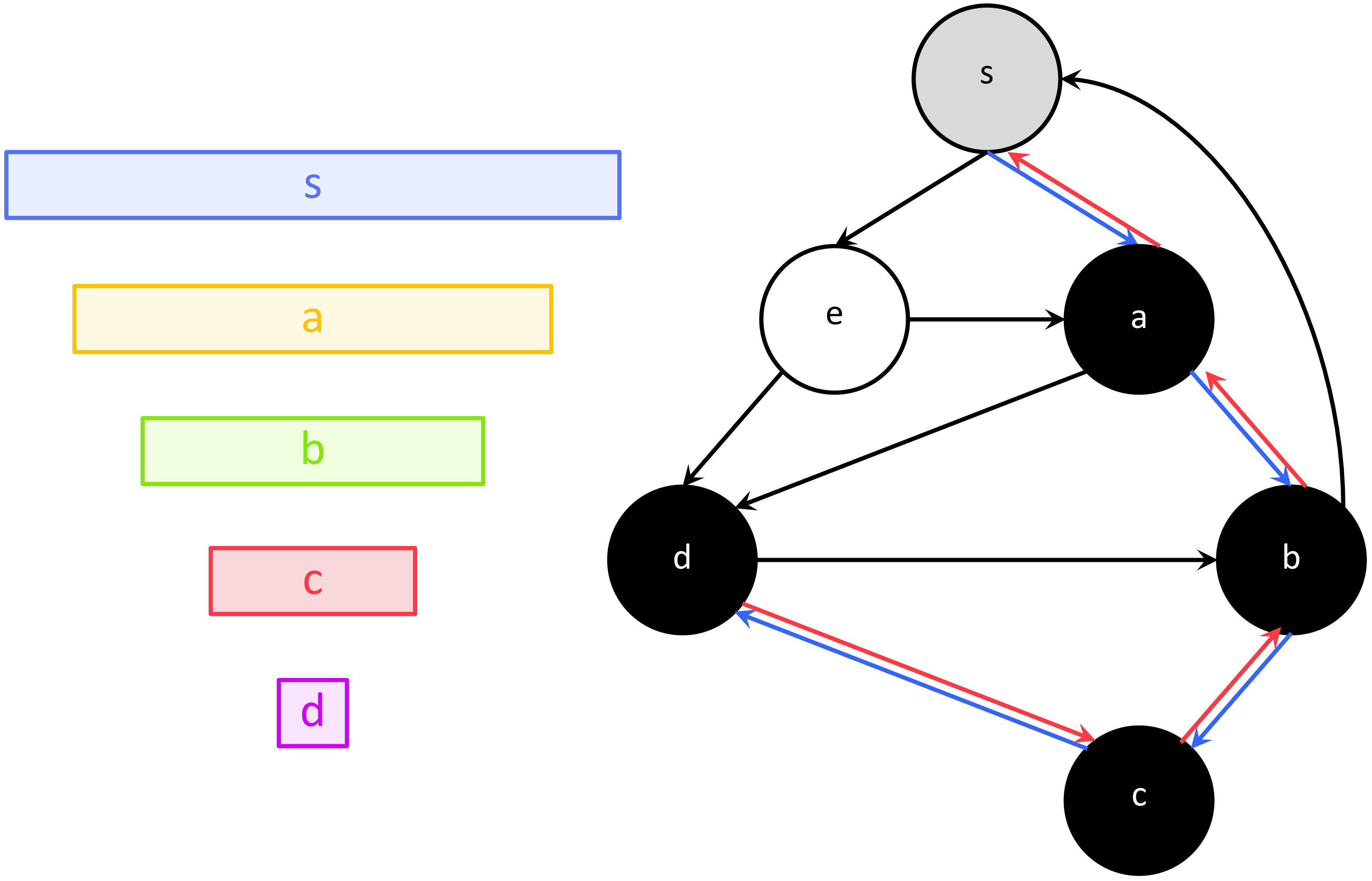
This is a *forward edge*.
We don’t follow it because **d** is coloured black.

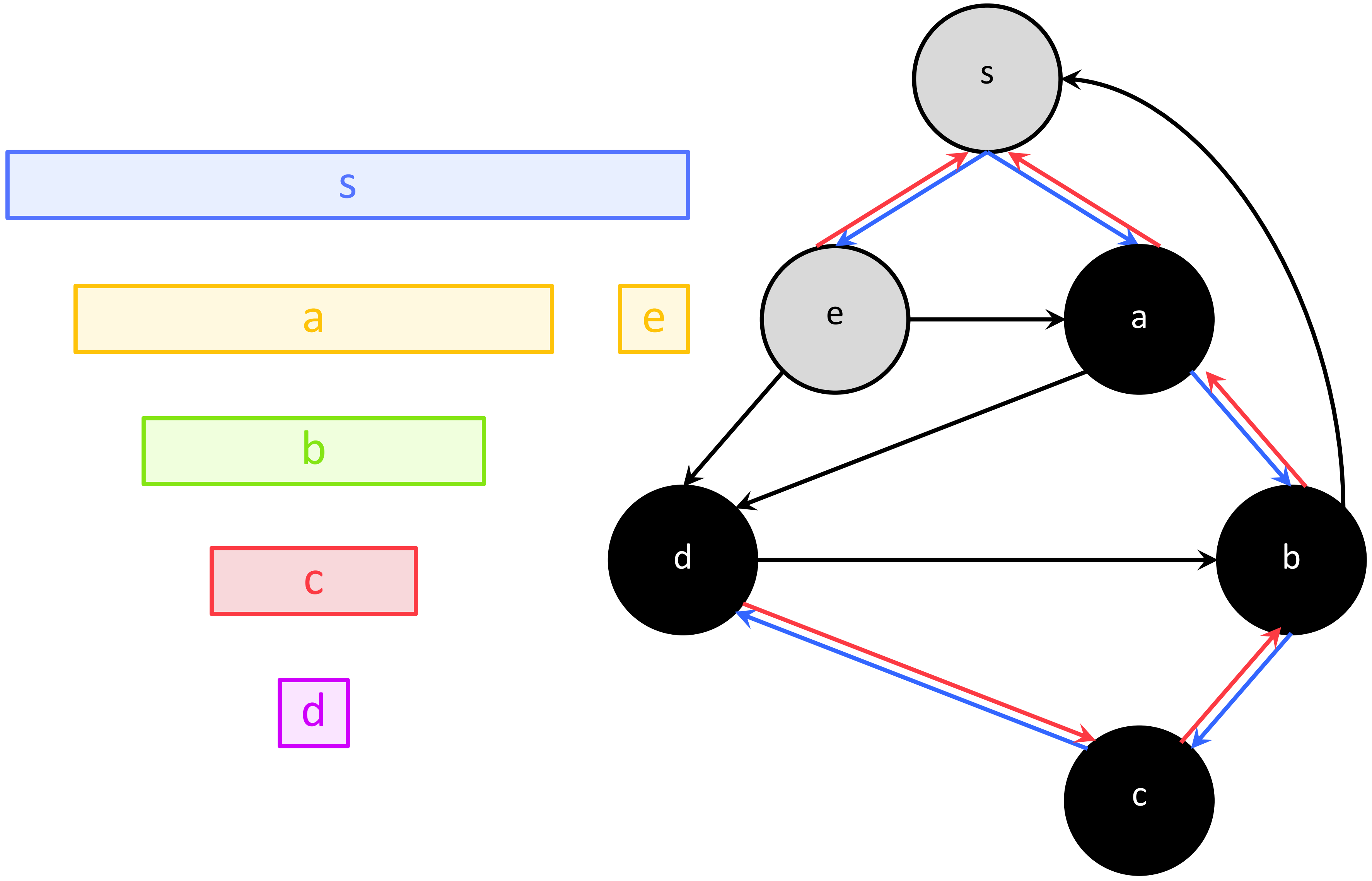


A *forward edge* connects a grey node to a black node.

