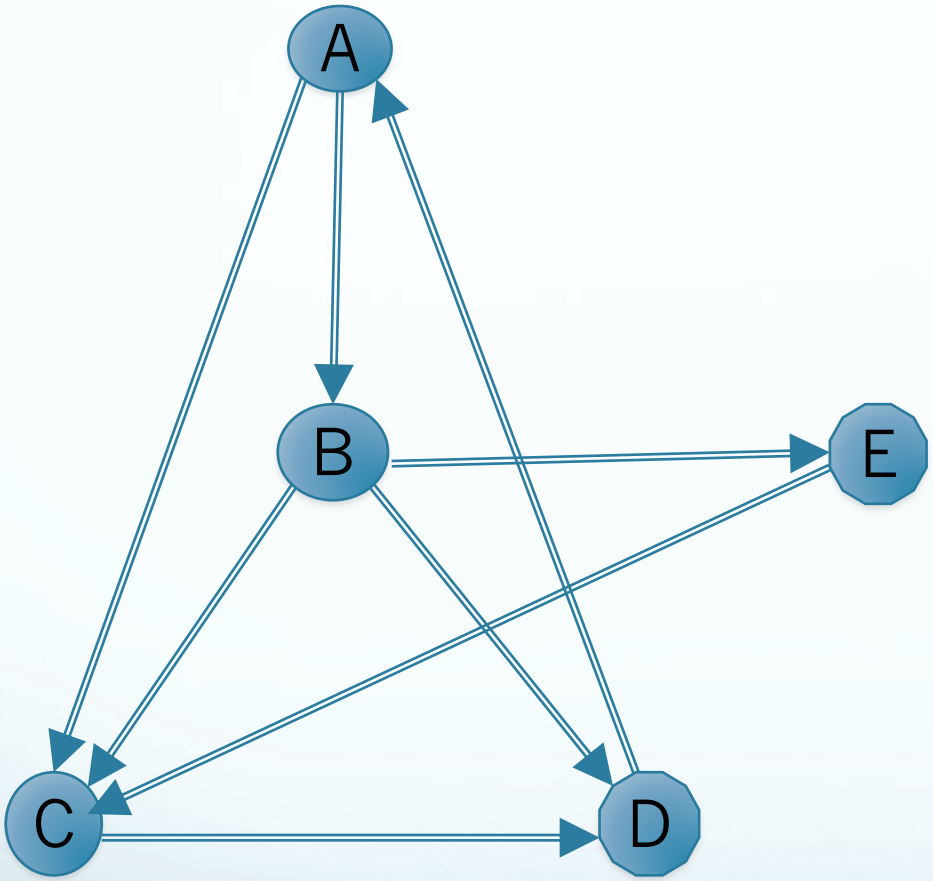
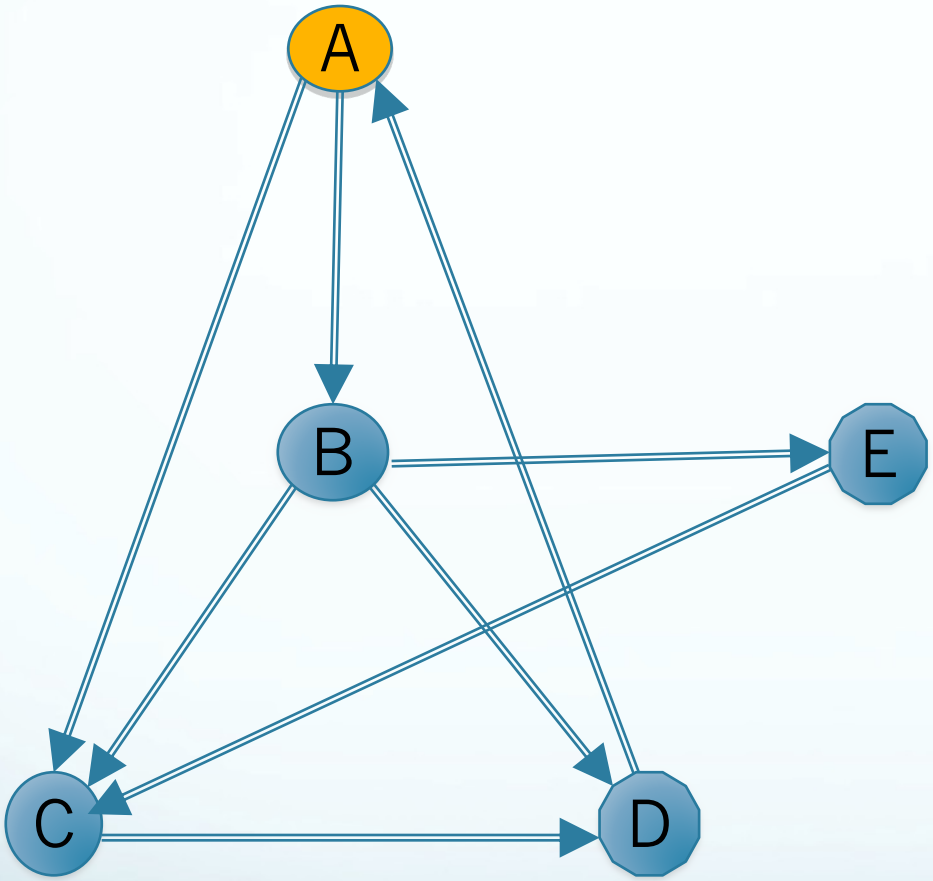
Extra for Workshop W5: DFS in di-graphs, and stack discipline, push & pop order, tree edges etc

