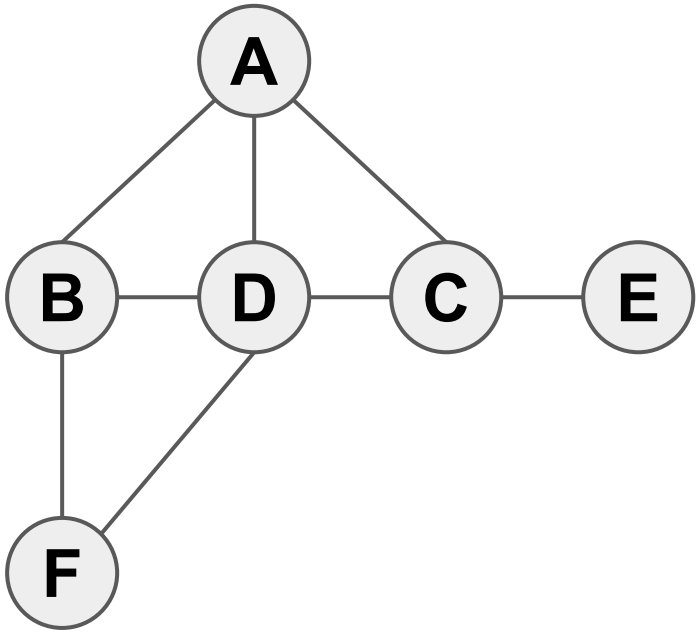
DFS Example

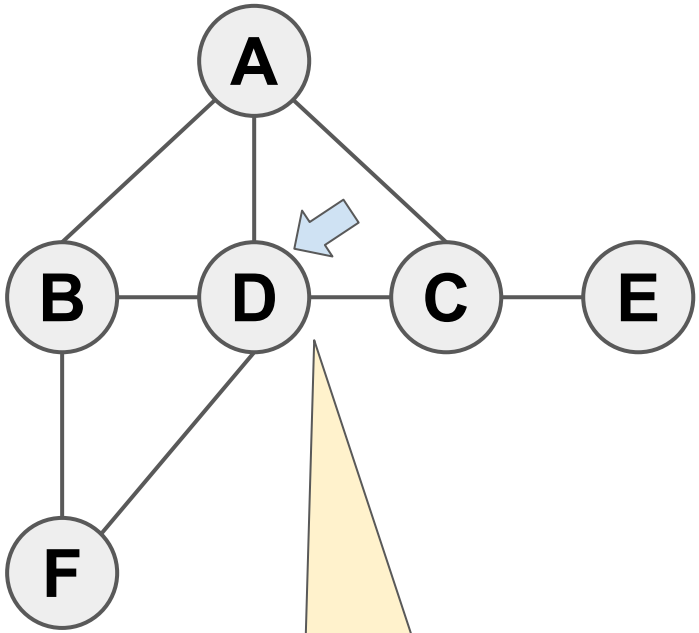


Visited

Stack

To do a depth-first search, we will need to keep track of the list of vertices we have visited already.

