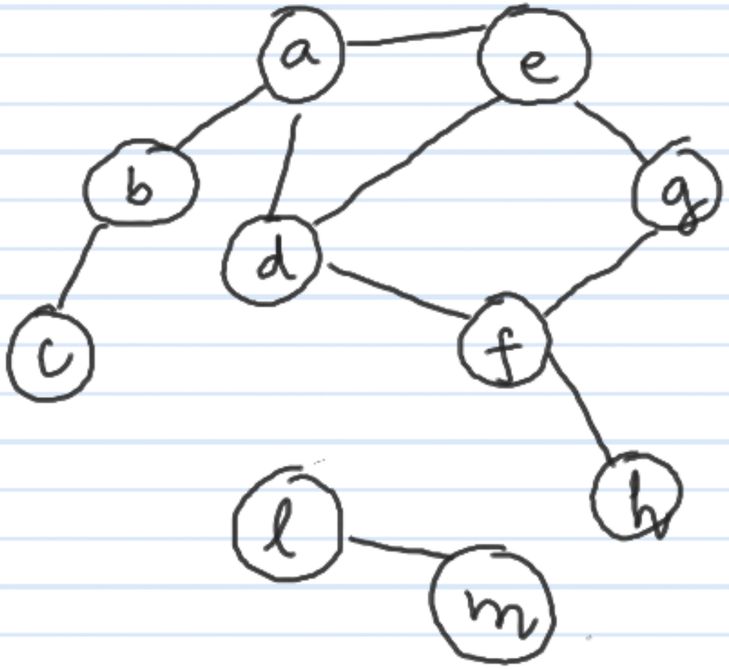
CS2700 : Programming and Data Structures.

GRAPH ADT.

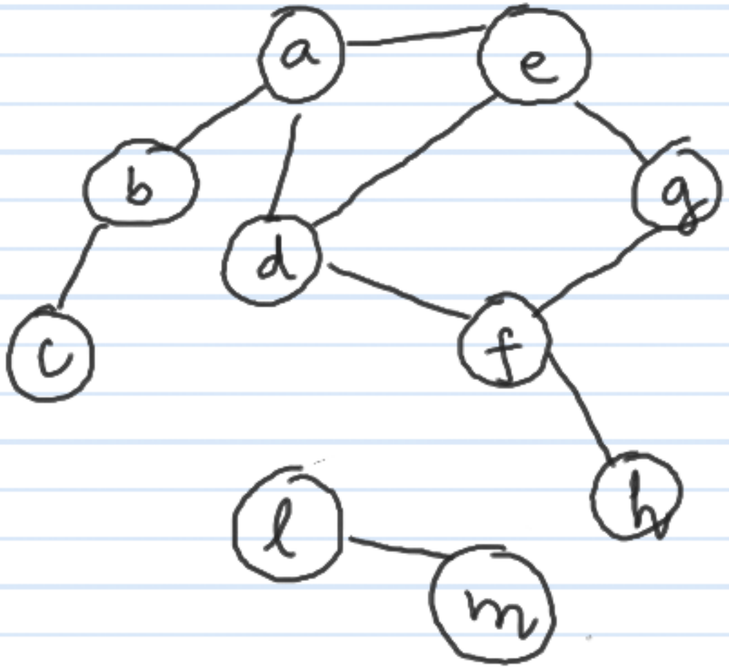
- Graphs in real life.
- How to represent graphs?
- some basic (and very useful) algorithms on graphs.

GRAPH TRAVERSALS ; BREADTH OR DEPTH FIRST?



- How does one "explore" a graph?
- where do you start?
- At vertex "a" do you go to b or d or e?
- After visiting b, do you go "deeper" or do you go "broader"?

GRAPH TRAVERSALS : DEPTH FIRST.



- Explore "deeper" before "breadth"

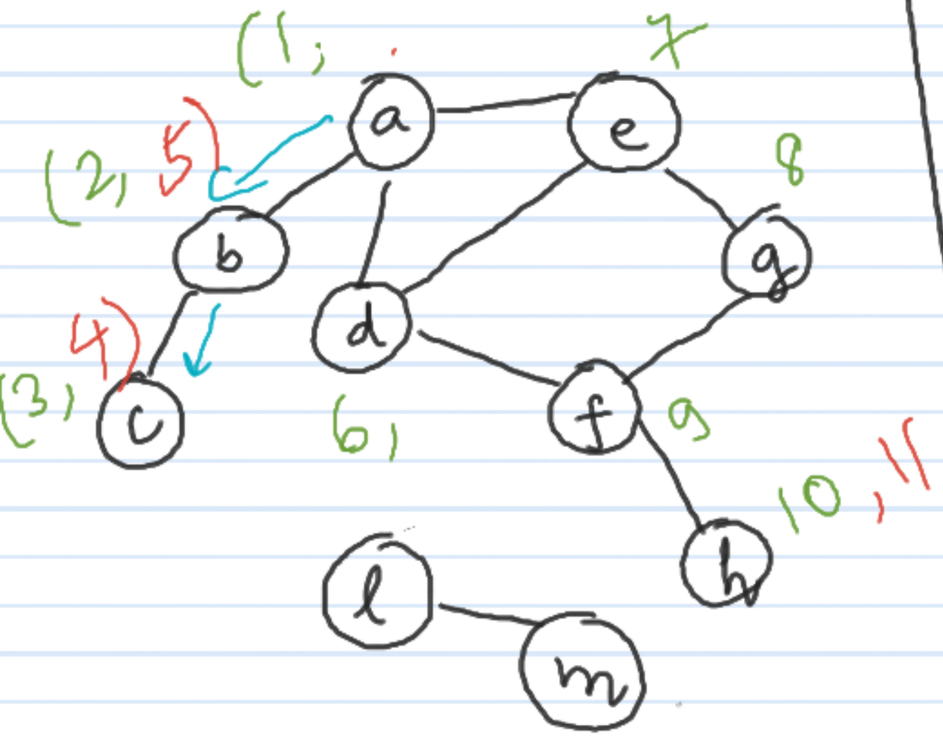
- Computes a DFS tree starting at a source node.