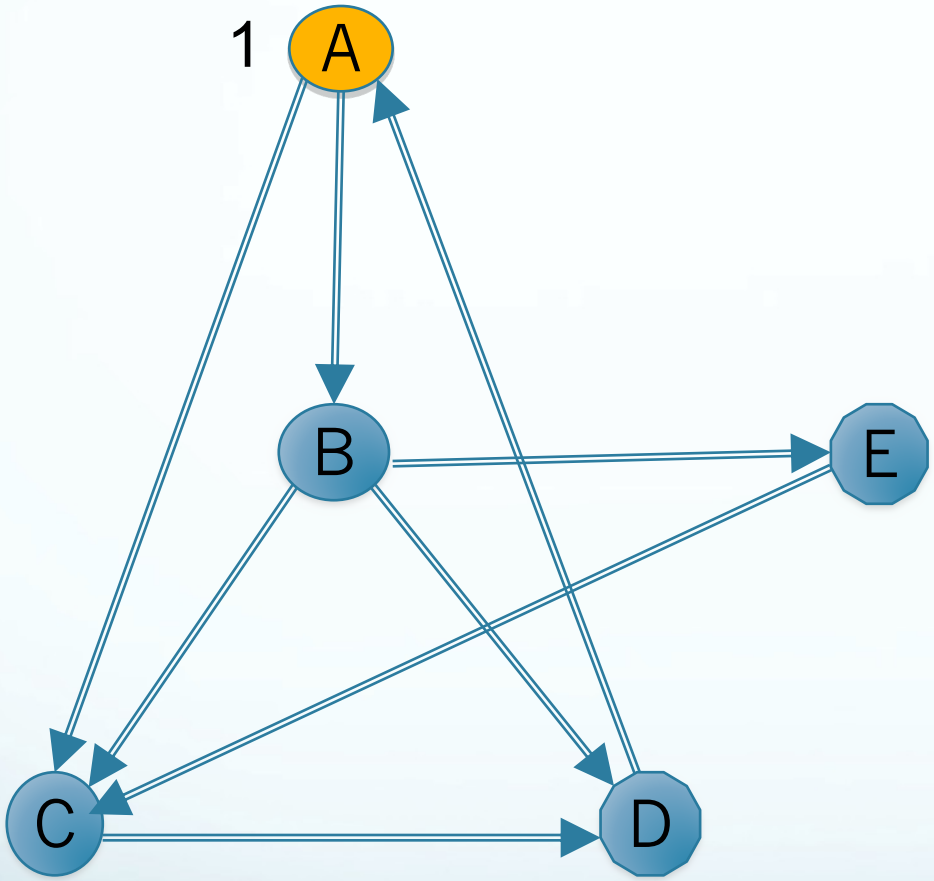


DfsExplore(A)



At the beginning stack is empty:
Stack content:
$
($ show the bottom of stack, `push` to, and `pop` from, the right)
Supposing the DFS starts from A.

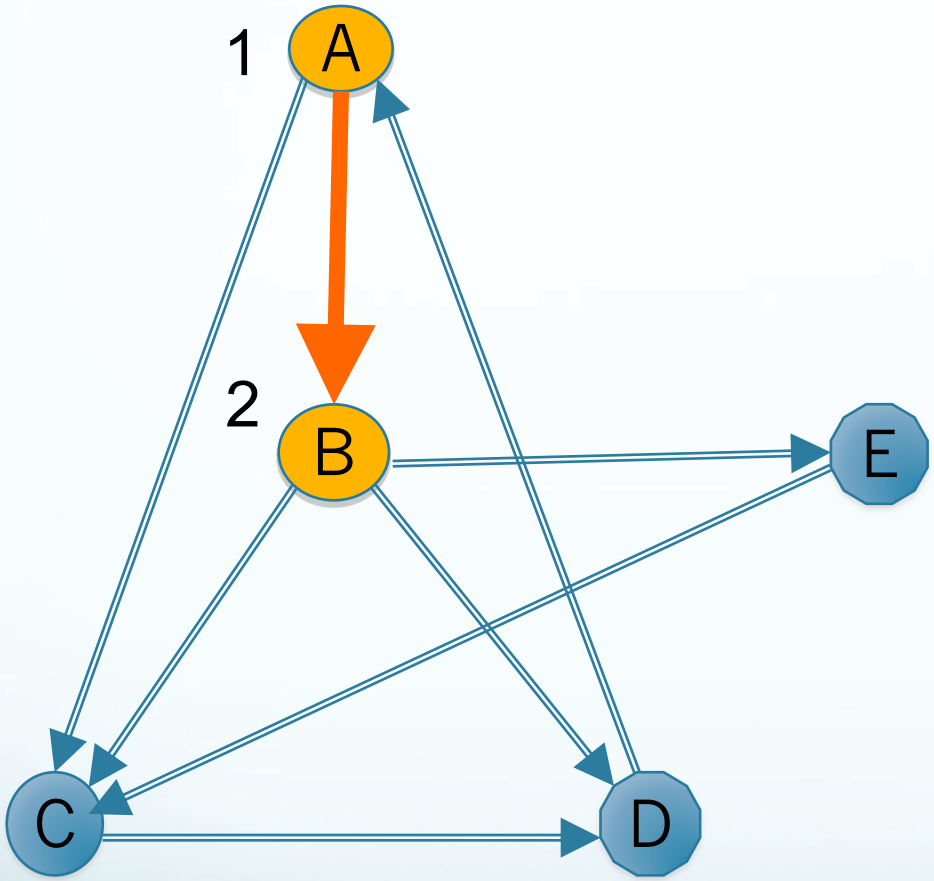
DFS, stack



Visit A. A now is in stack and has the push order 1, which is shown by a black number of the left of A.