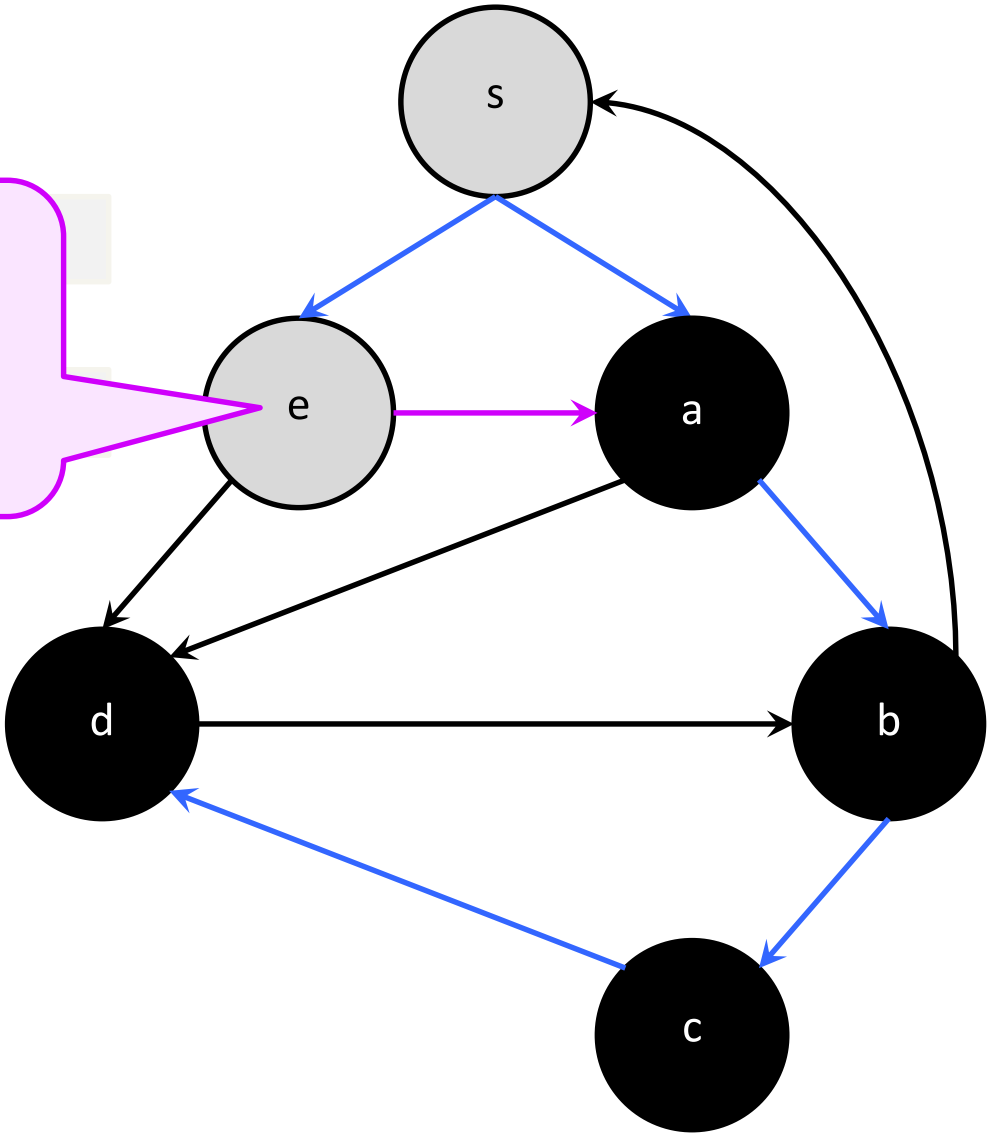




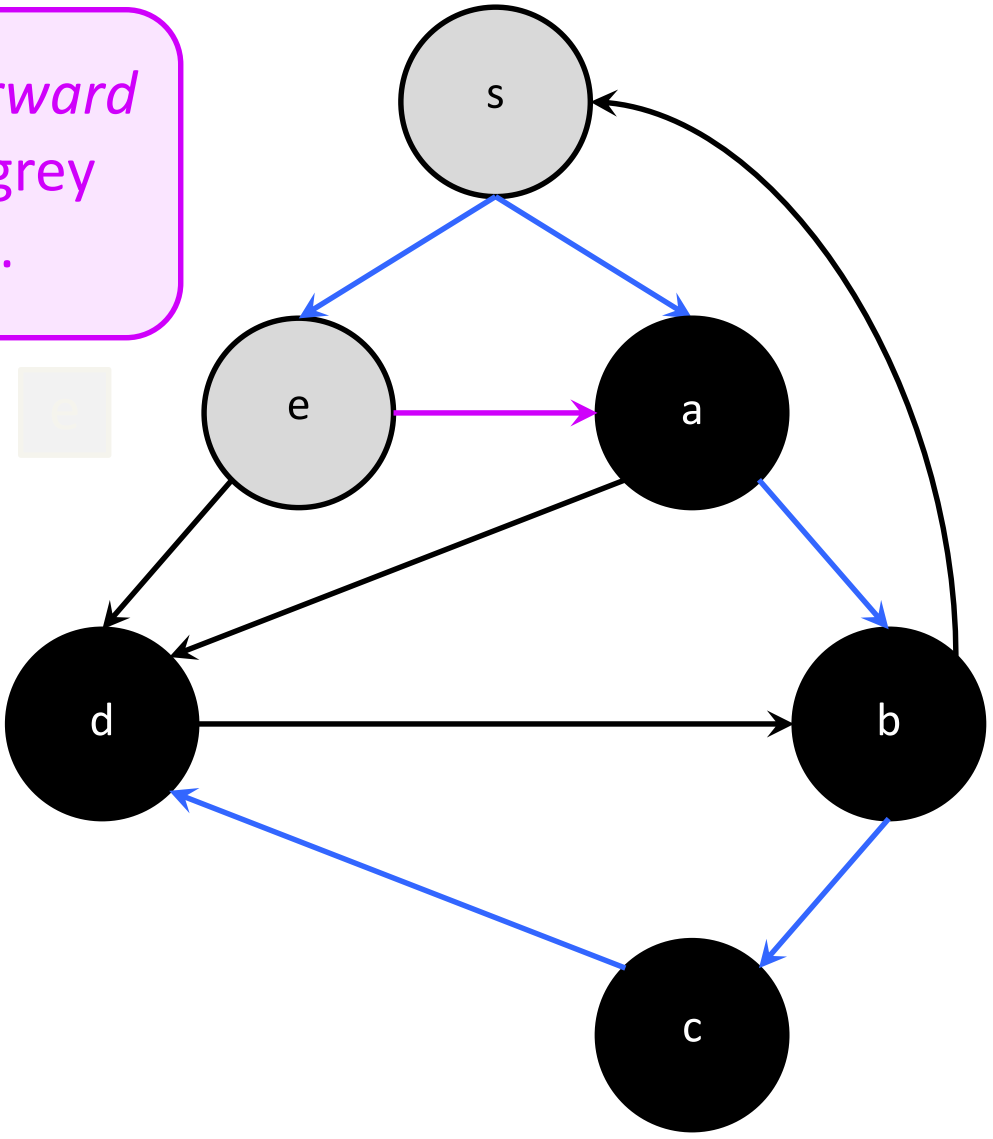

s
a
b
c
d
s
e
a
d
b
c

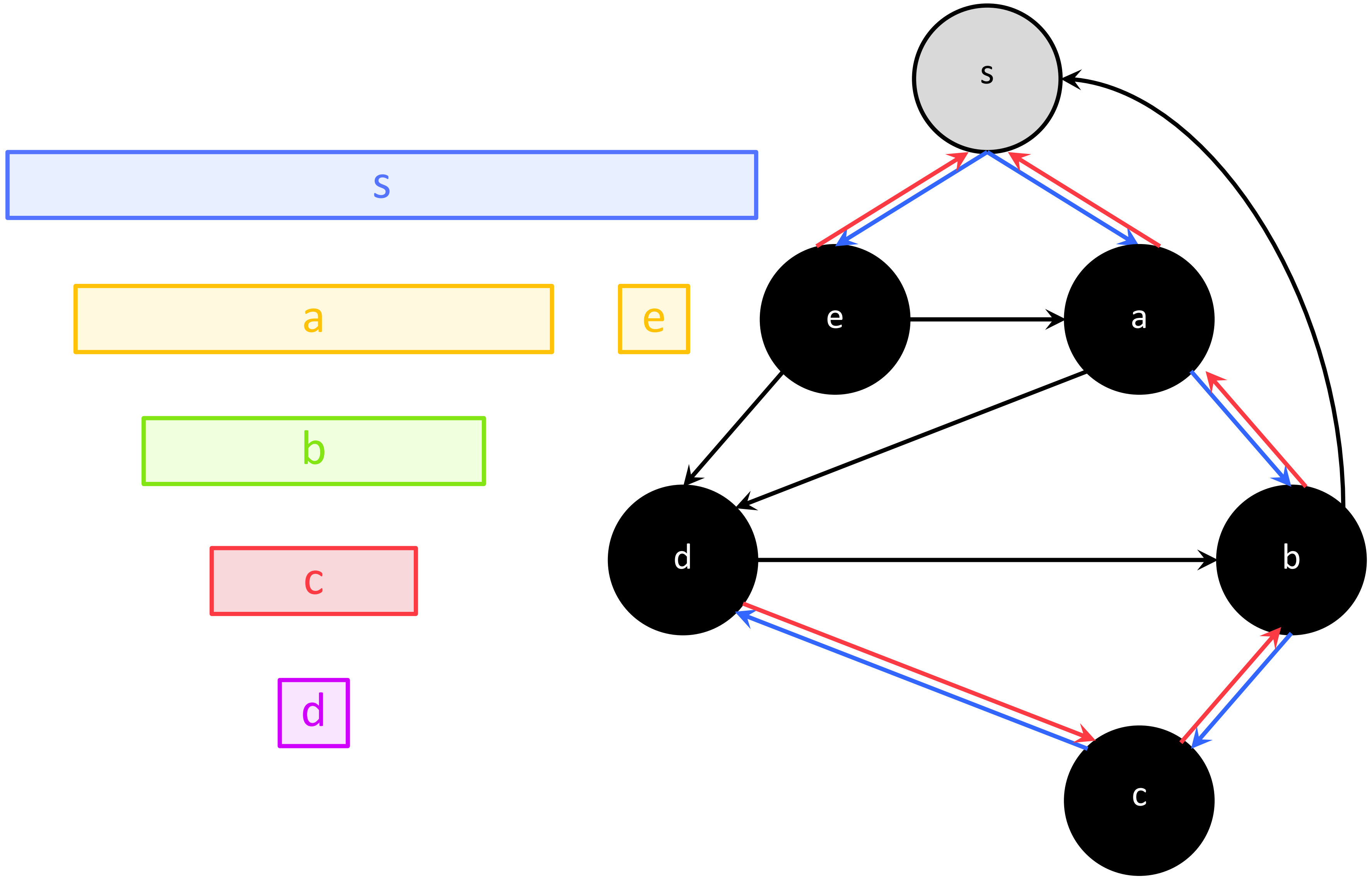

s
a
e
b
c
d
s
e
a
d
b
c

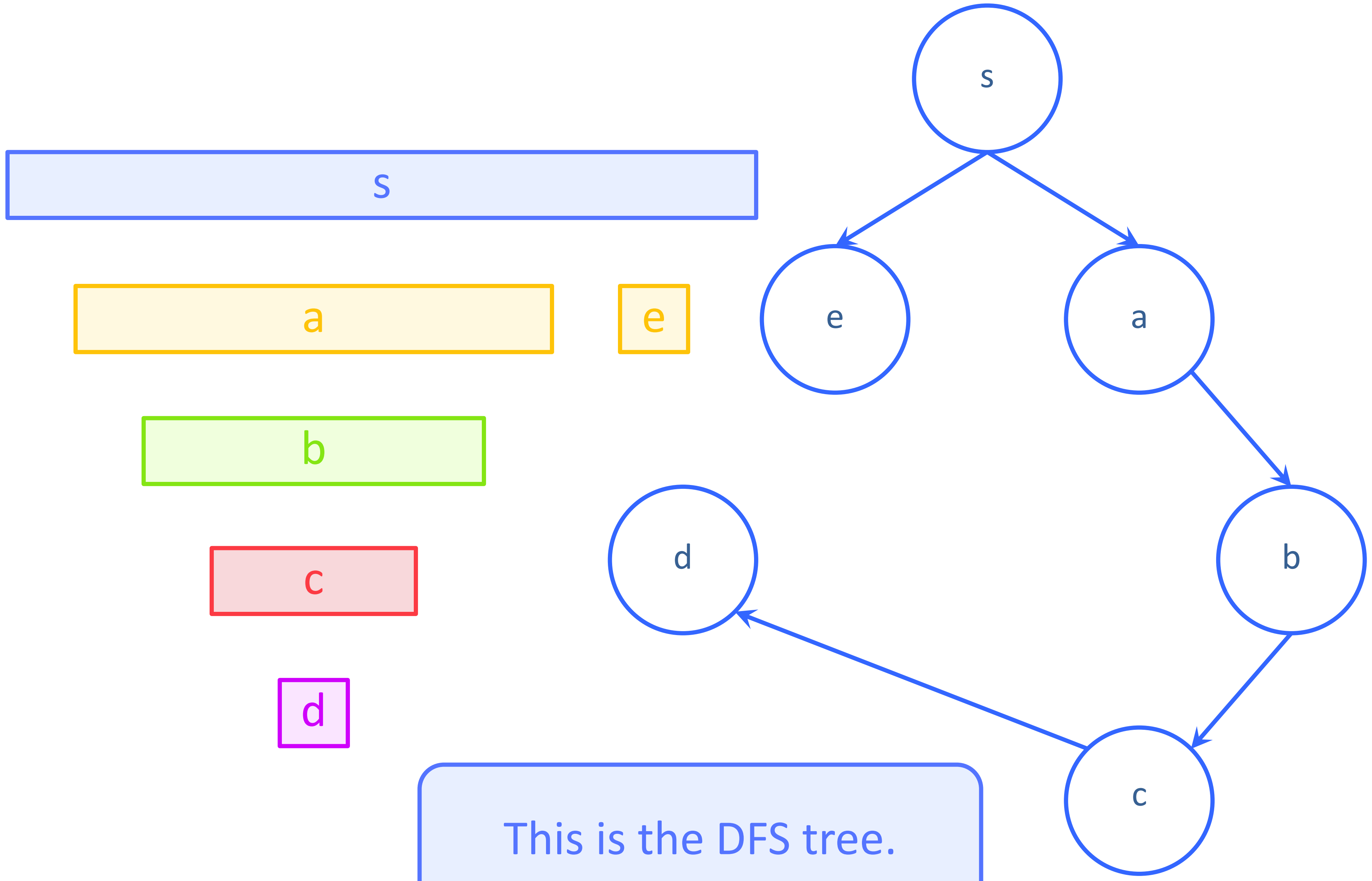
This is a *cross edge*.
It connects between two
different subtrees.



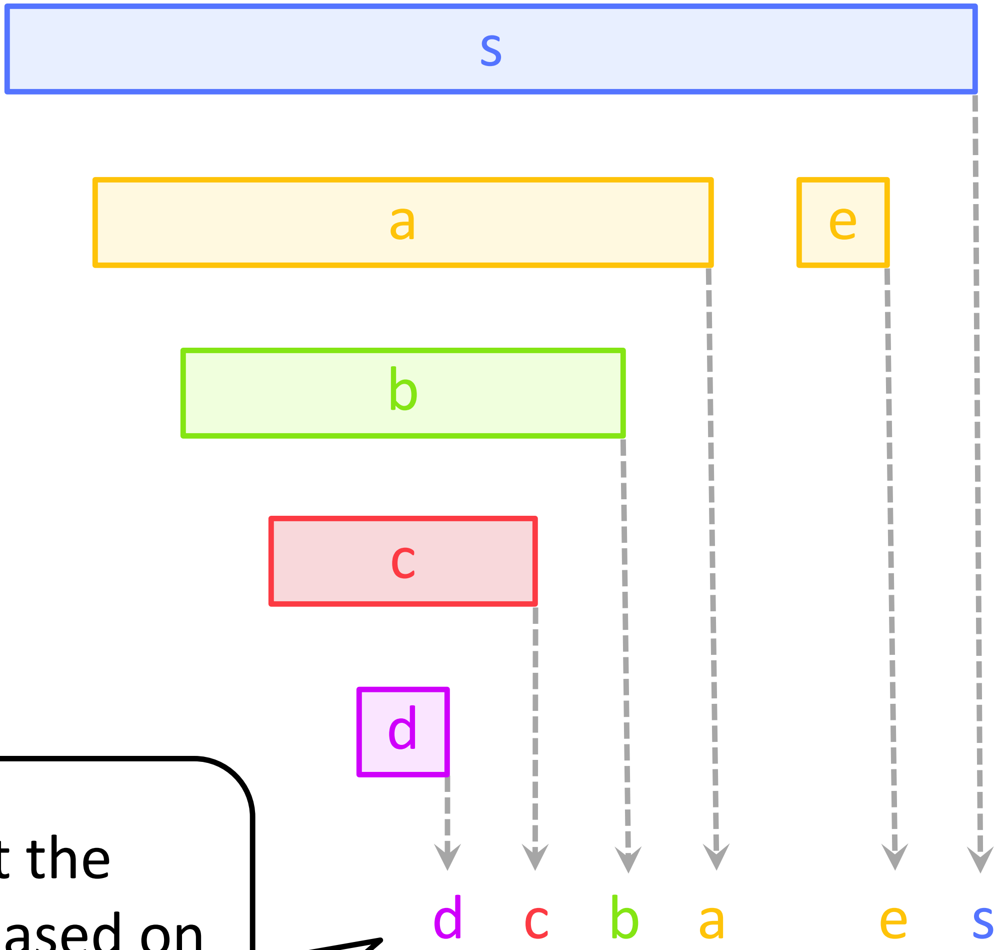
Both *cross edges* and *forward edges* connect from a grey node to a black one.