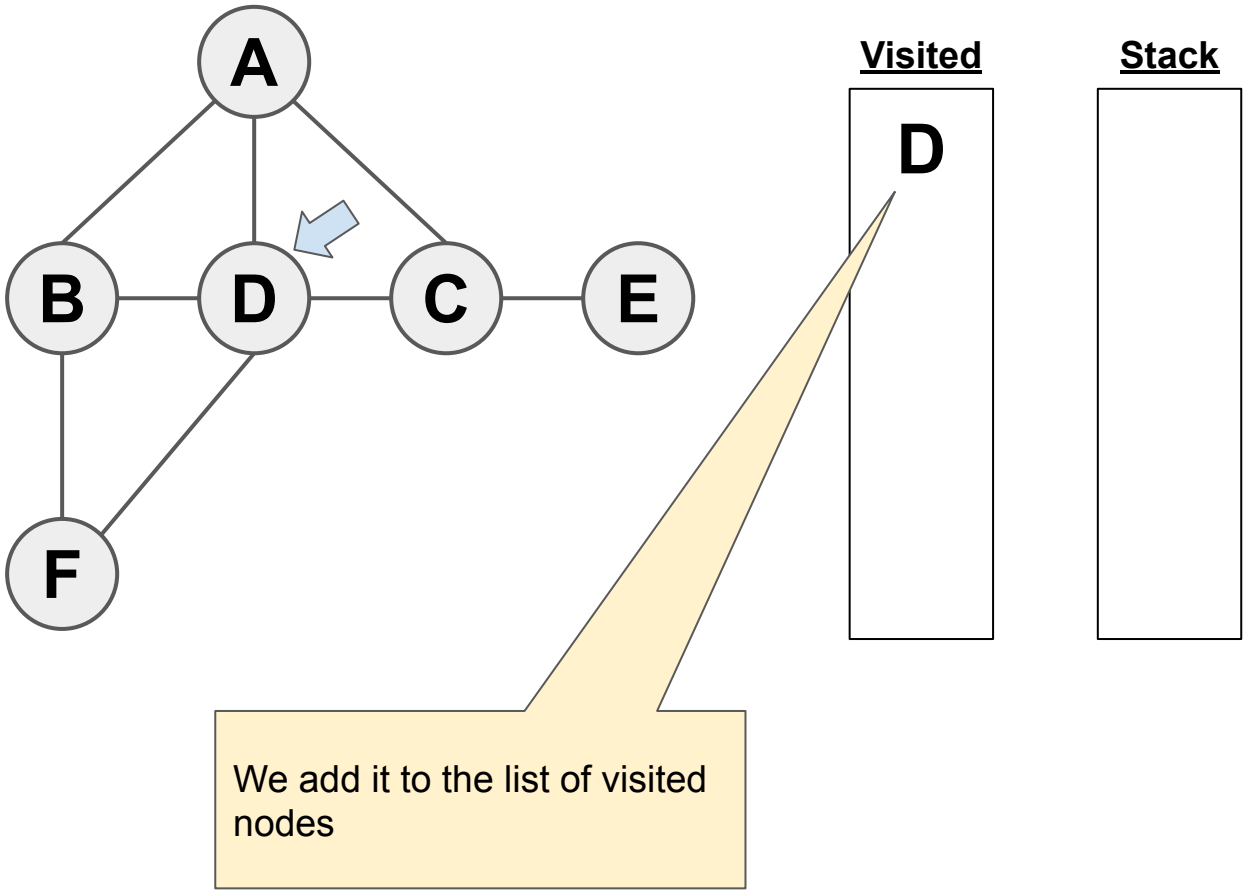
And a stack to keep track of the nodes we are still processing (and whose neighbors we may need to process further). This will also correspond to our recursive calls in DFS