Stack:
$A

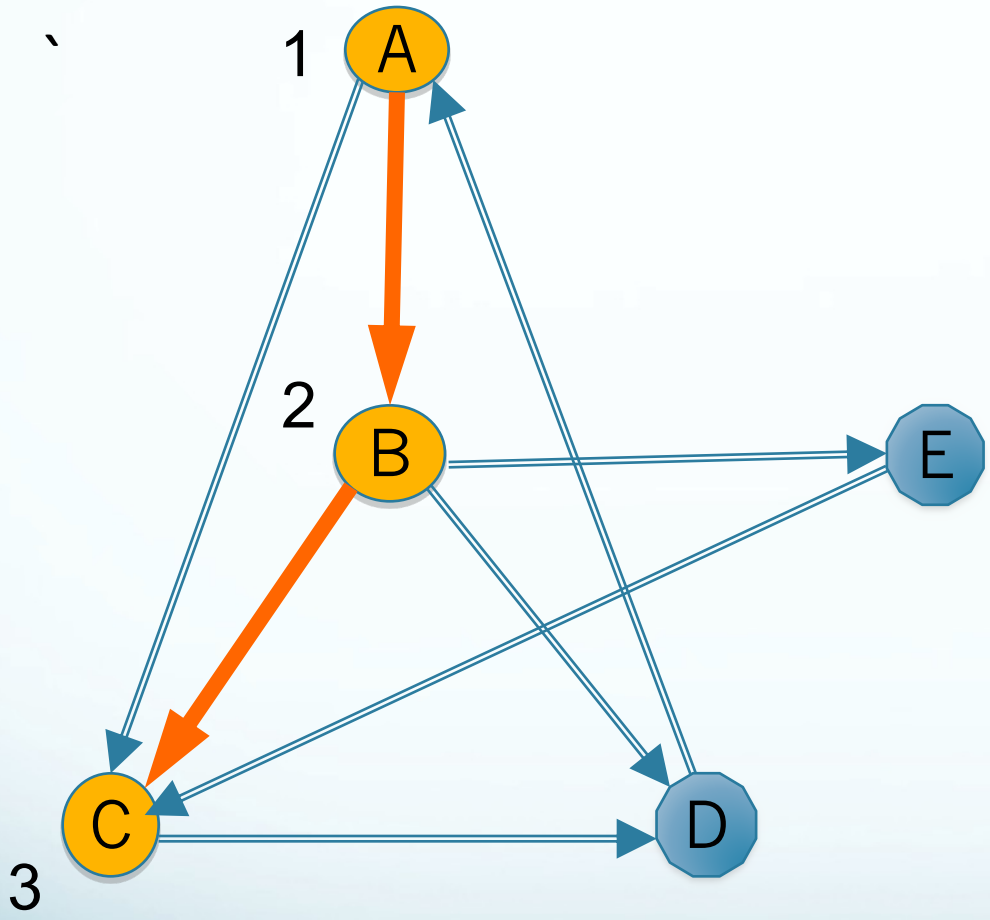
DFS



Visit B (following the alphabetic order, B is chosen instead of C).