s
a
e
b
c
d
s
e
a
d
b
c



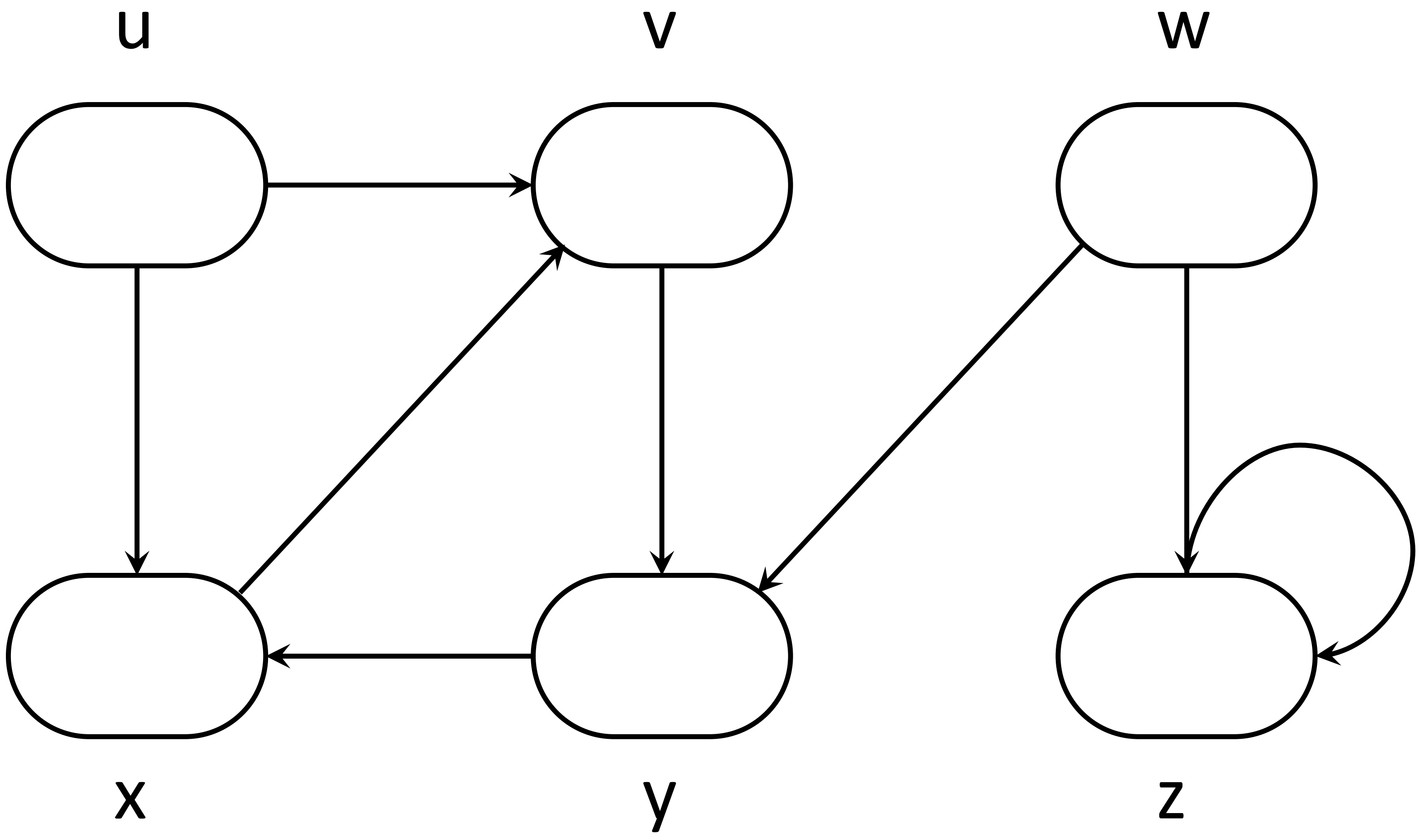
This is the DFS tree.



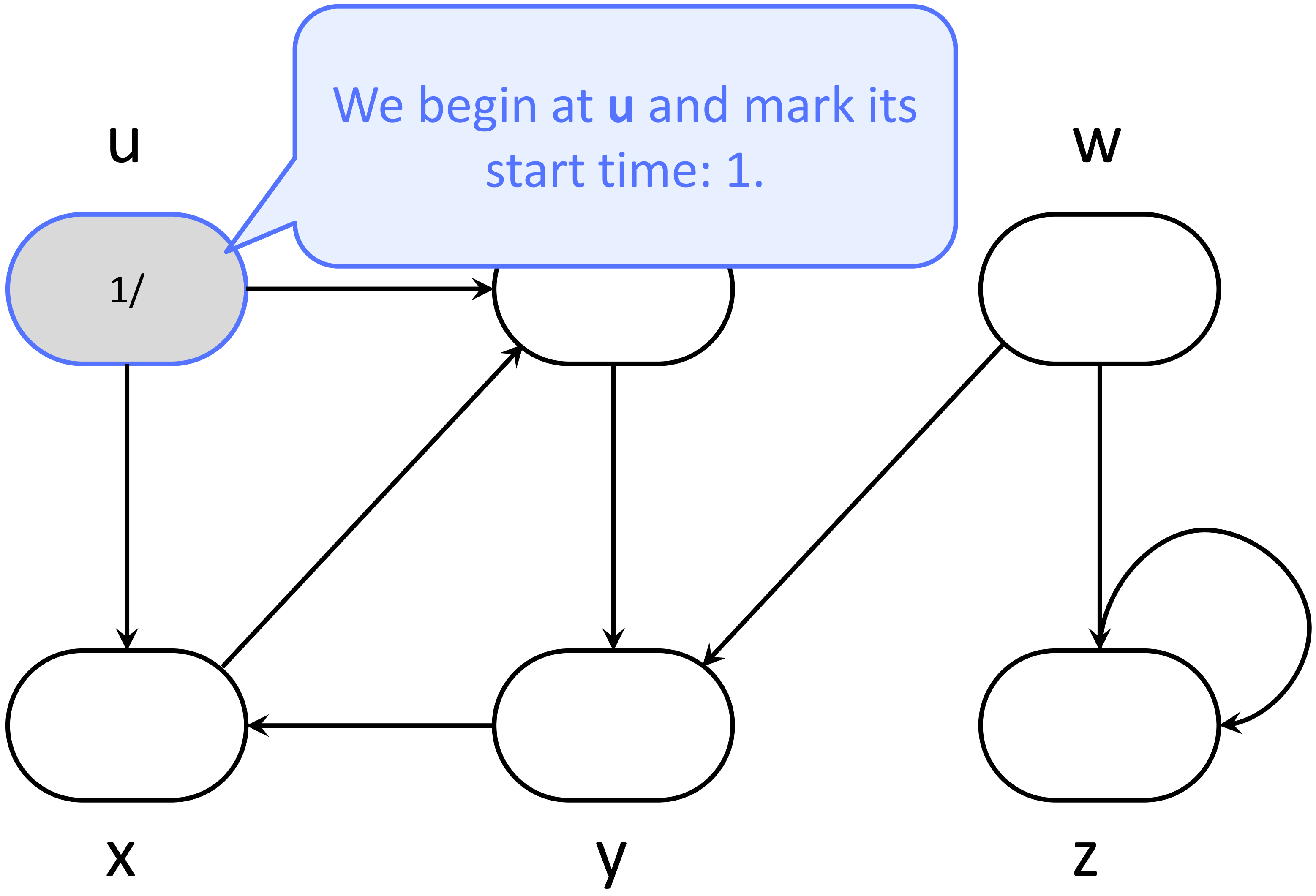
We sort the elements based on their finish times.

DFS
example

- Start / end times
- Classification of edges

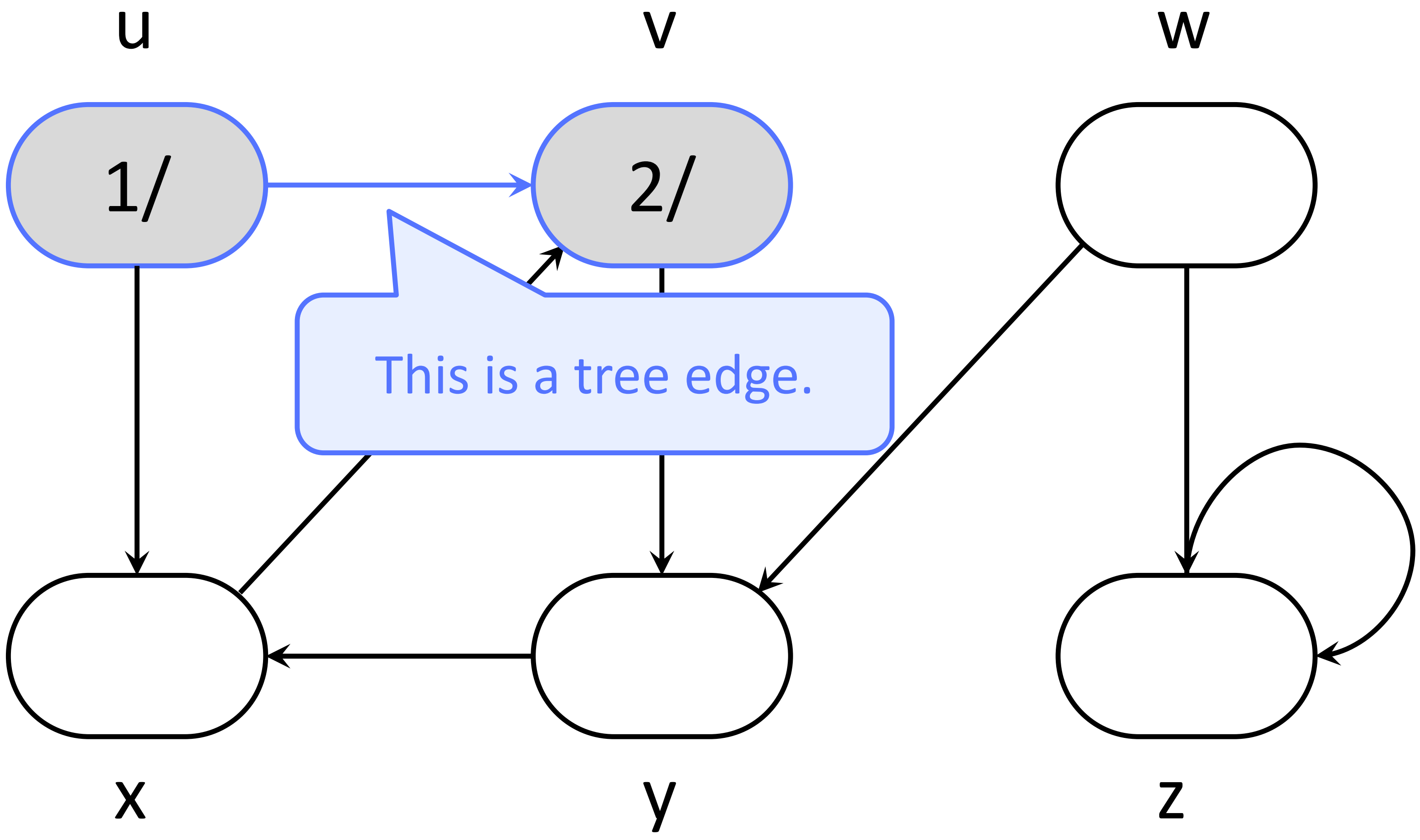


Time: 0

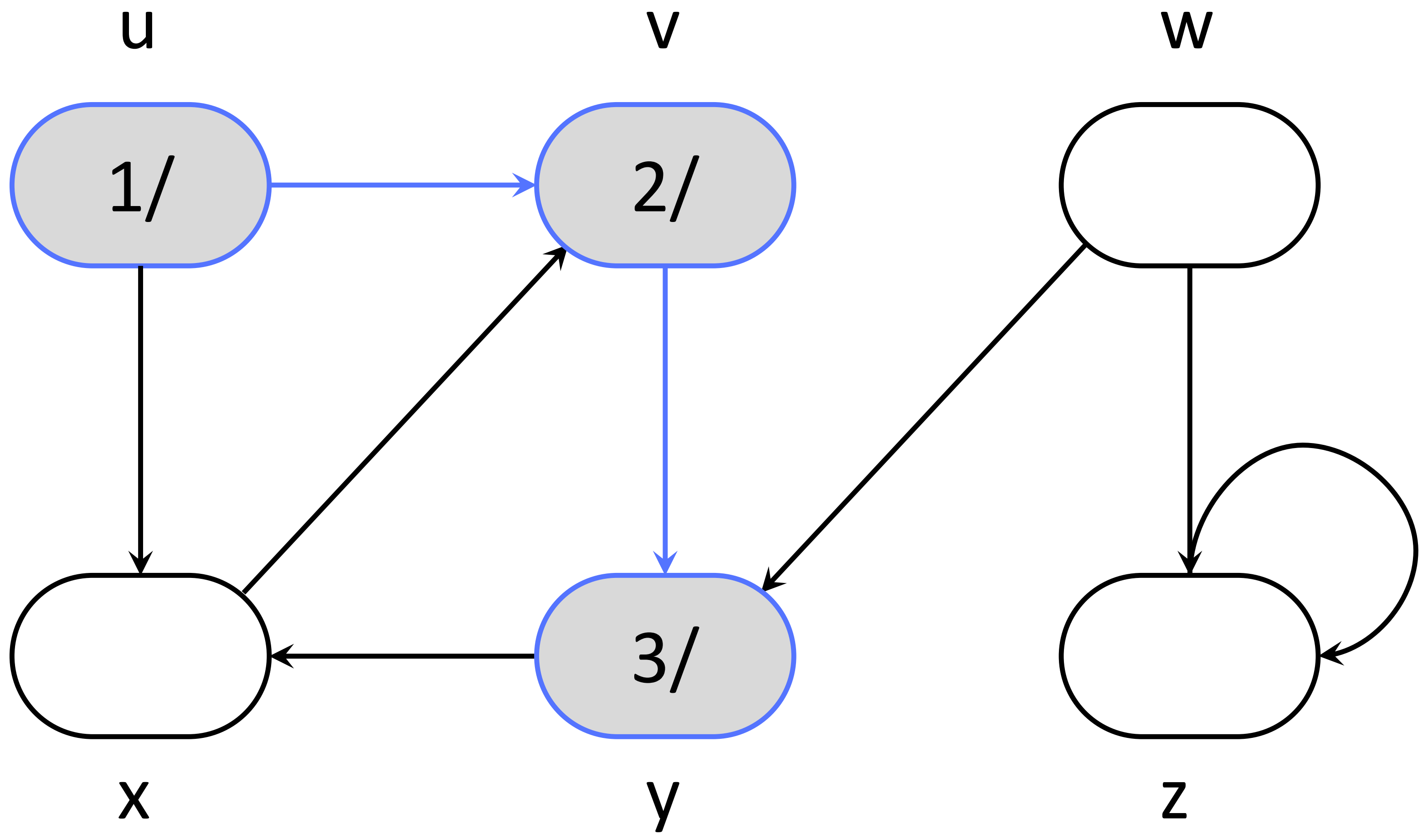


We begin at **u** and mark its start time: 1.