- Uses a stack to keep track of the vertices

or is written recursively

- DFS tree has interesting and useful properties

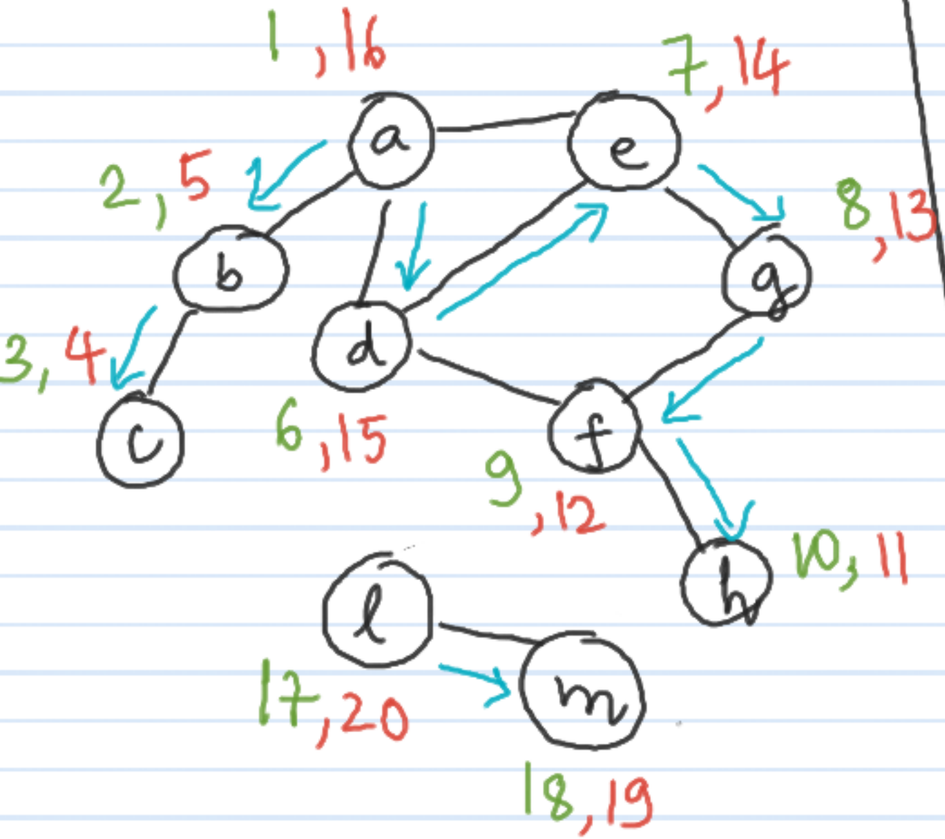
GRAPH TRAVERSALS : DEPTH FIRST.




```
DFS(G)

∀u ∈ V { clr(u) = white
       { π(u) = 1

time = 0;

for every u in G
    if clr(u) == white
        DFS-visit(G,u)
```

```
DFS-visit(G,u)

time++;
u.d = time;
clr(u) = gray;

for every v ∈ Nbr(u)
    if (clr(v) == white)
        π(v) = u;
        DFS-visit(G,v)
    }
}
clr(u) = Black;
time++
u.f = time
```

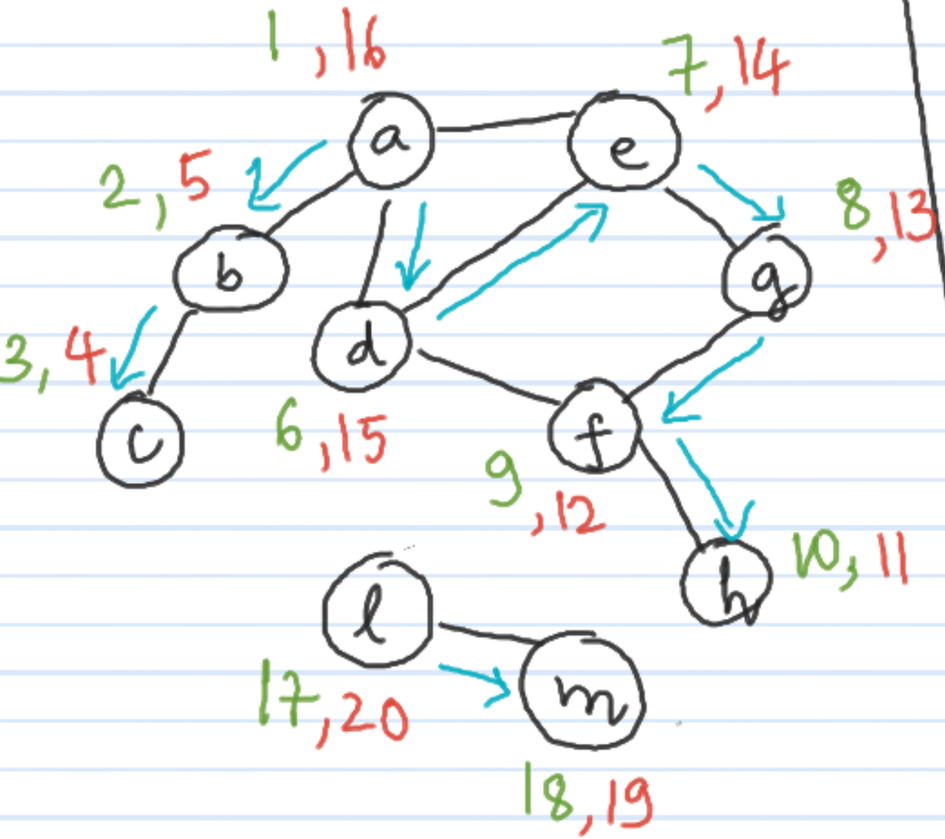
GRAPH TRAVERSALS : DEPTH FIRST.




```
DFS(G)

∀u∈V { clr(u) = white
       π(u) = 1

time = 0;

for every u in G
    - if clr(u) == white
          DFS-visit(G,u)
```

```
DFS-visit(G,u)

time++;
u.d = time;
clr(u) = gray;
for every v ∈ Nbr(u)
    if(clr(v) == white)
        π(v) = u;
        DFS-visit(G,v)
    }
}
clr(u) = Black;
time++
u.f = time
```