Stack:
$AB

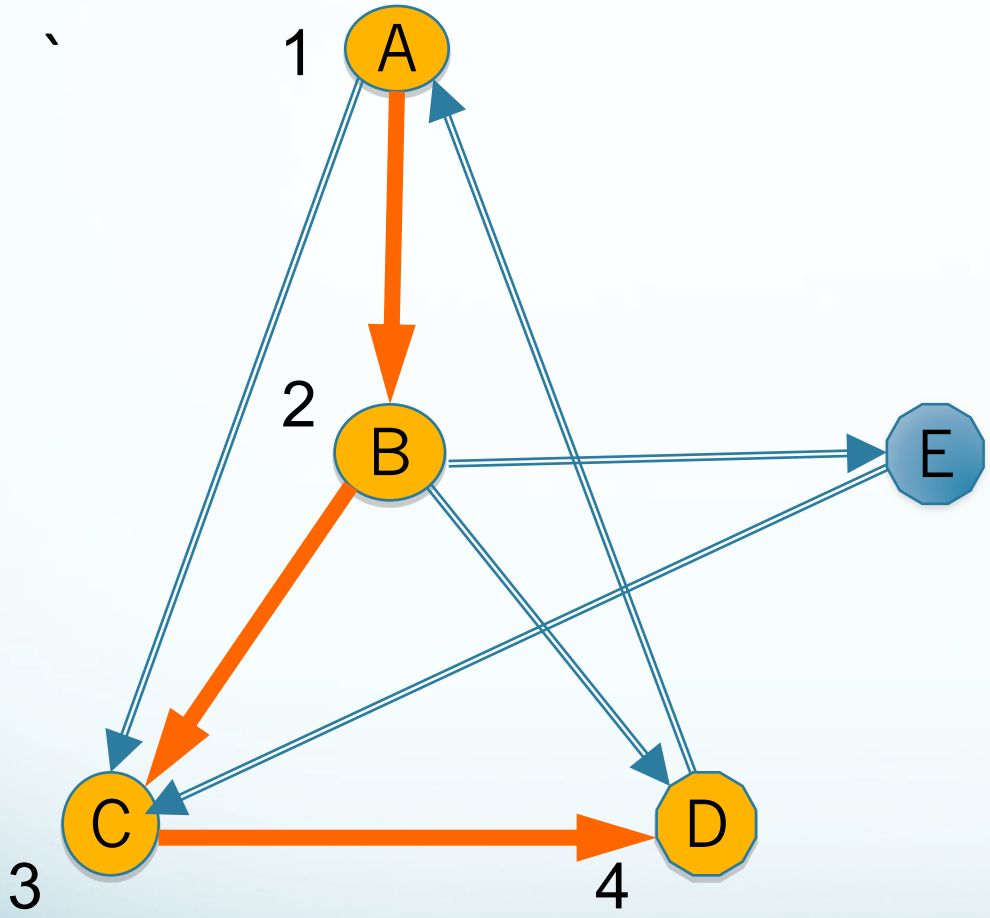
DFS



From B, visit C.

Stack:
$ABC

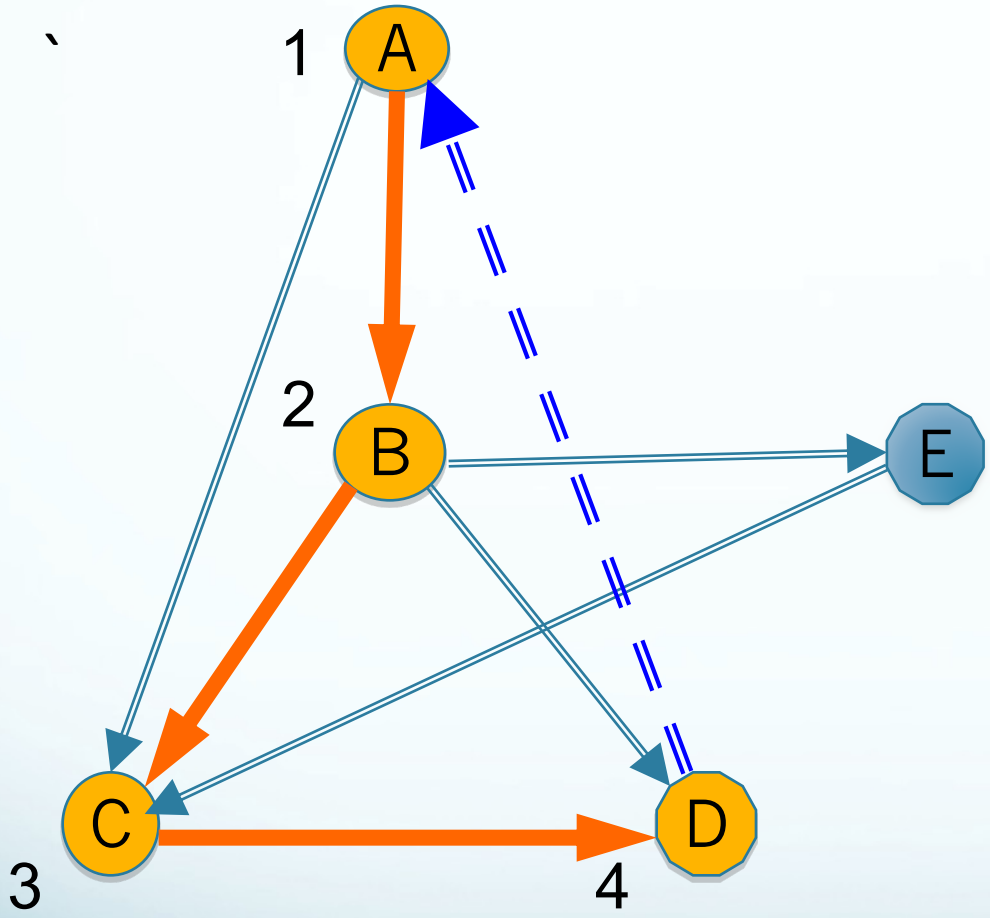
DFS



From C, visit D.