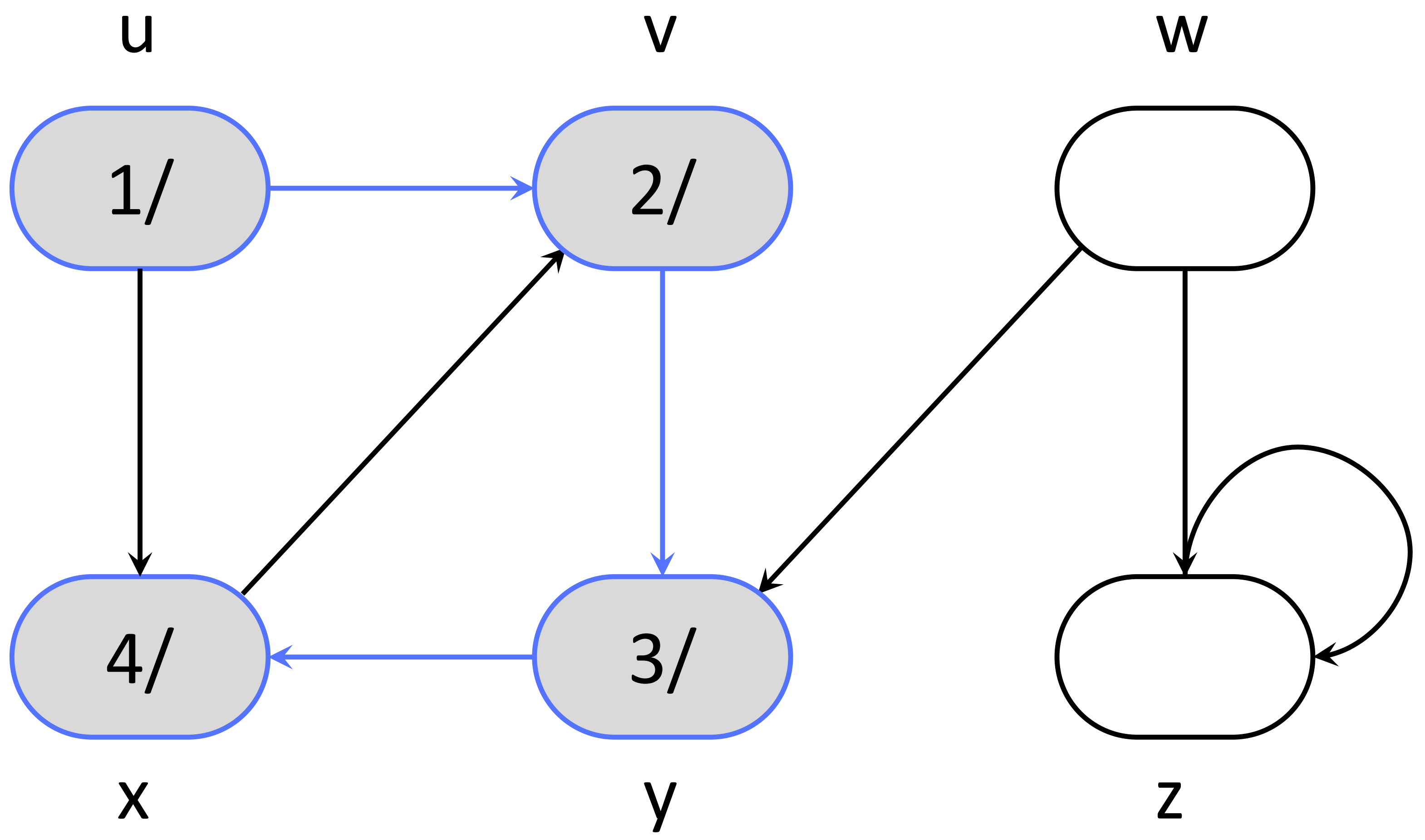
Time: 1



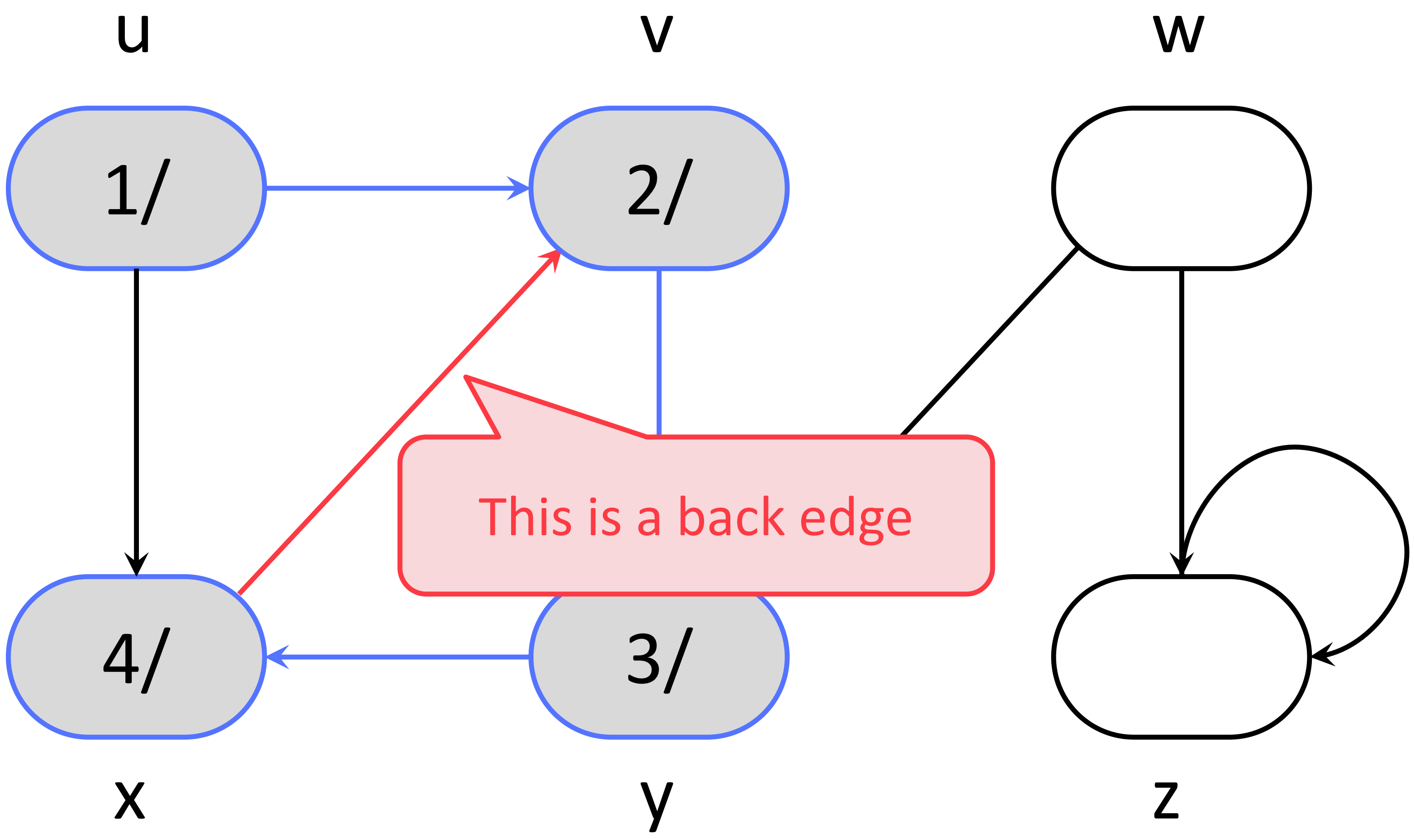
Time: 2



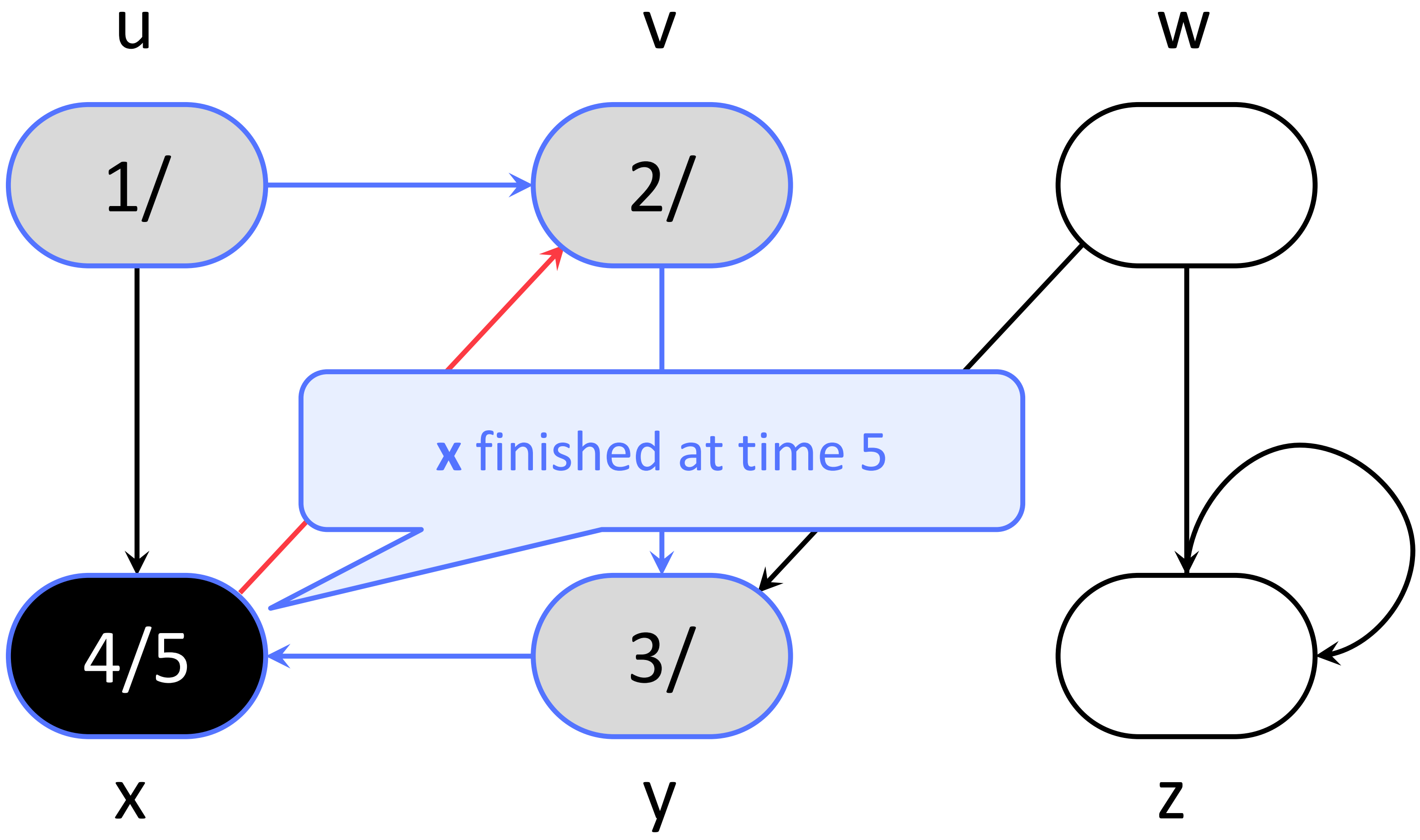
Time: 3



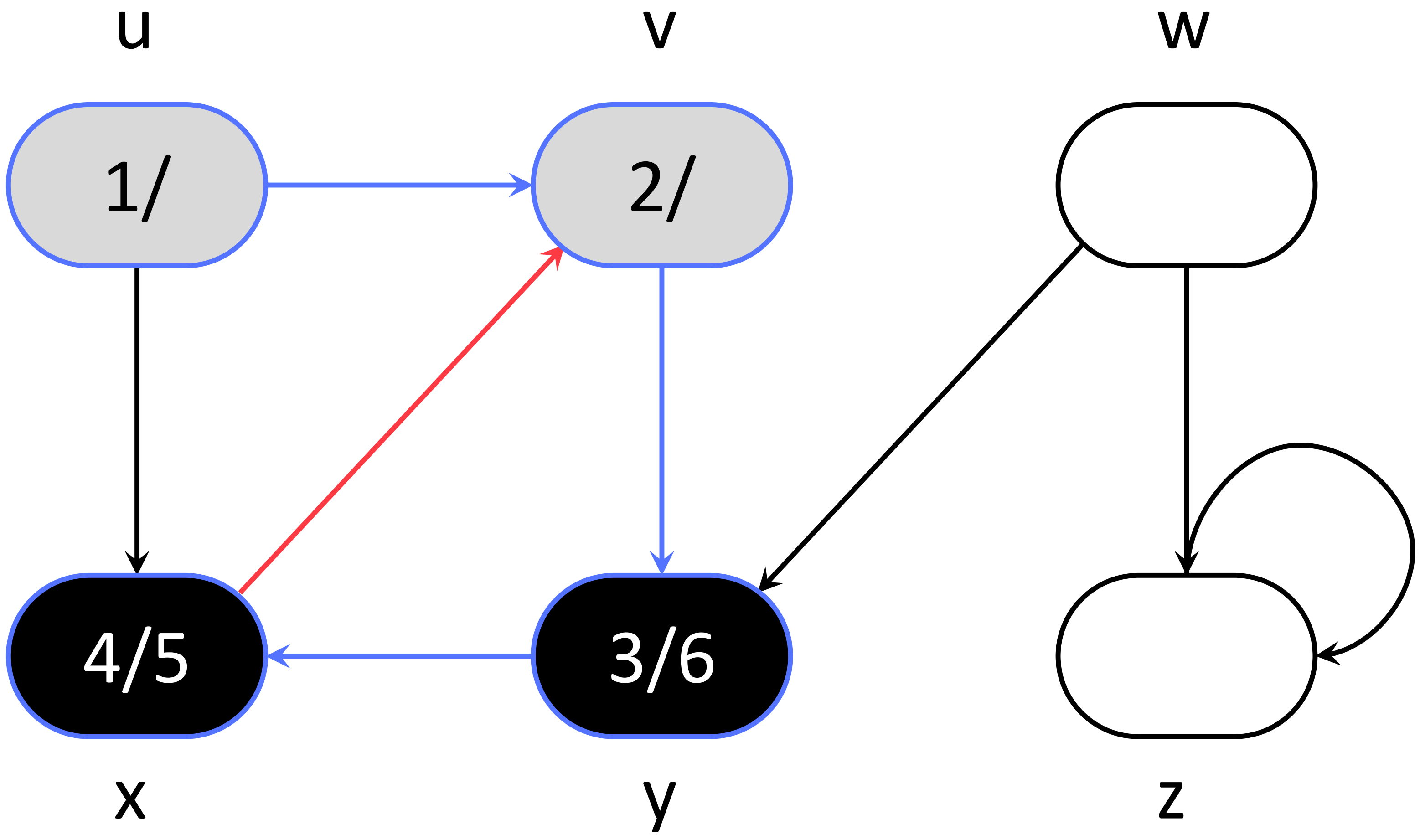
Time: 4



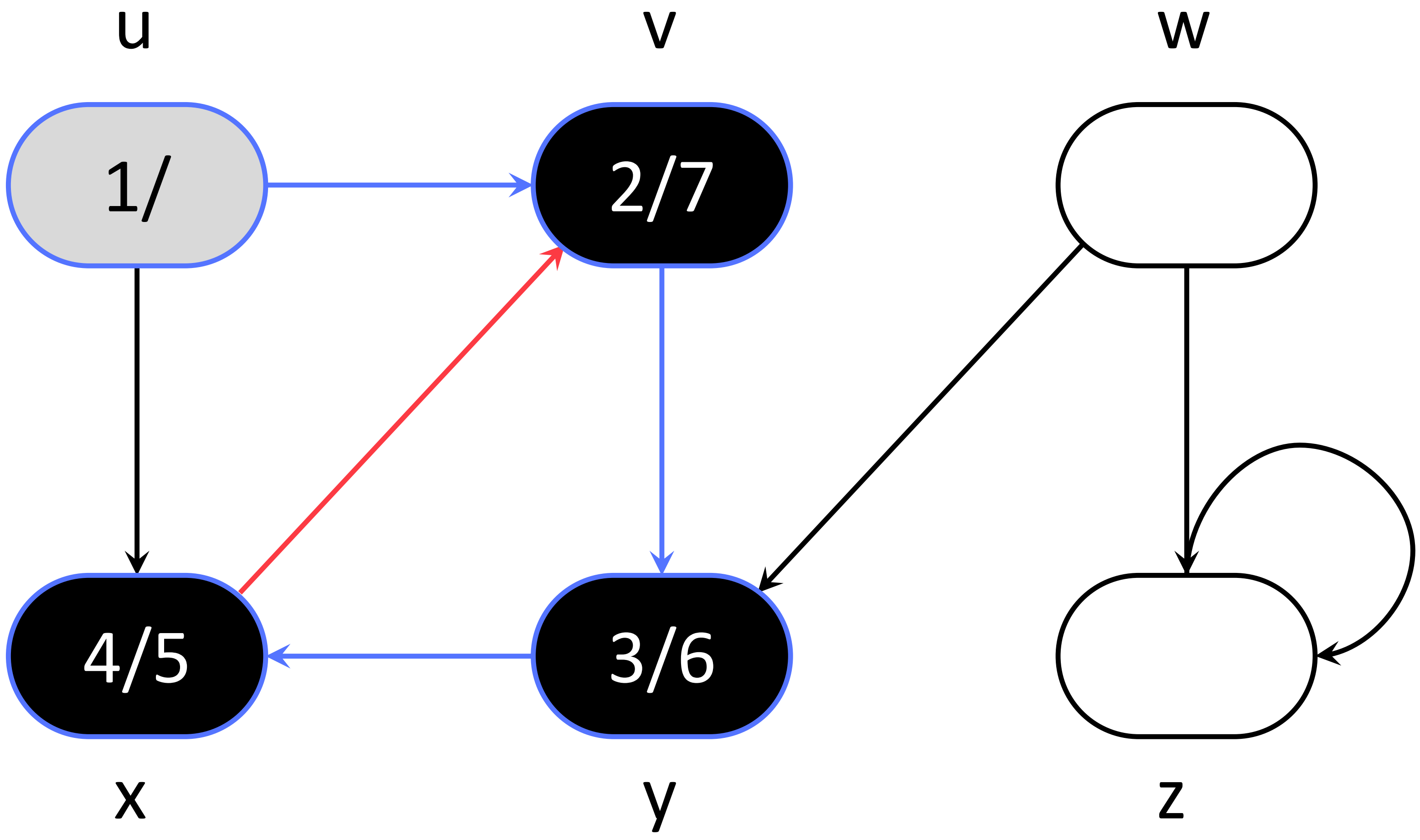
Time: 5



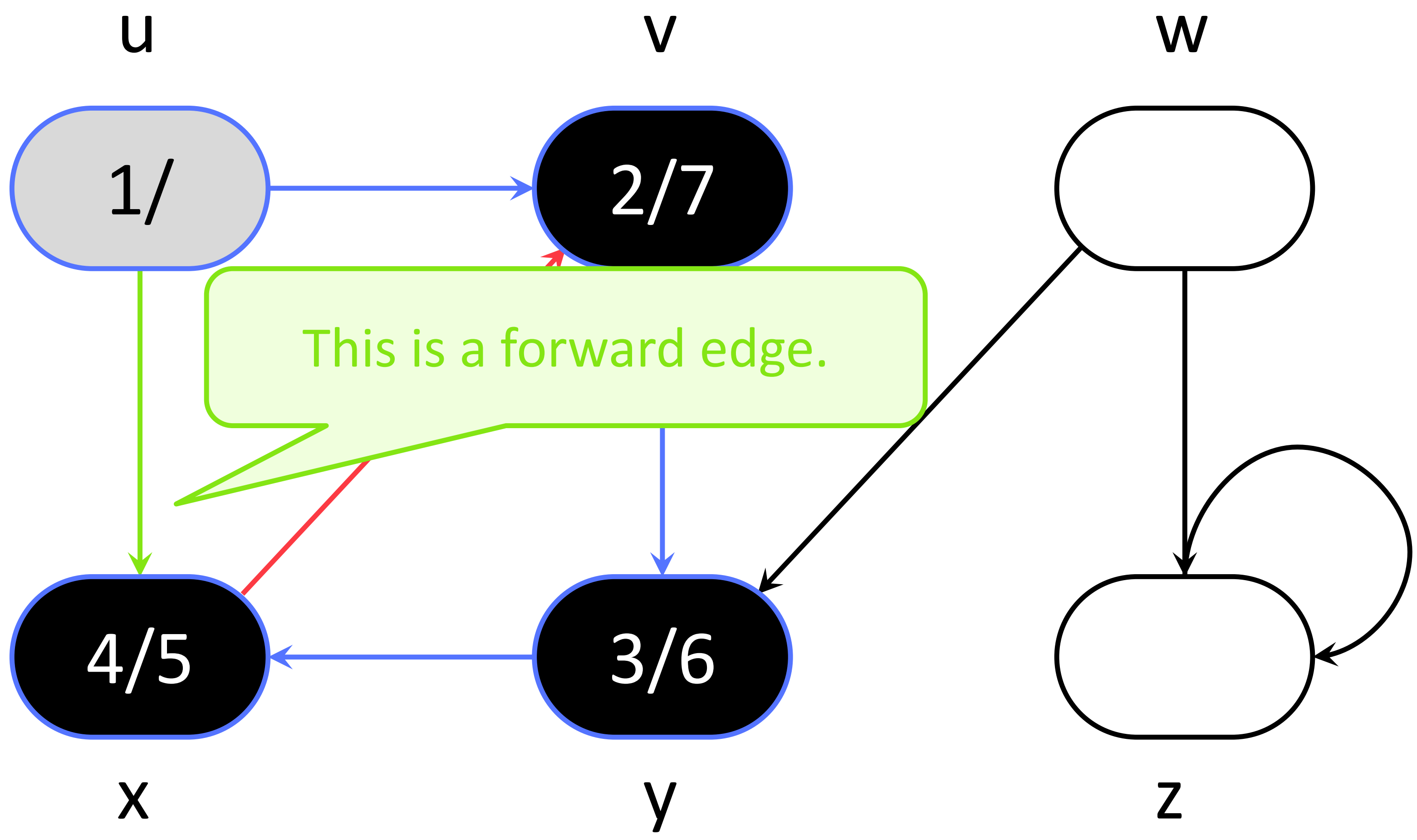
Time: 5