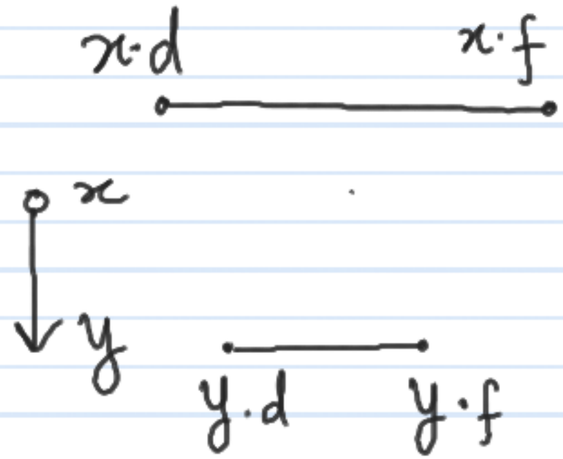
GRAPH TRAVERSALS : DEPTH FIRST.



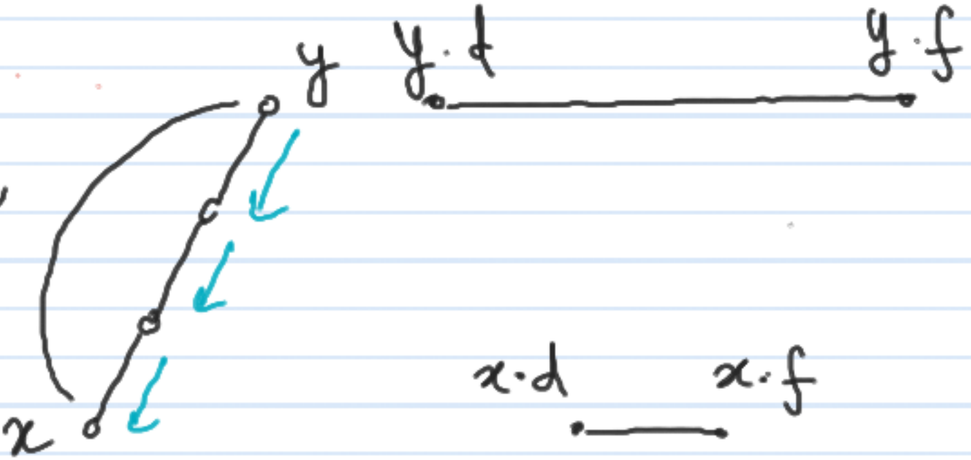
Undirected graphs

2 types of edges.