Stack:
$ABCD

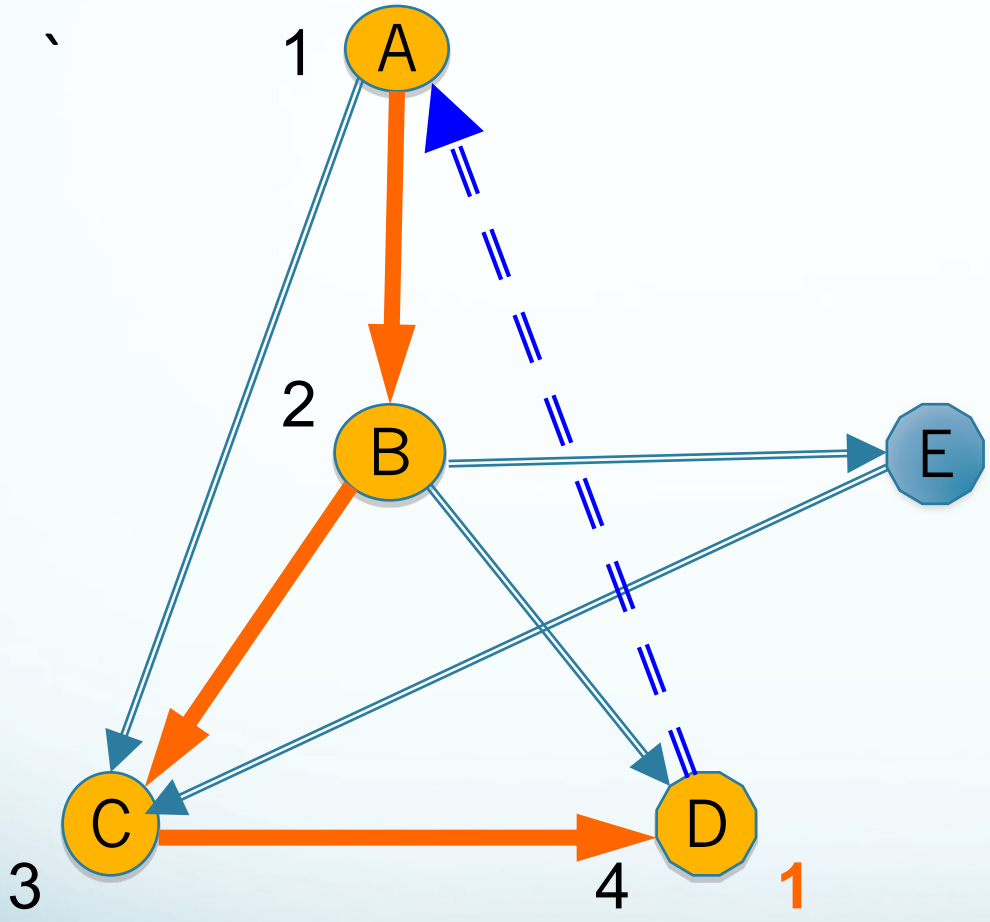
DFS



From D: examine edge D→A. A has already marked as visited, so no more visit for A. The blue dashed edge D→A is not a tree edge, it's a back edge. Note: A is currently in the stack.

Stack:
$ABCD

DFS

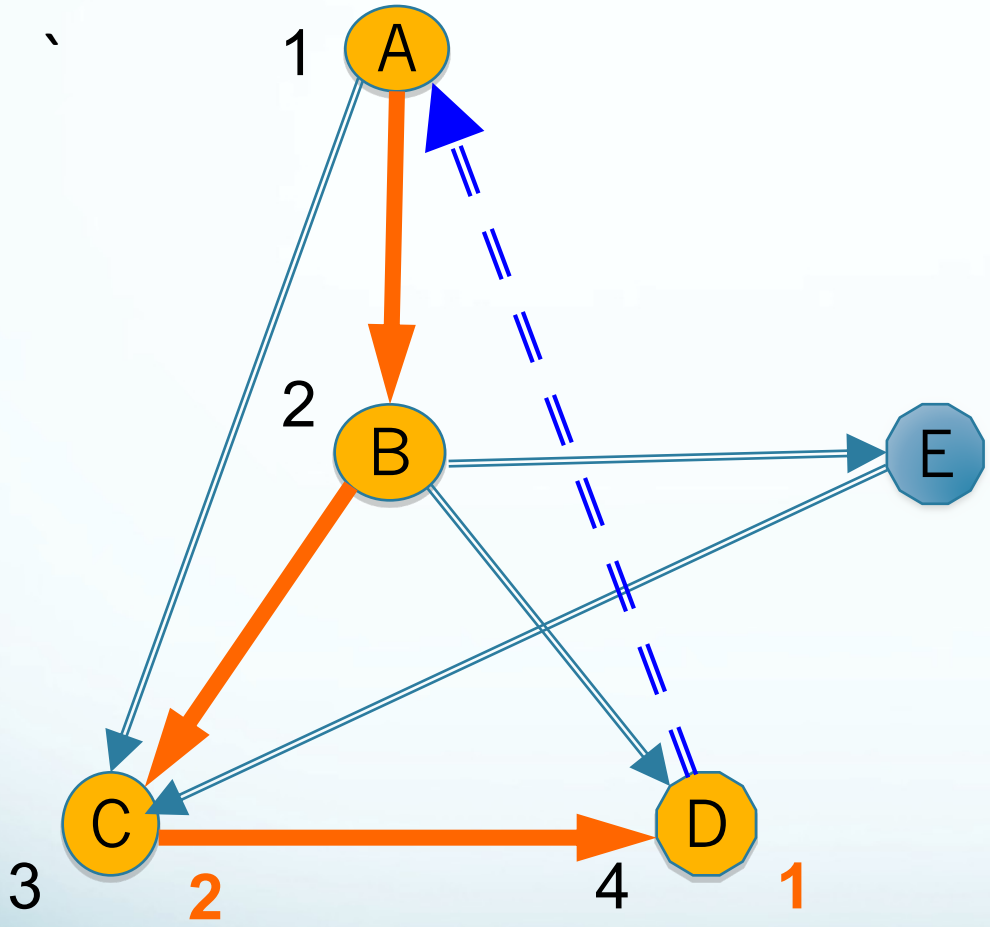


No more un-examined edge from D. Now the DFS backtracks from D to C. In the stack, it means that D is popped out, and C becomes the currently being visited node.