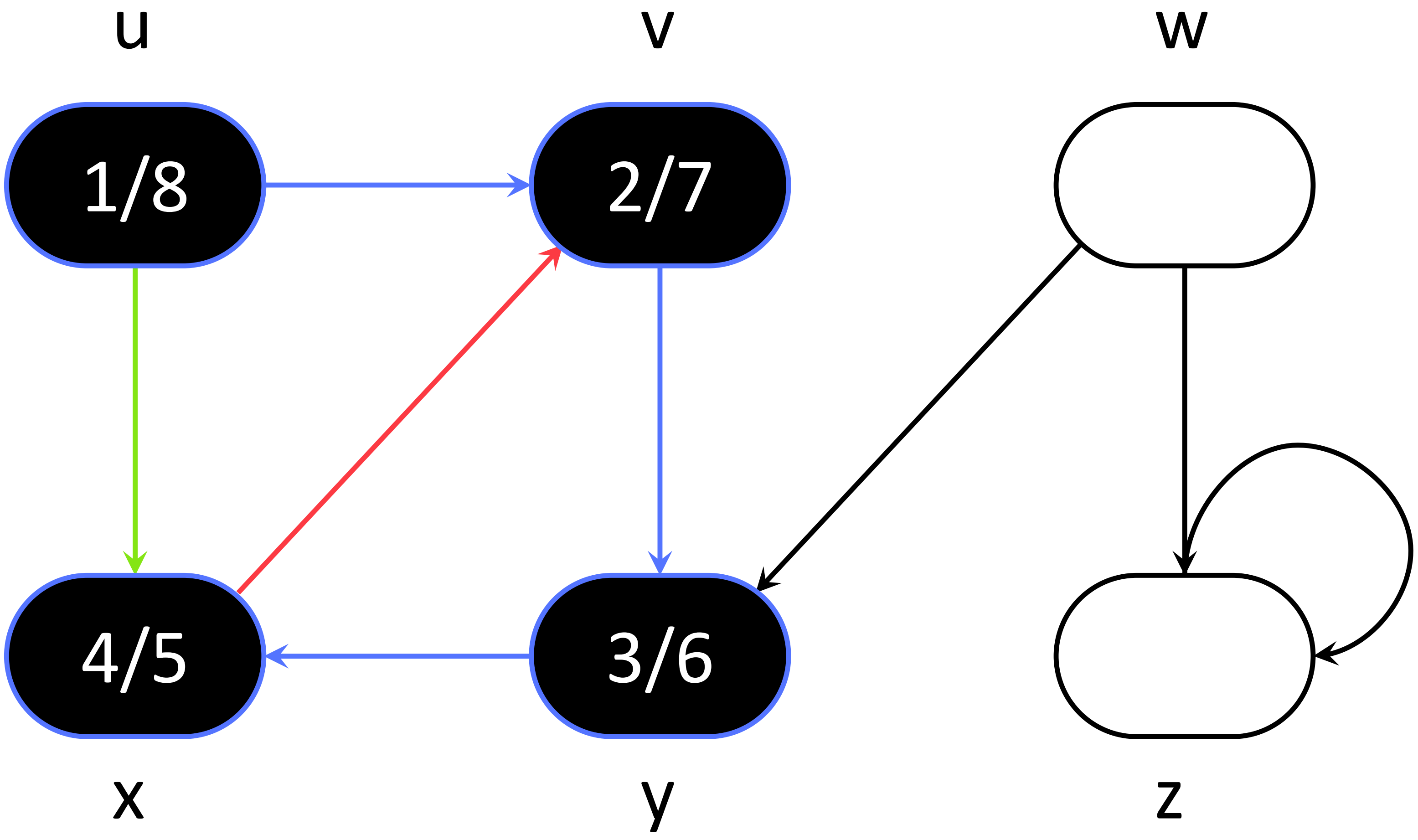
Time: 6



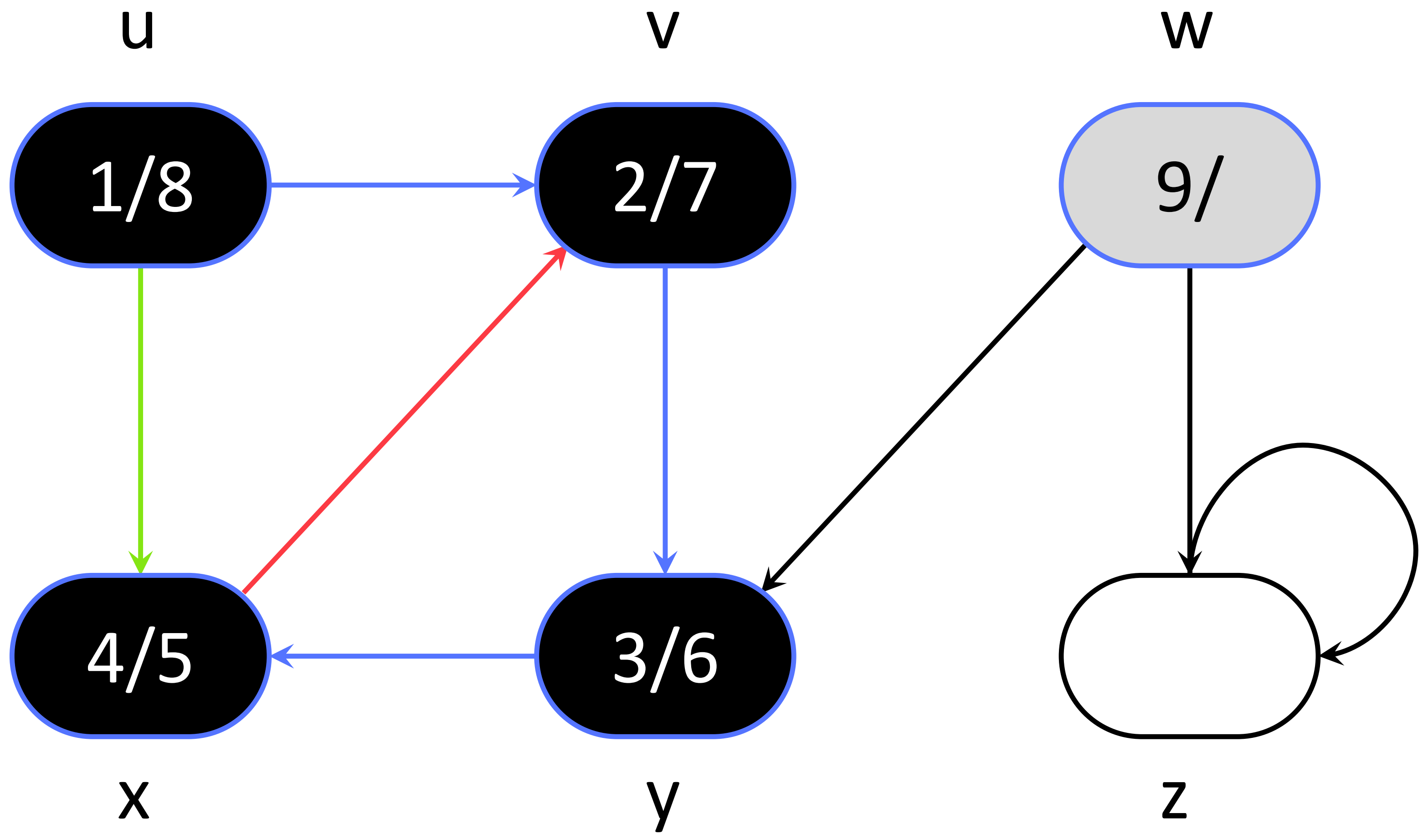
Time: 7



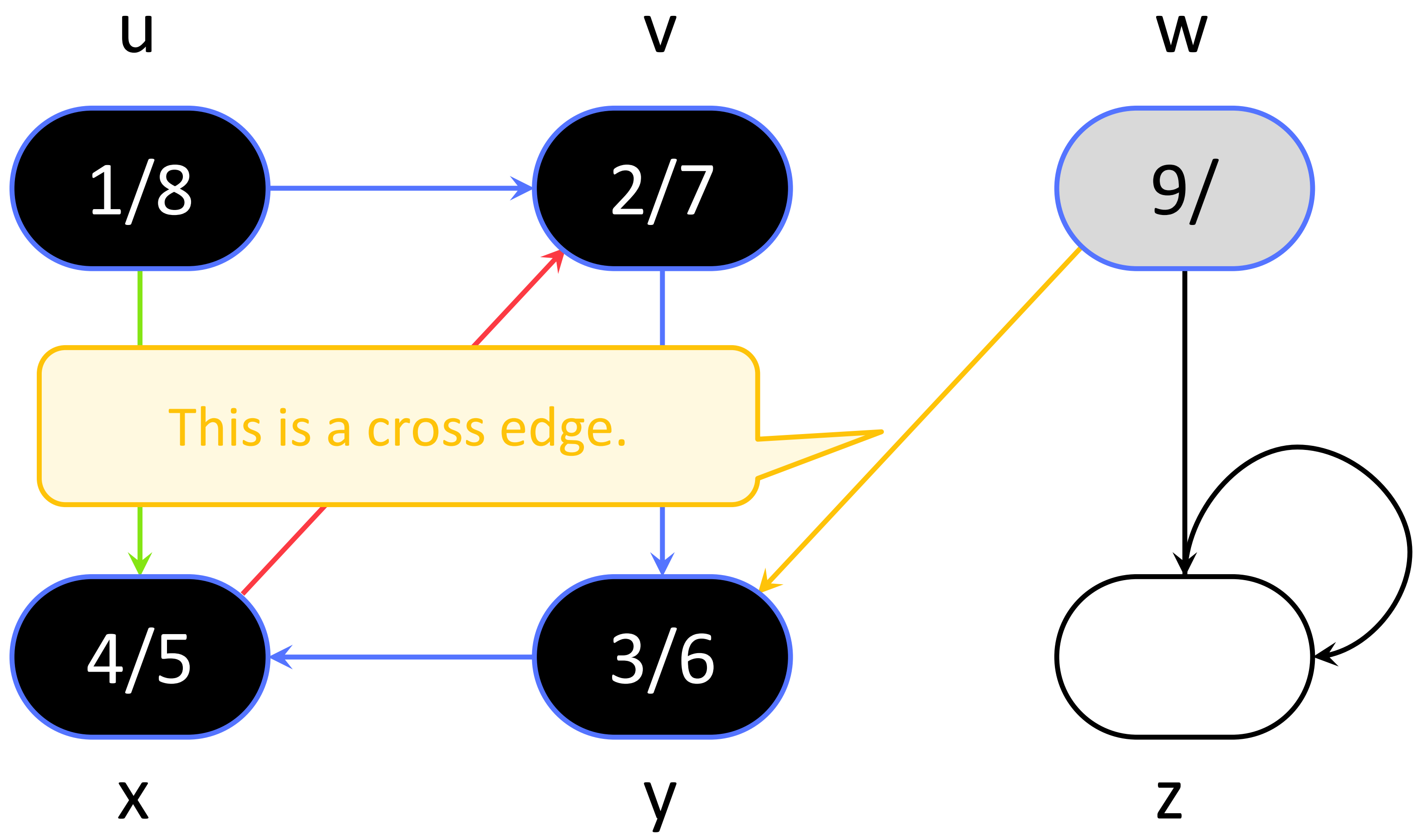
Time: 8



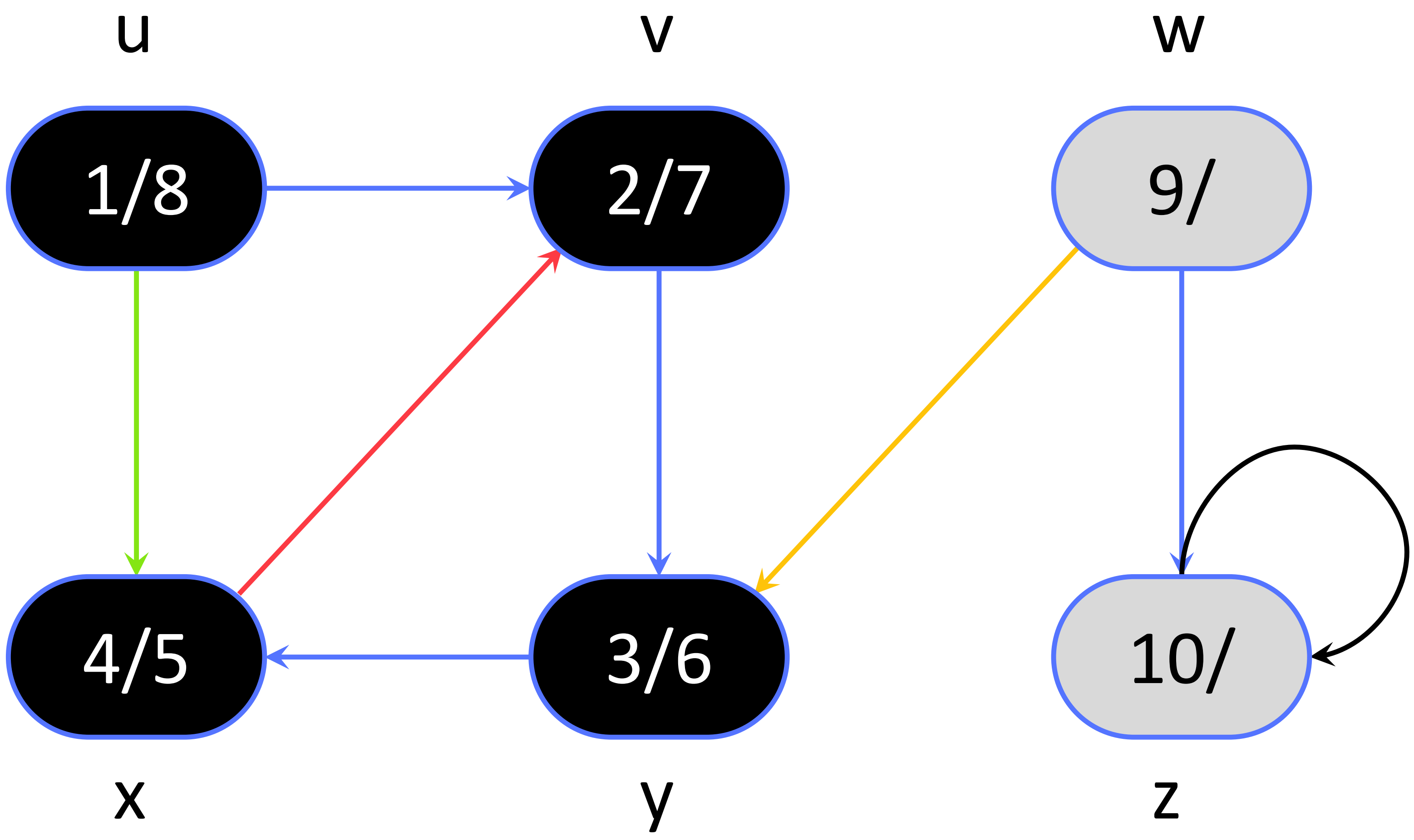
Time: 8



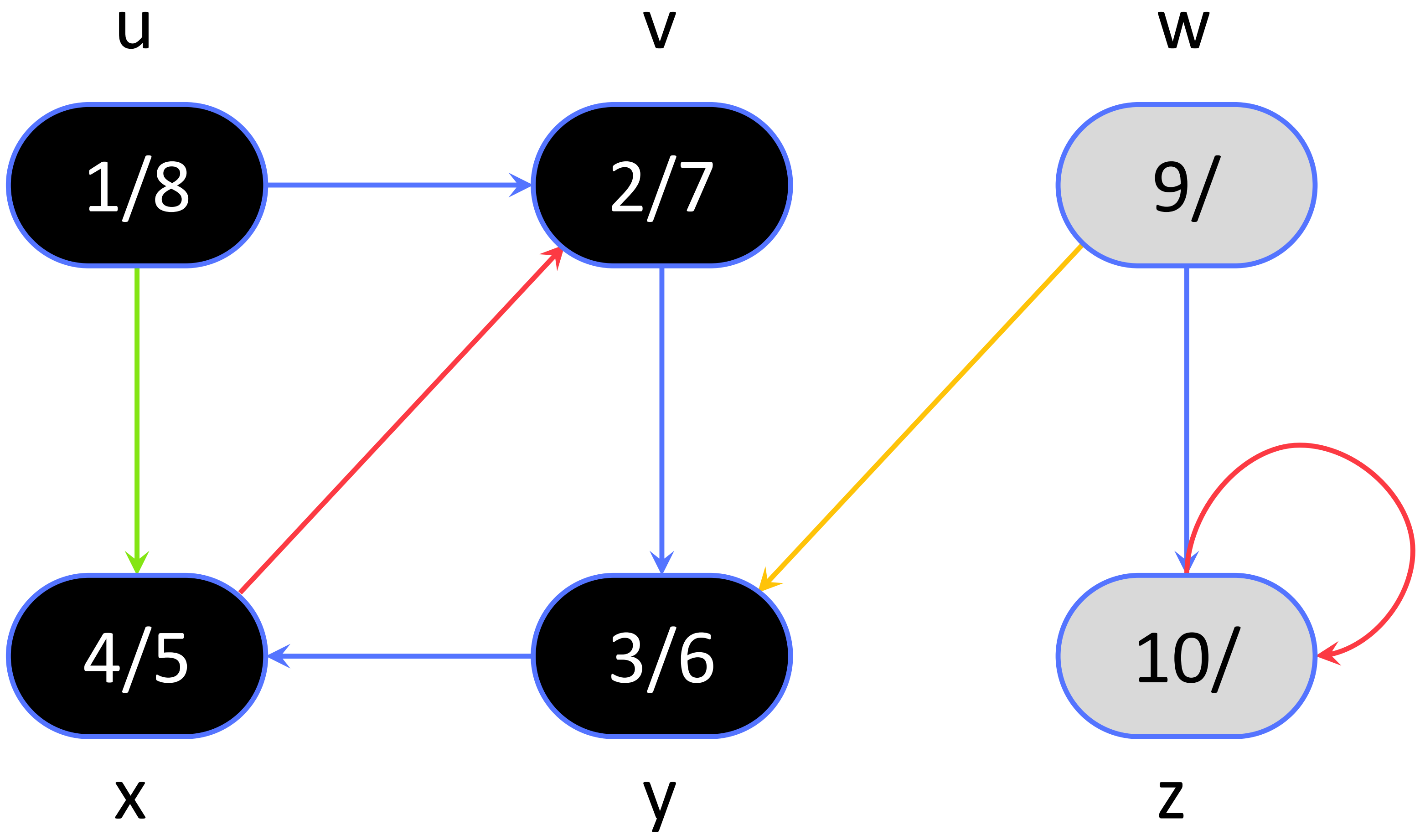
Time: 9



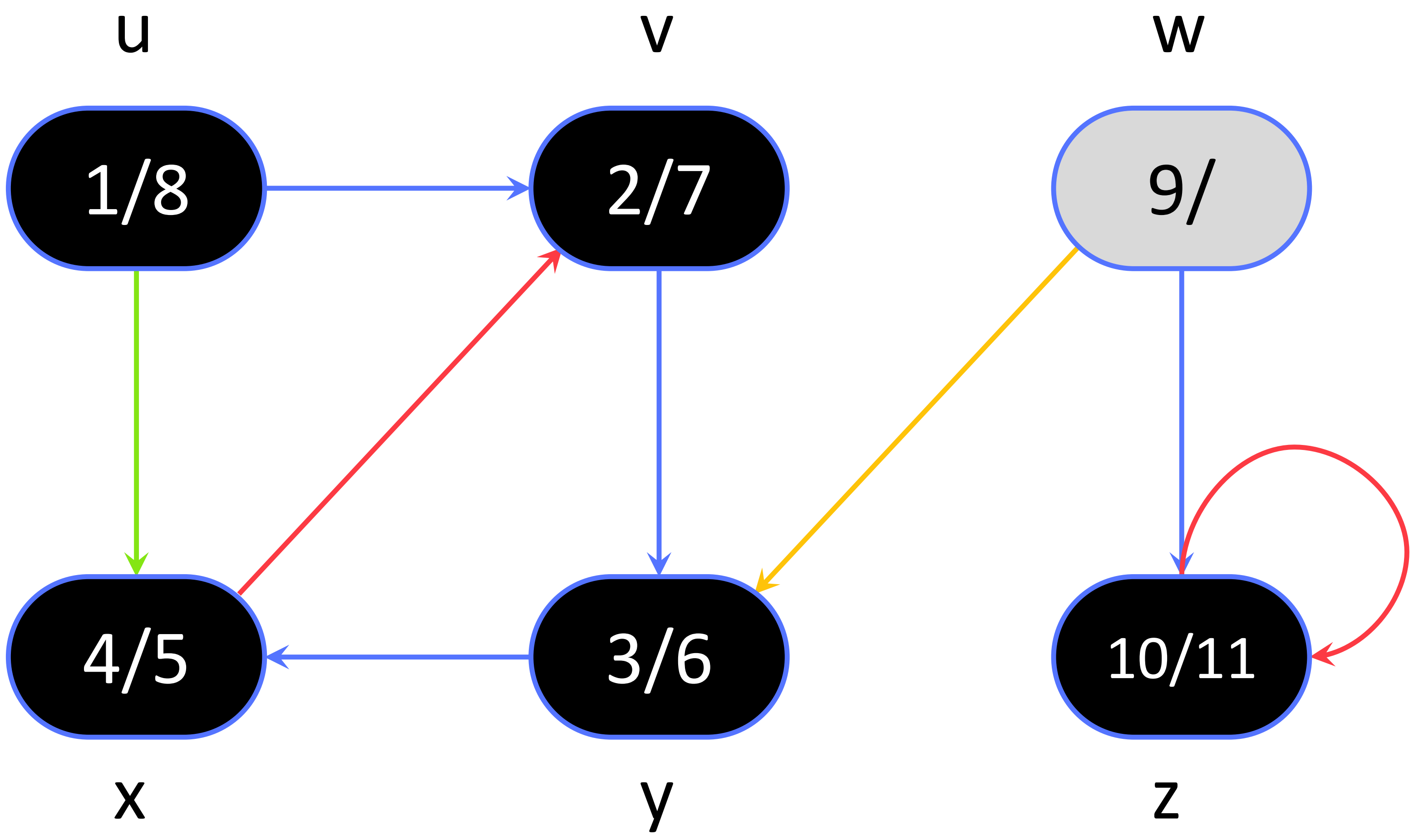
Time: 9