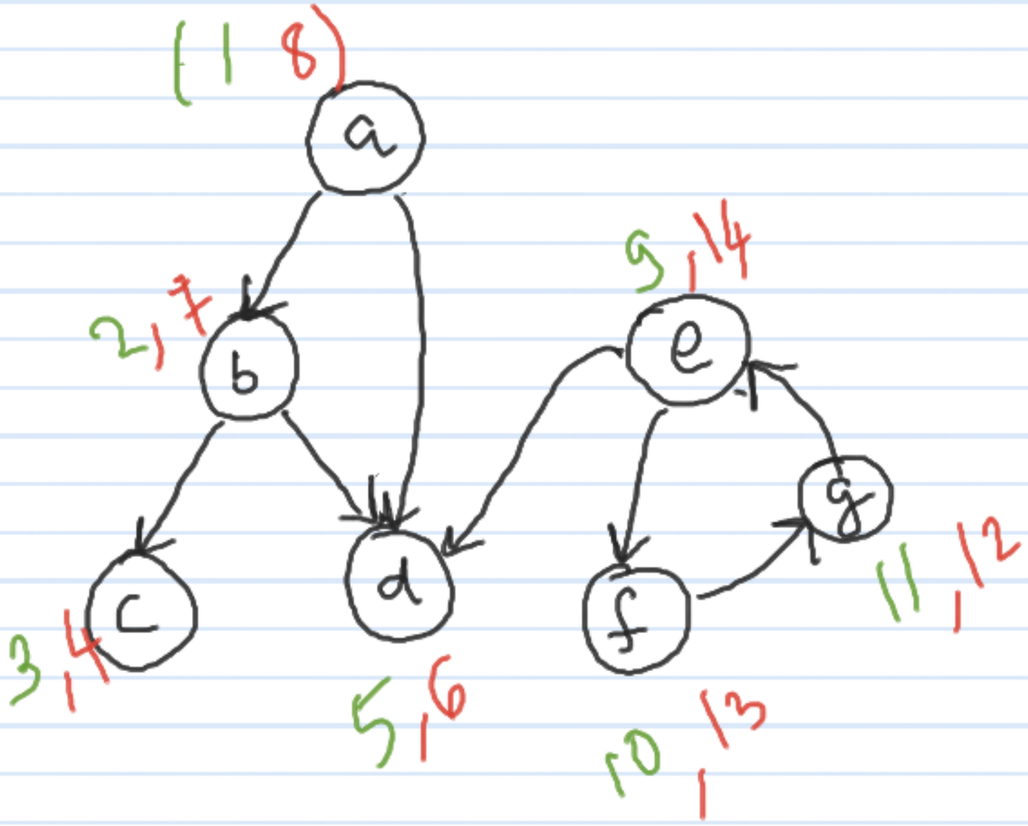
- Tree edge



- Back edge

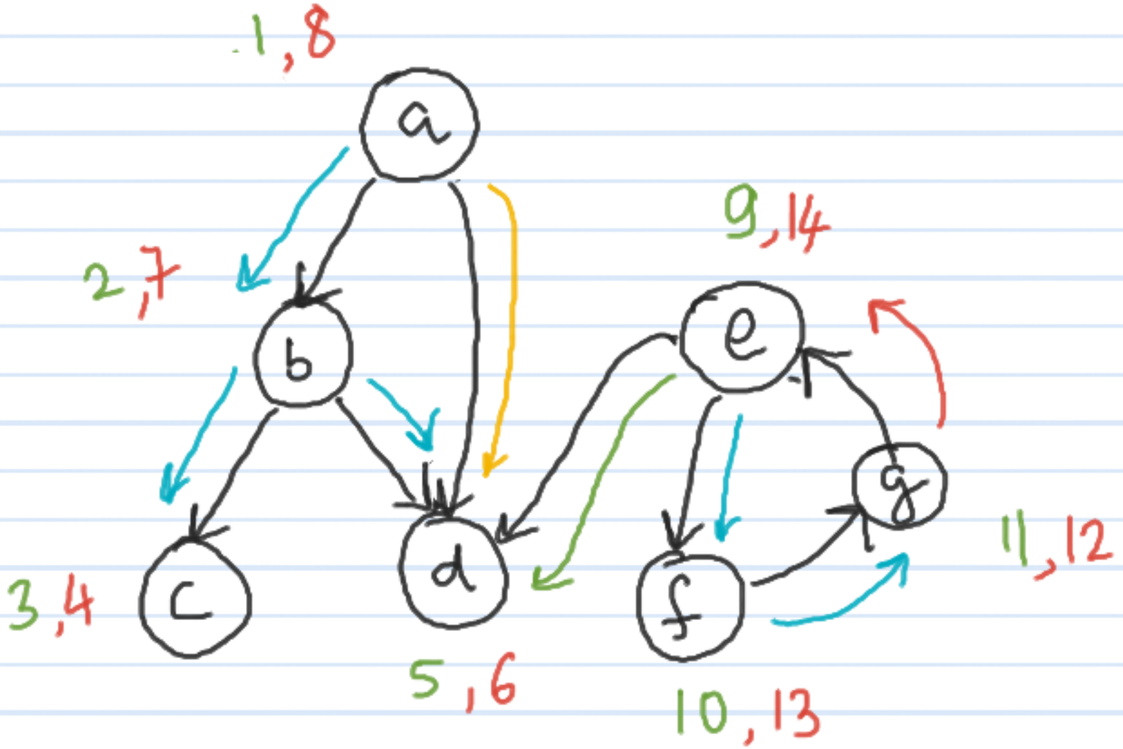


GRAPH TRAVERSALS : DEPTH FIRST.



- perform a DFS traversal of the graph
- what are the different types of edges?

GRAPH TRAVERSALS : DEPTH FIRST.



- perform a DFS traversal of the graph

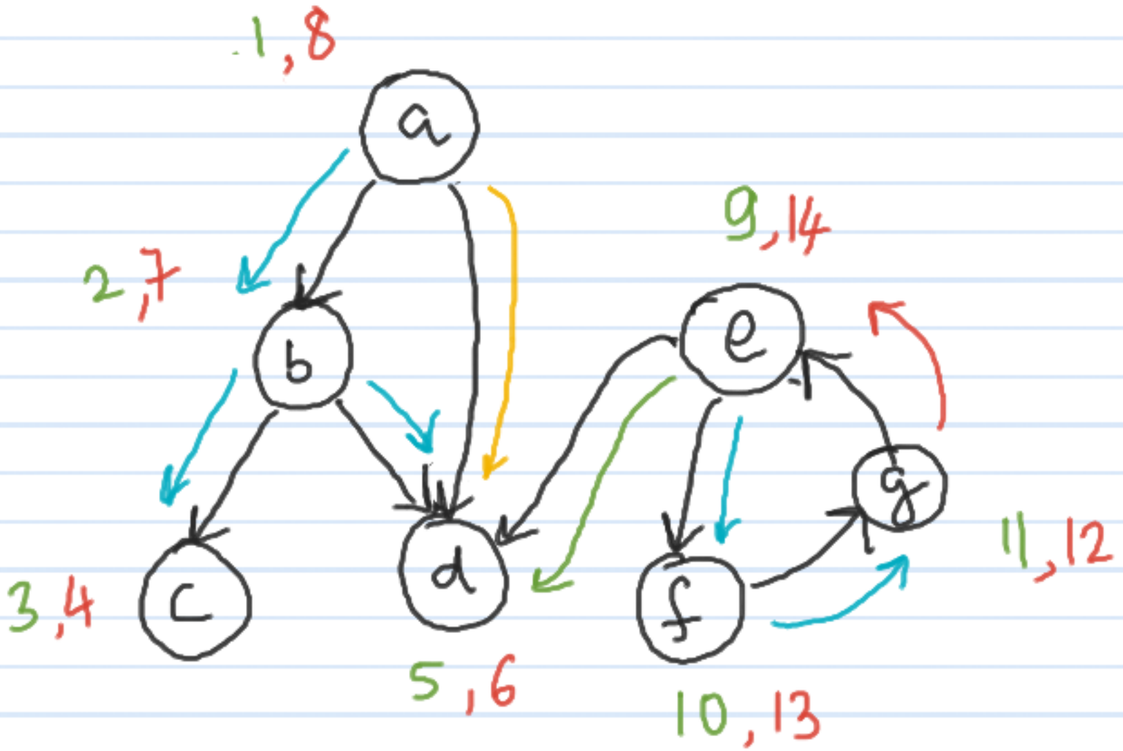
- What are the different kinds of edges

→ Back edge
(x, y)