Stack:
$ABC

- note: orange number 1 represents the pop order

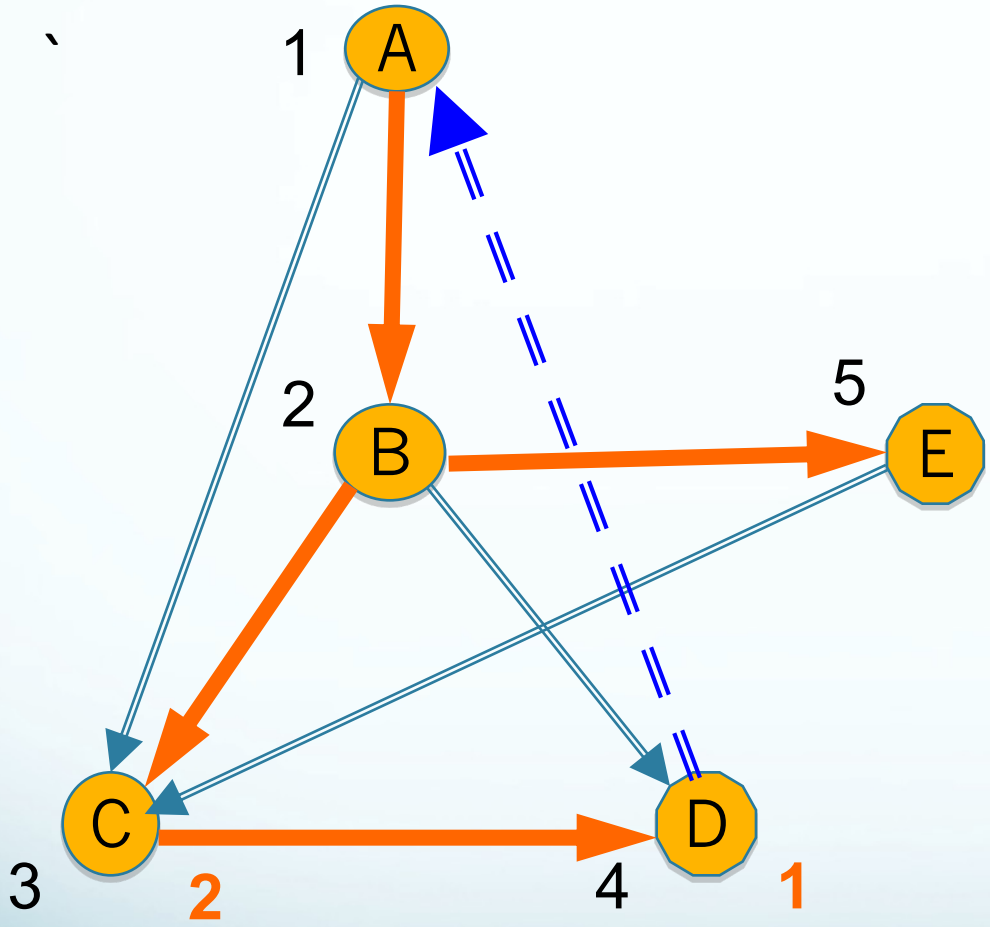
DFS



Continue with backtracking from C to B.