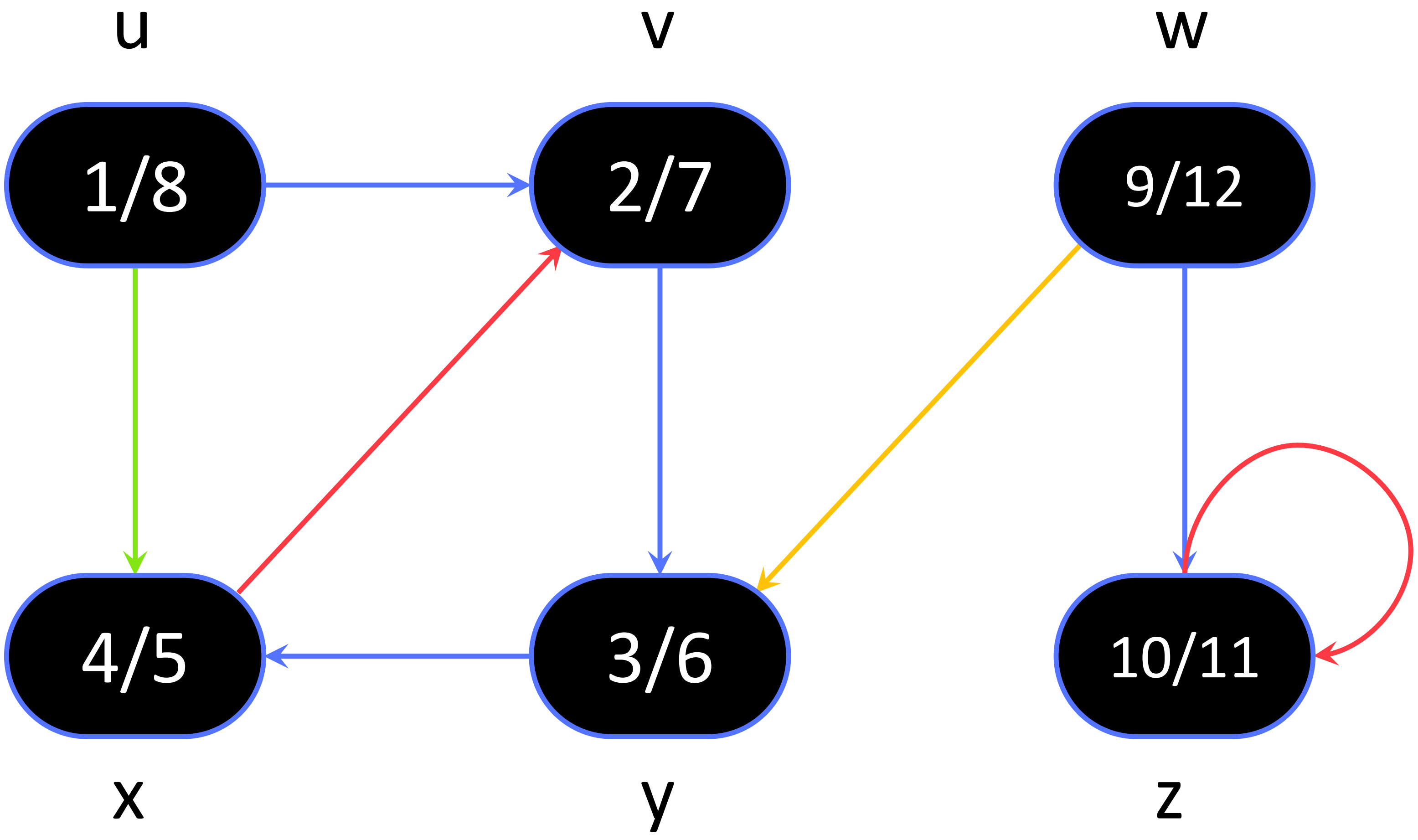
Time: 10



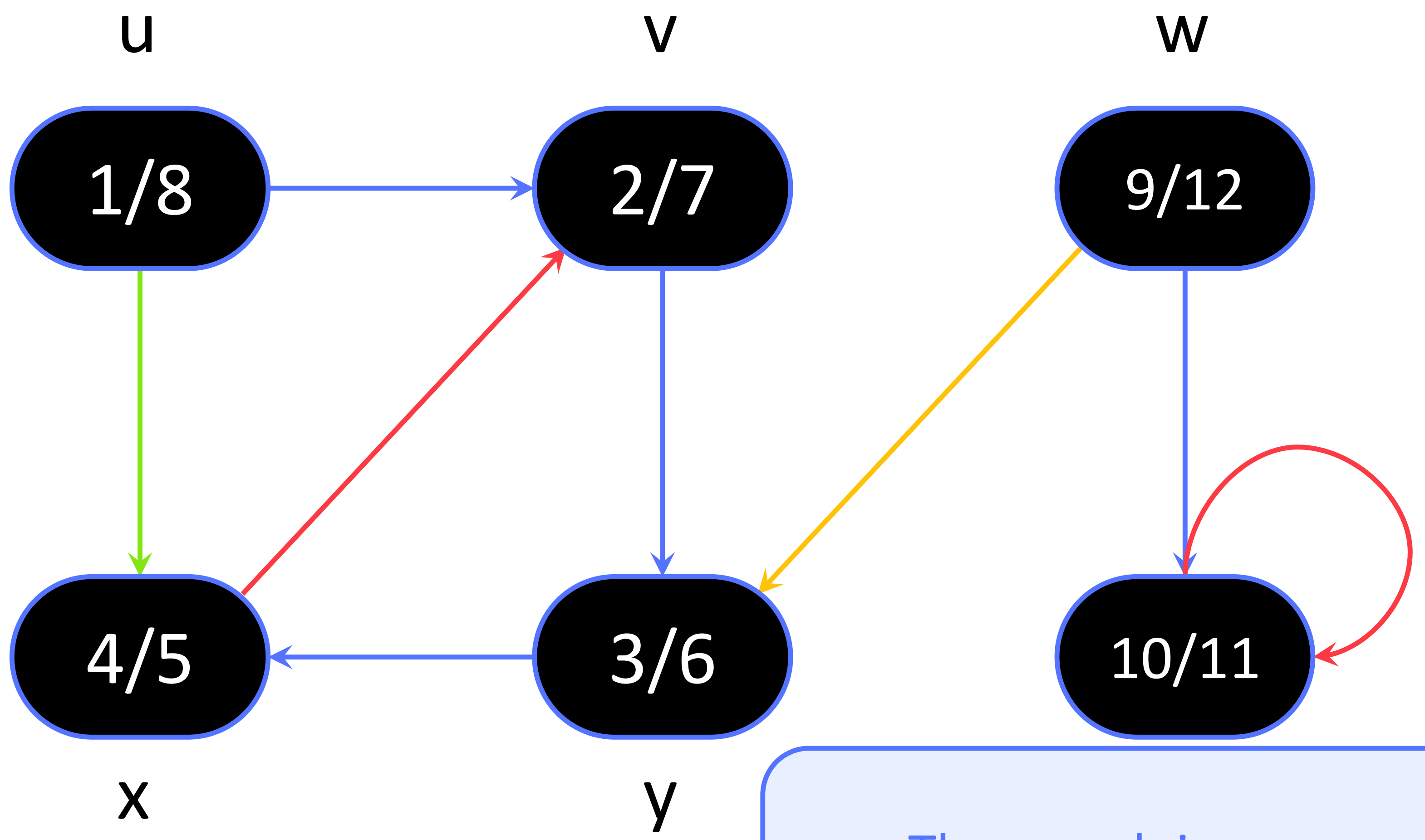
Time: 10



Time: 11



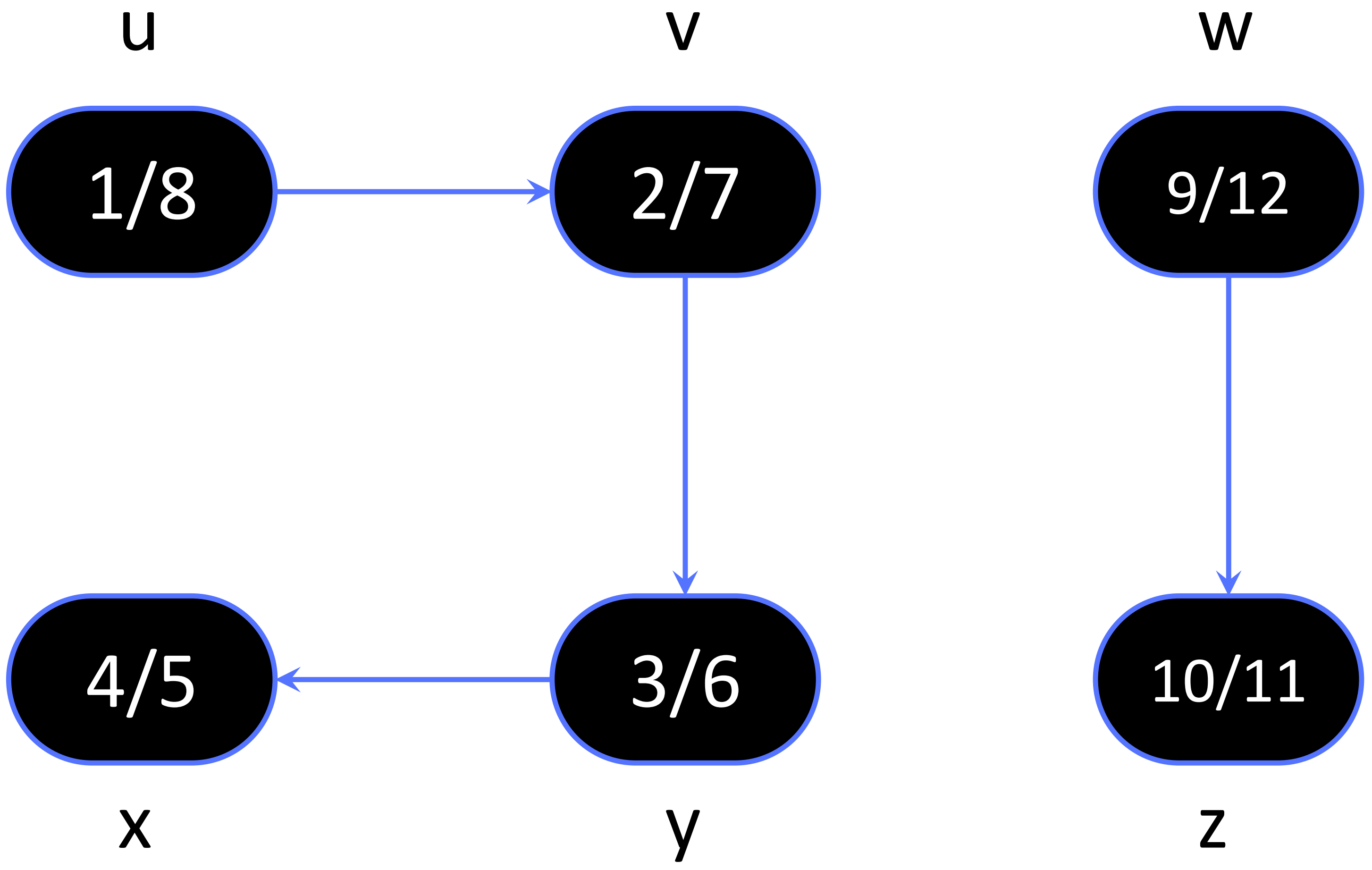
Time: 12



The search is now complete.

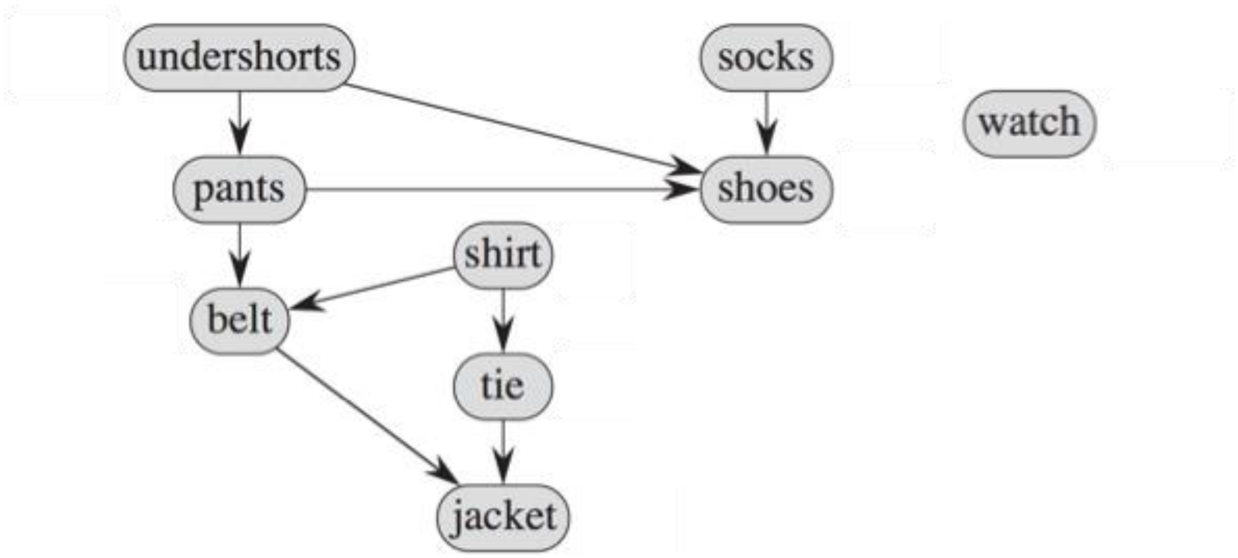
Time: 12

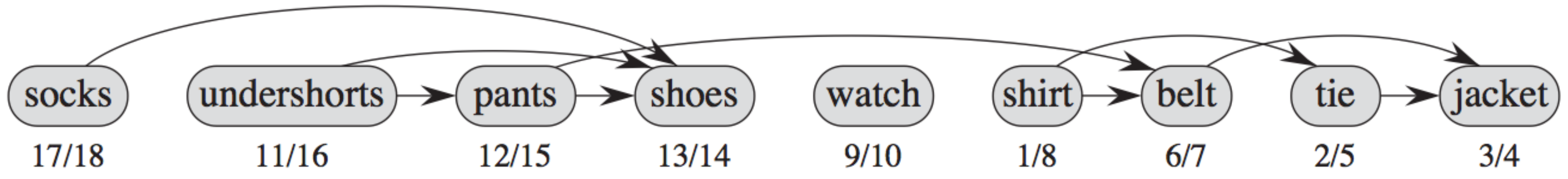