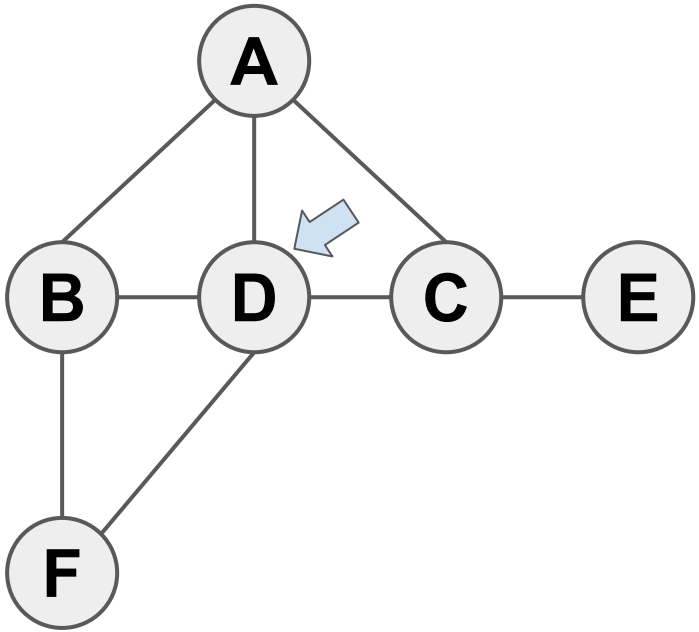


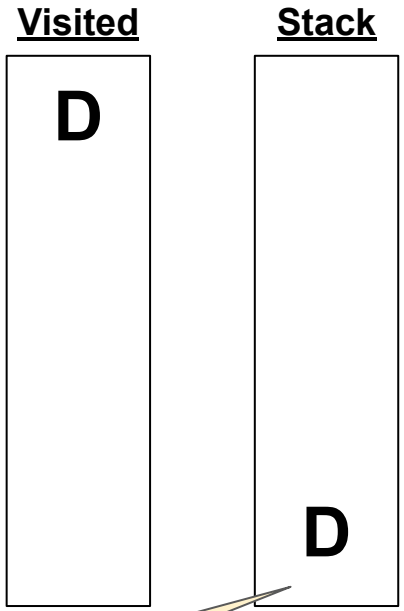
Our start node will be D (the start node is an input to the algorithm)

Visited

Stack

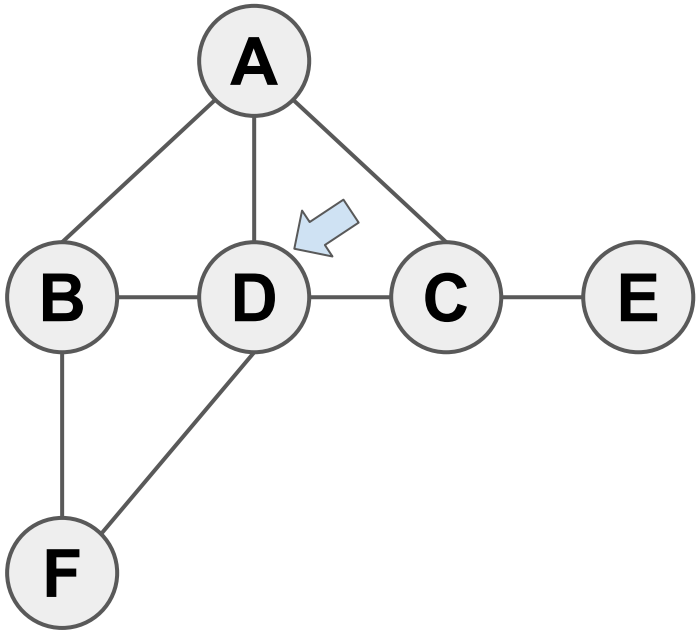

A
B
D
C
E
F
Visited
Stack
D
We add it to the list of visited nodes





A, B, C, F

And we keep track of the fact that we're processing D, and will be processing its neighbors.