→ Forward edge
(x, y)

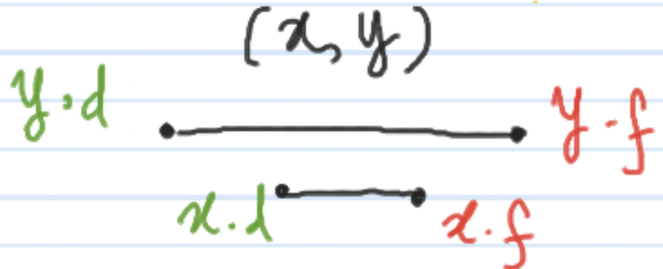
→ Cross edge
(x, y)

GRAPH TRAVERSALS : DEPTH FIRST.

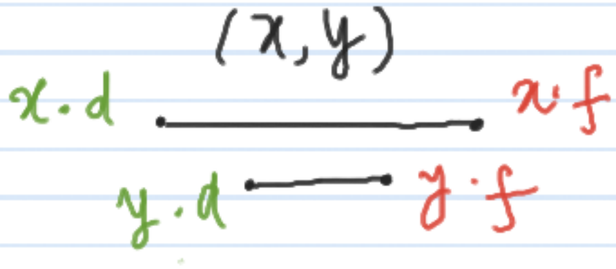


- perform a DFS traversal of the graph
- What are the different kinds of edges

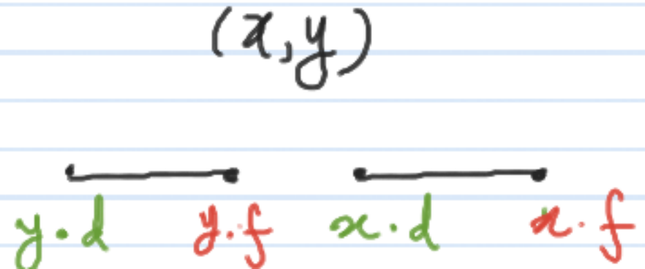
→ Back edge



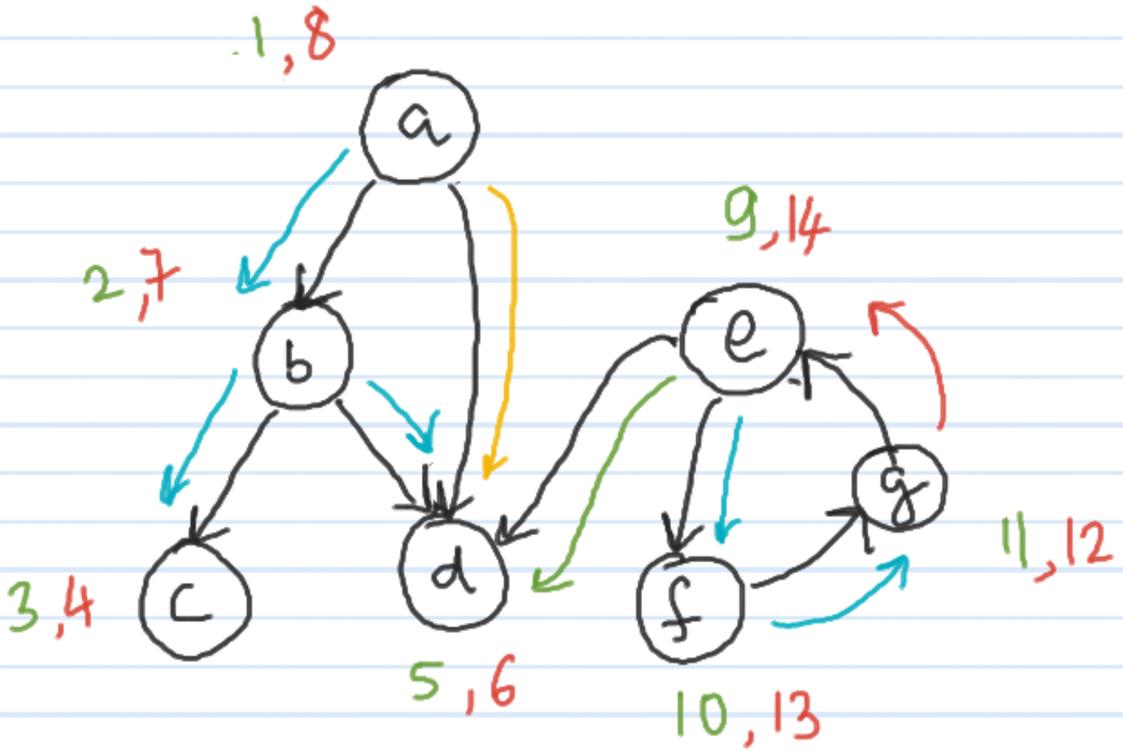
→ Forward edge



→ Cross edge

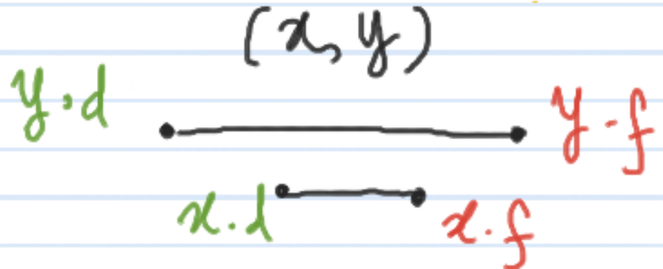


GRAPH TRAVERSALS : DEPTH FIRST.