Stack:
$AB

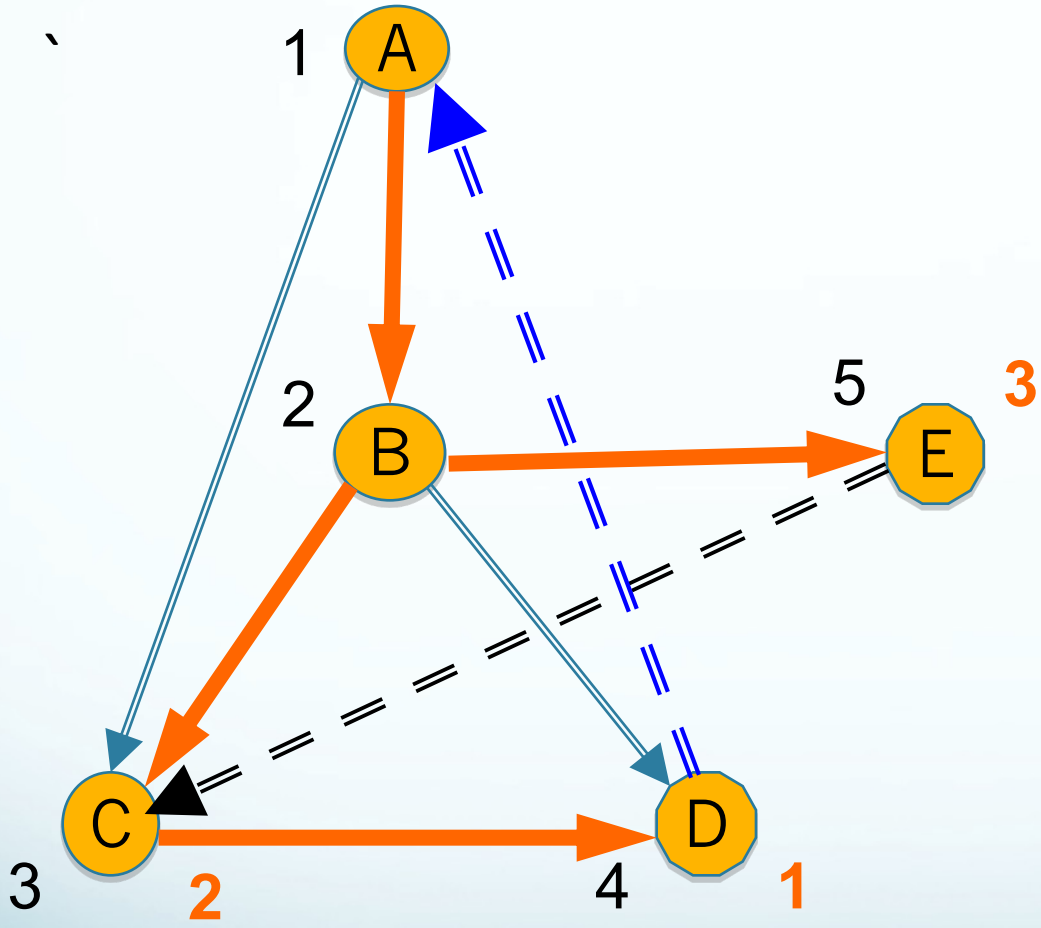
DFS



Now from B, there is an edge to unvisited node E, visits it.

Stack:
$ABE

DFS

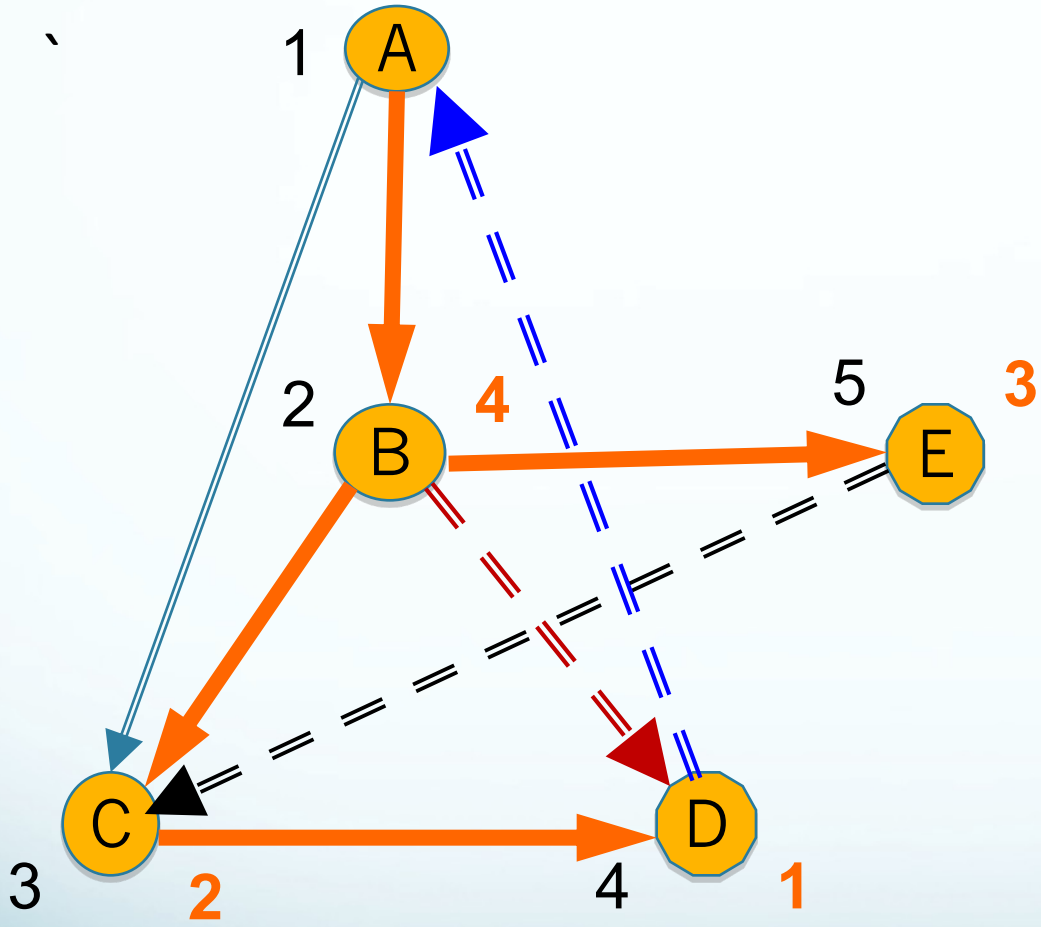


The black dashed edge is a cross edge. Note that C is not in the stack.

Continue with backtracking from E to B.

Stack:
$AB

DFS



The red dashed edge is a forward edge.