With only tree edges visible, we can see that this DFS path has two trees.



Background of Exercise 1

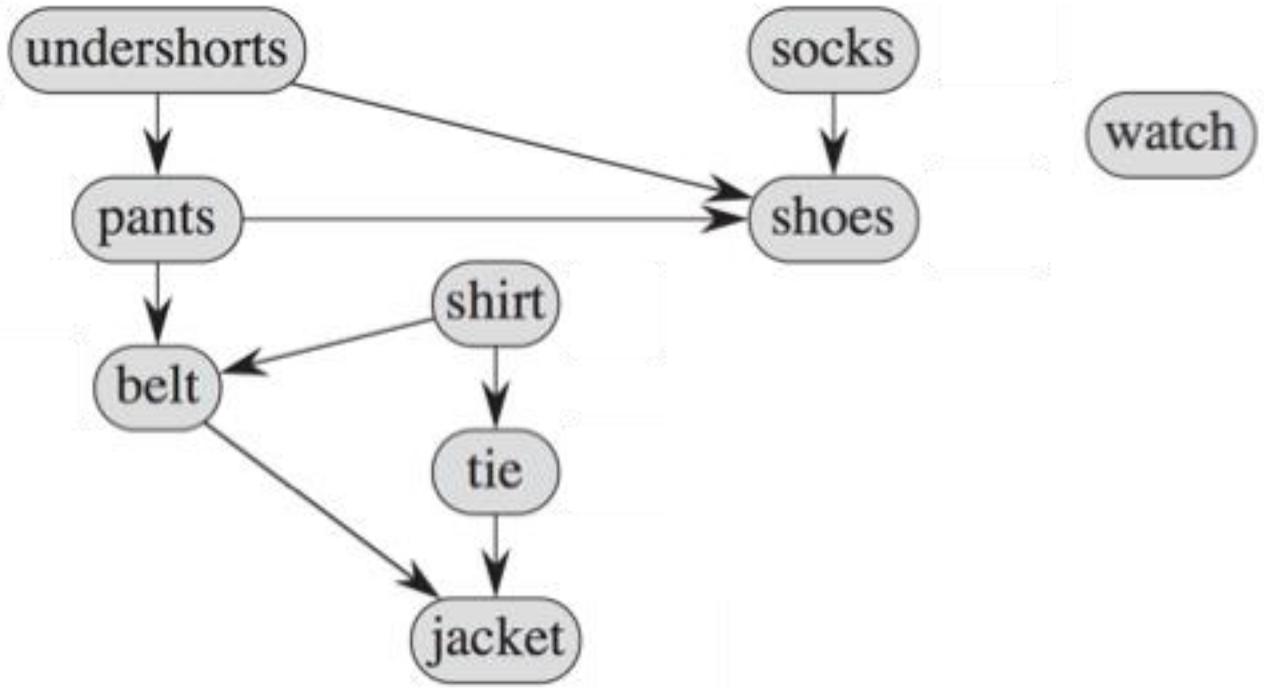
- "I don't want to wear socks first! I want something else first!
- … ok … let's use it as an exercise





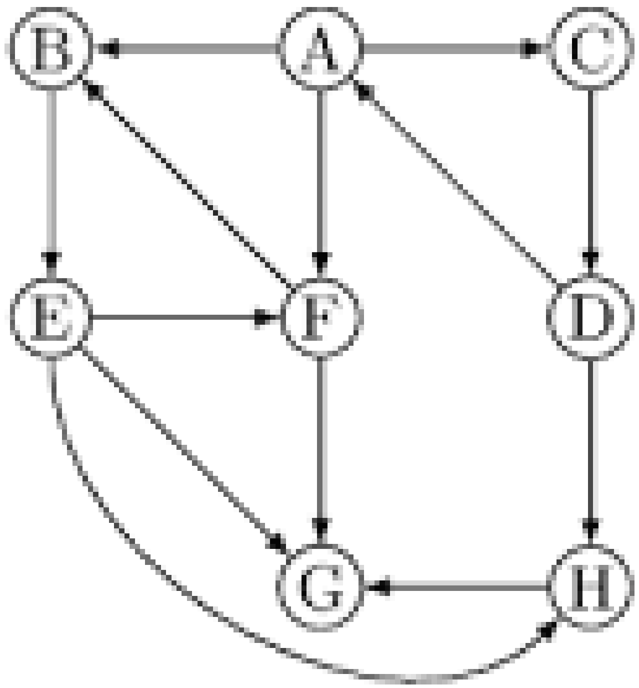
Exercise 1

- Start from undershorts / socks / watch,