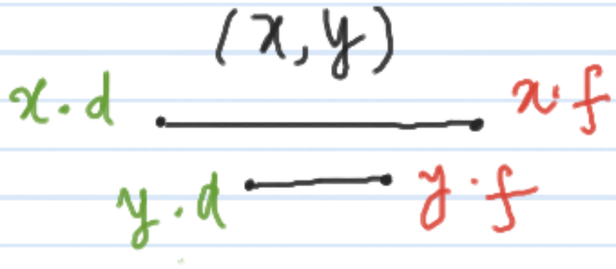


- perform a DFS traversal of the graph
- What are the different kinds of edges

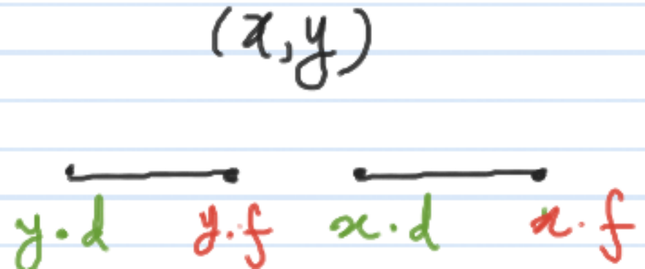
→ Back edge



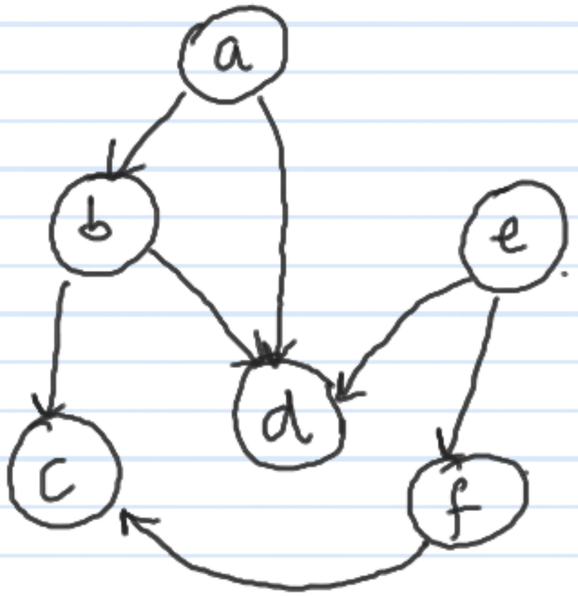
→ Forward edge



→ Cross edge

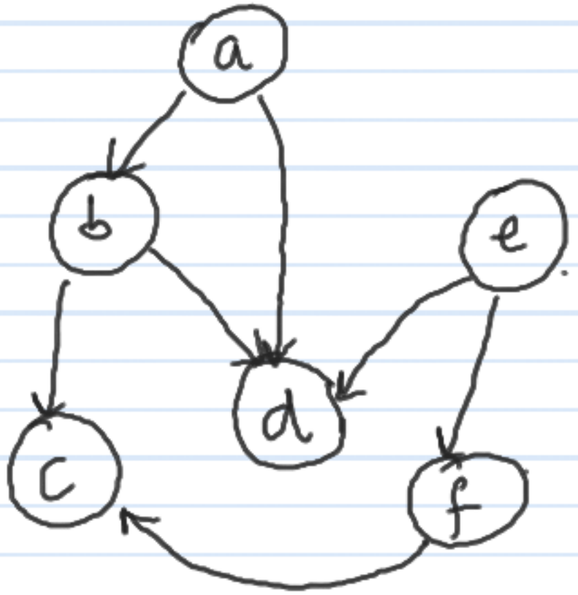


DIRECTED ACYCLIC GRAPHS (DAGs)



Given a graph in adjacency list form, detect if it is a DAG.

DIRECTED ACYCLIC GRAPHS (DAGs)

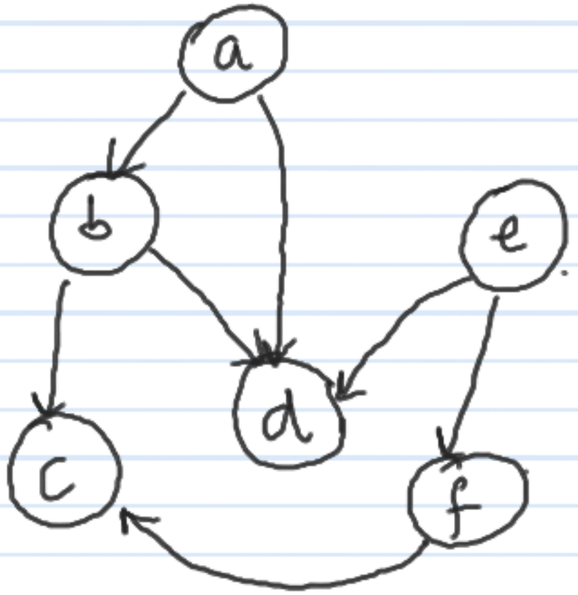


Given a graph in adjacency list form, detect if it is a DAG.

- What type of edges are forbidden?

- Tree edges
- Back edges
- Forward edges
- Cross edges

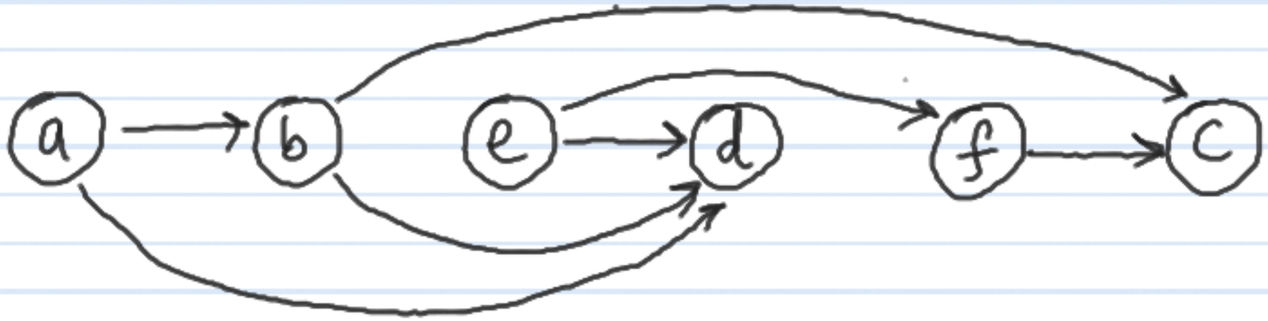
DIRECTED ACYCLIC GRAPHS (DAGs)