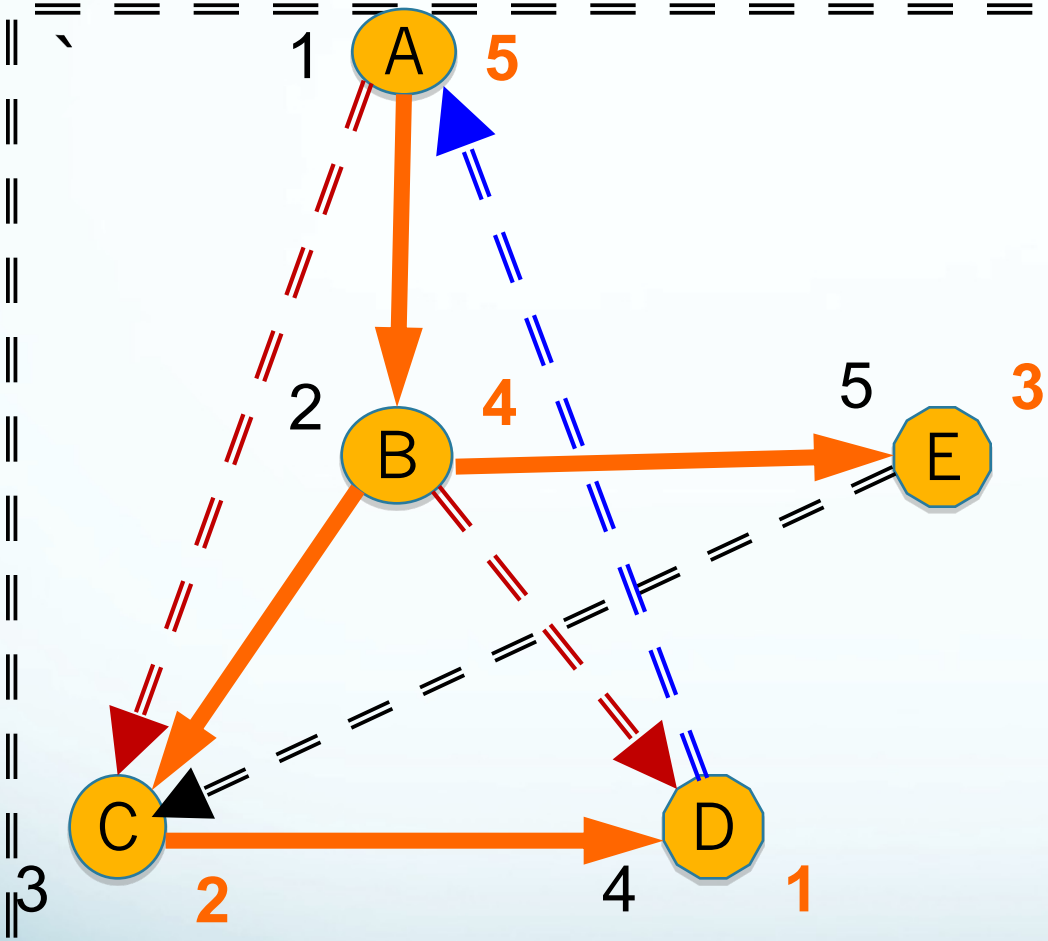
Then back from B to A

Stack:
$A

Then, stack become empty at the next step:
Stack:
$

and the BfsExplore(A) finished!

DFS

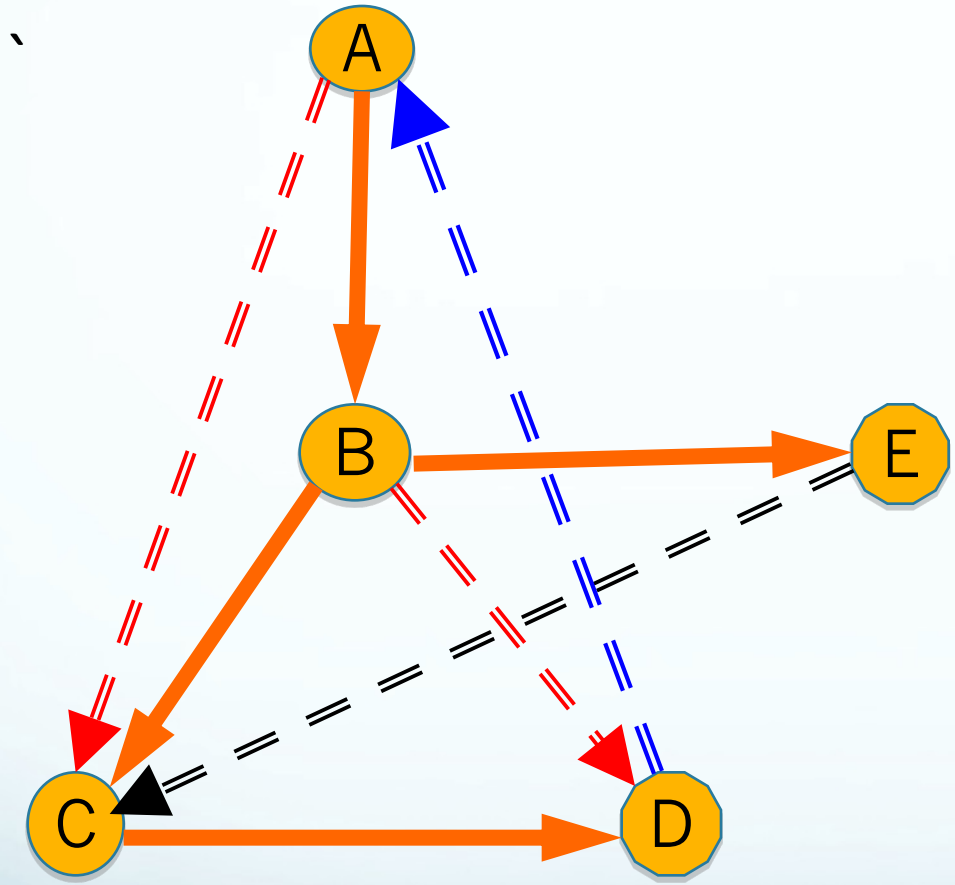


Solid orange edges are *tree edges*.

Non-tree edges:

- all would be *back-edges* if the graph was **un-directed**

DFS



Oranges edges are *tree edges*.

Non-tree edges:
- red are *forward edges*
- blue are *back-edges*
- other dashed are *cross edges*

Why don't we have forward and cross edges in undirected graph?