  - 1) do another DFS
  - 2) do another topological sort



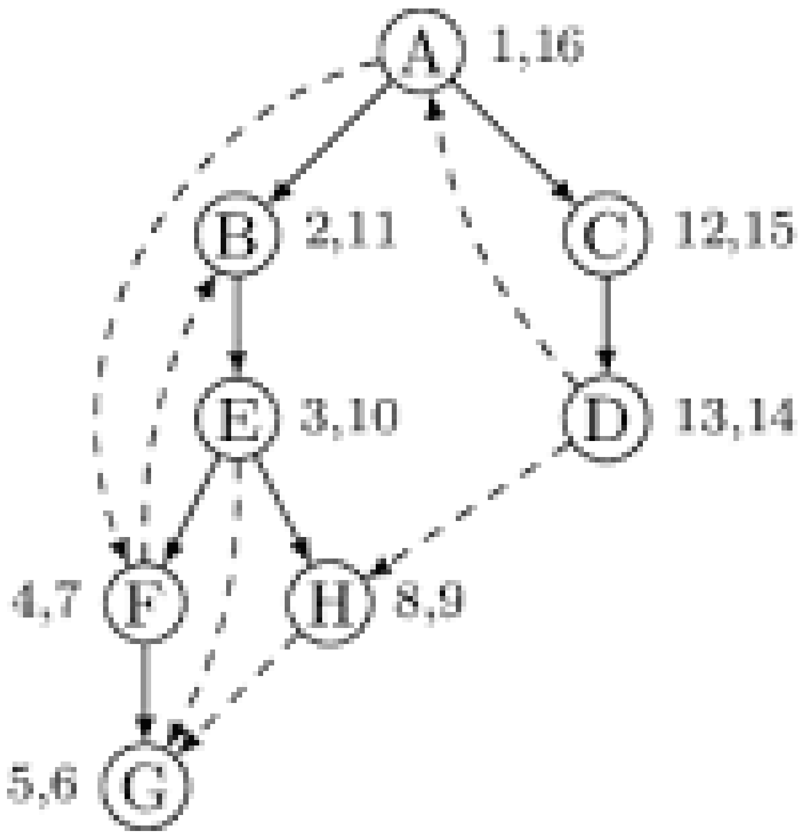
Exercise 2

- Start from A, construct the DFS of the follow.
- Do a Topological search



Exercise 2 answer

One POSSIBLE answer of the DFS tree traversal:



One POSSIBLE answer of the topological sort:
A C D B E H F G