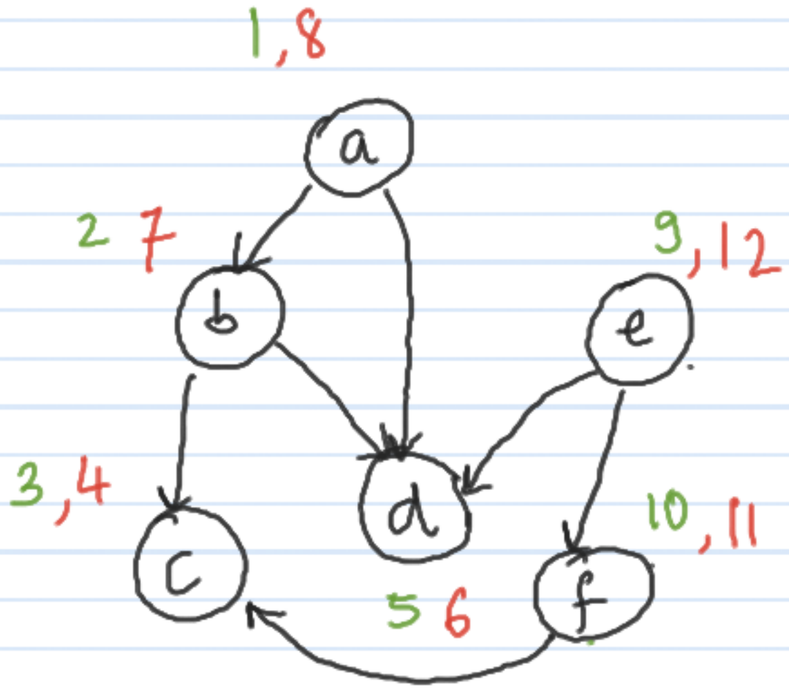
Given a graph in adjacency list form, detect if it is a DAG.

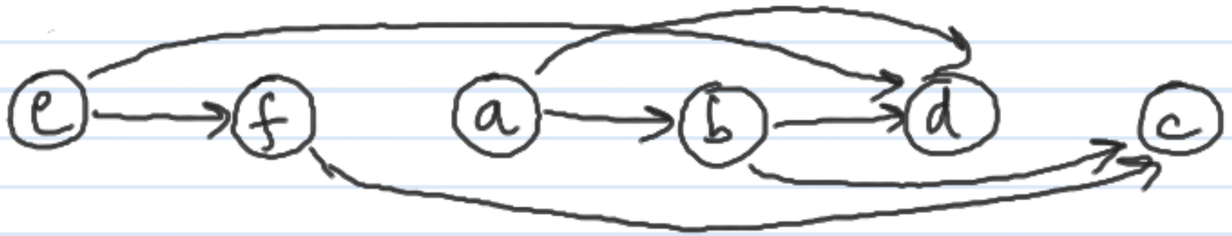
Vertices of a DAG can be linearly ordered s.t. all edges in G go from left to right.

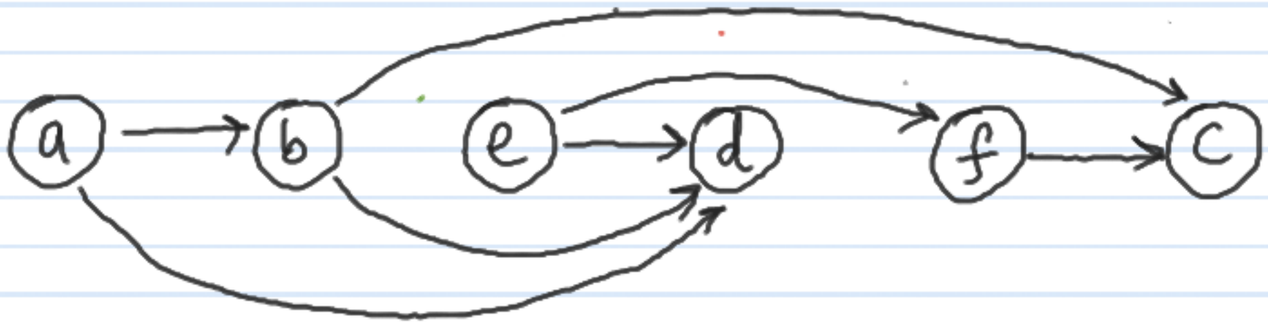
How to generate this ordering?



DIRECTED ACYCLIC GRAPHS (DAGs)



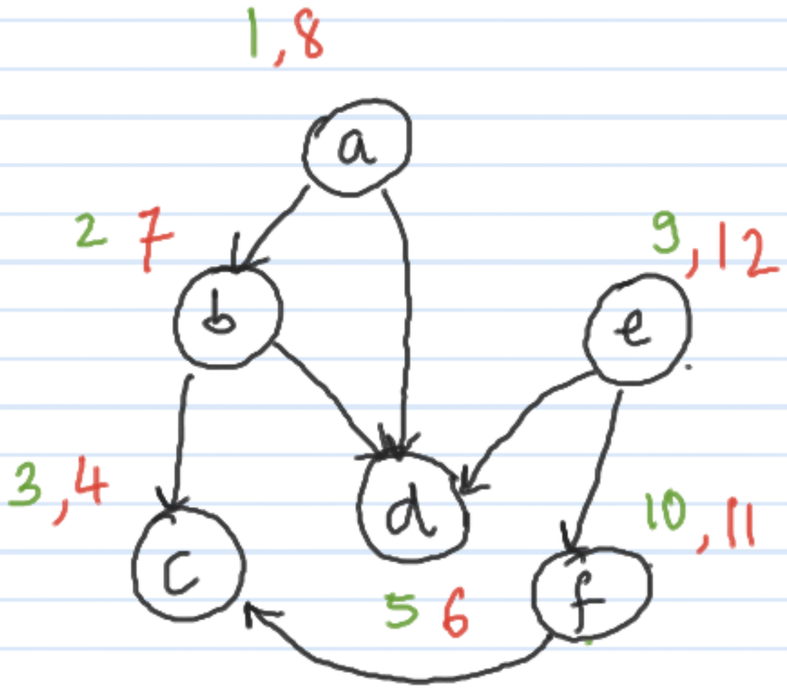


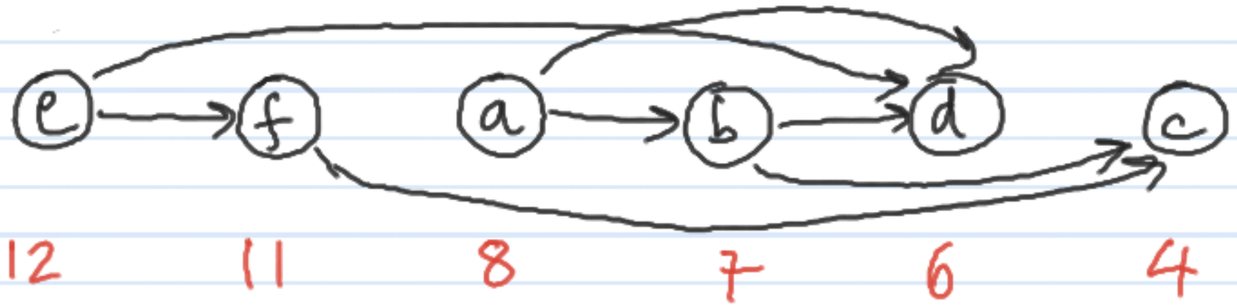


Vertices of a DAG can be linearly ordered s.t. all edges in G go from left to right.

How to generate this ordering?

DIRECTED ACYCLIC GRAPHS (DAGs)





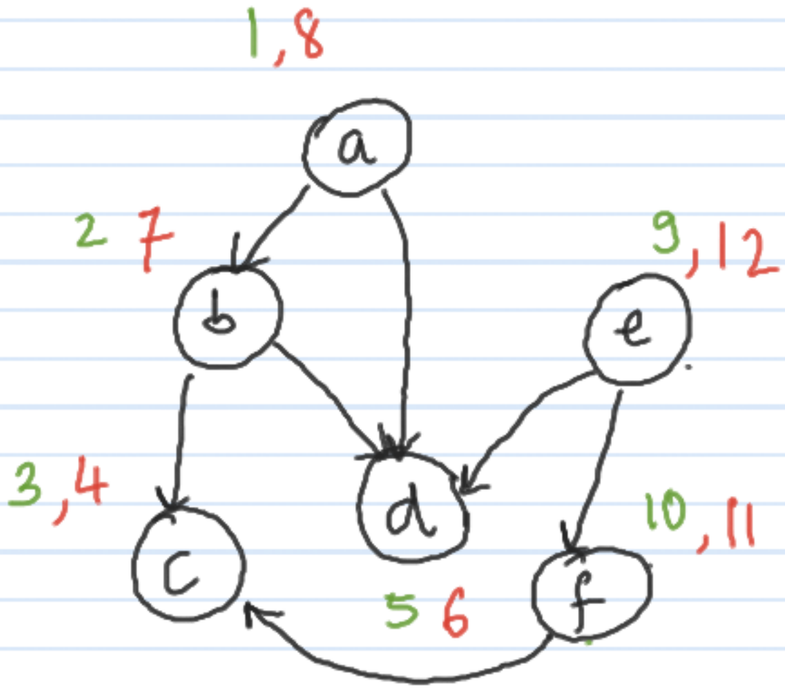
Topological Sort :

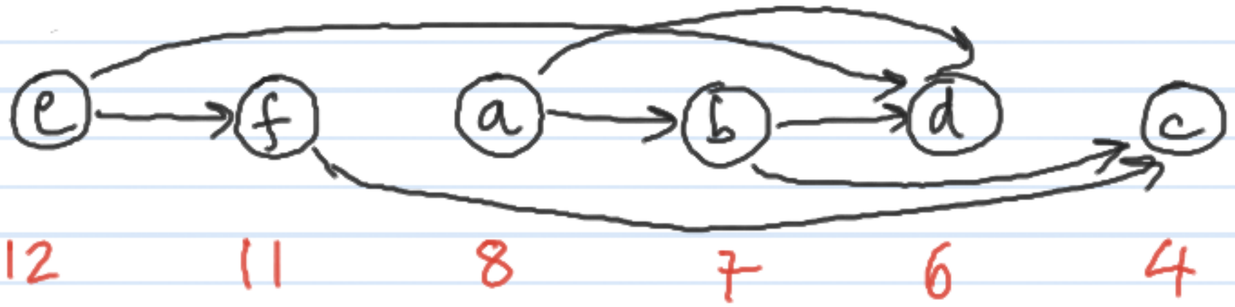
- perform DFS
- reverse order vertices based on finish times

---

does this always produce a correct order?

DIRECTED ACYCLIC GRAPHS (DAGs)





Topological Sort :

- perform DFS
- reverse order vertices based on finish times

$x \circ\!\!\longrightarrow\!\!\circ\, y$

$x.f$ $y.f$

suffices to show : $y.f < x.f$