Visited

D

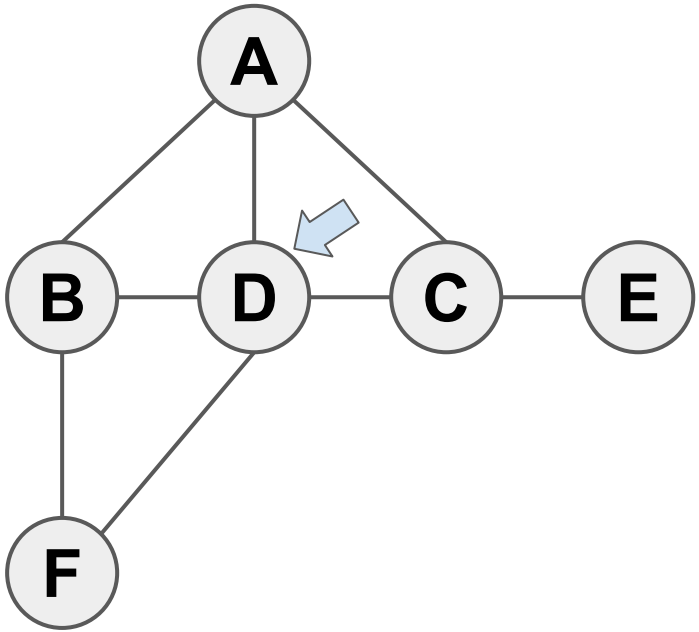
Stack

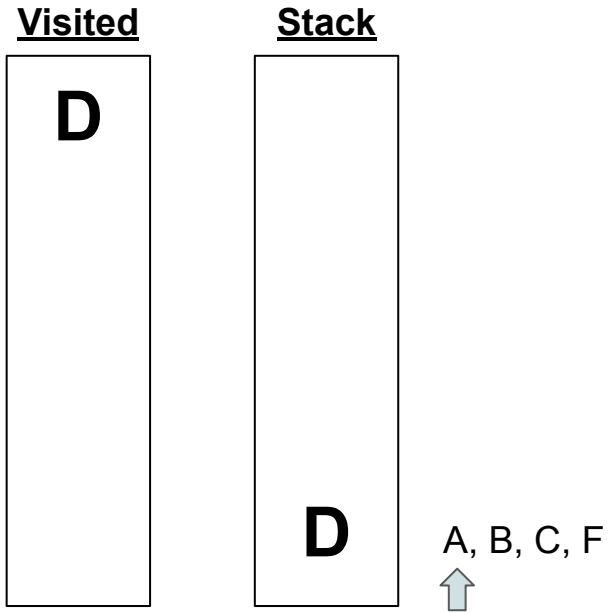
D

A, B, C, F

While we can use an actual stack data structure, our recursive implementation won't need one.

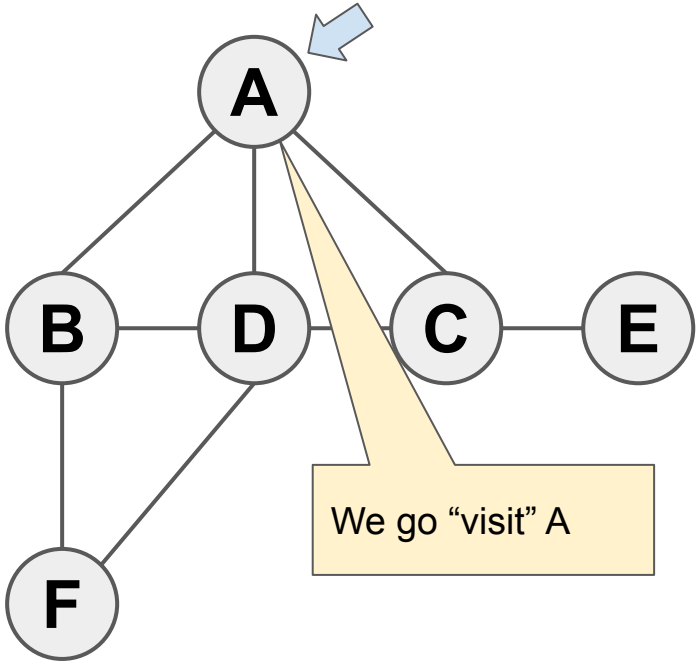
And we keep track of the fact that we're processing D, and will be processing its neighbors.

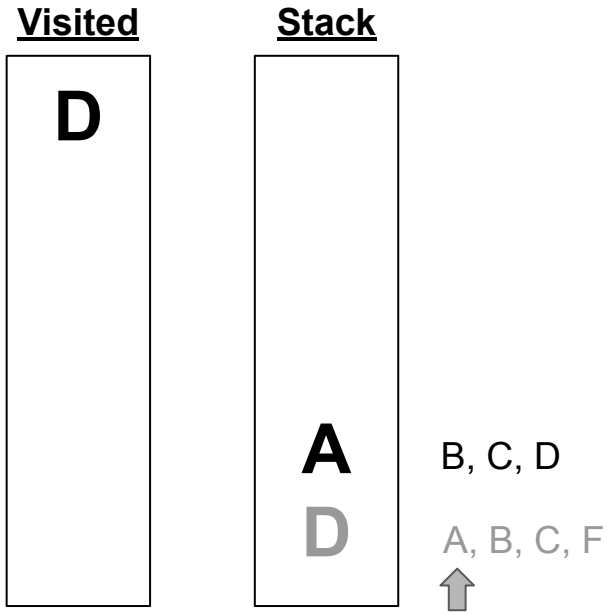


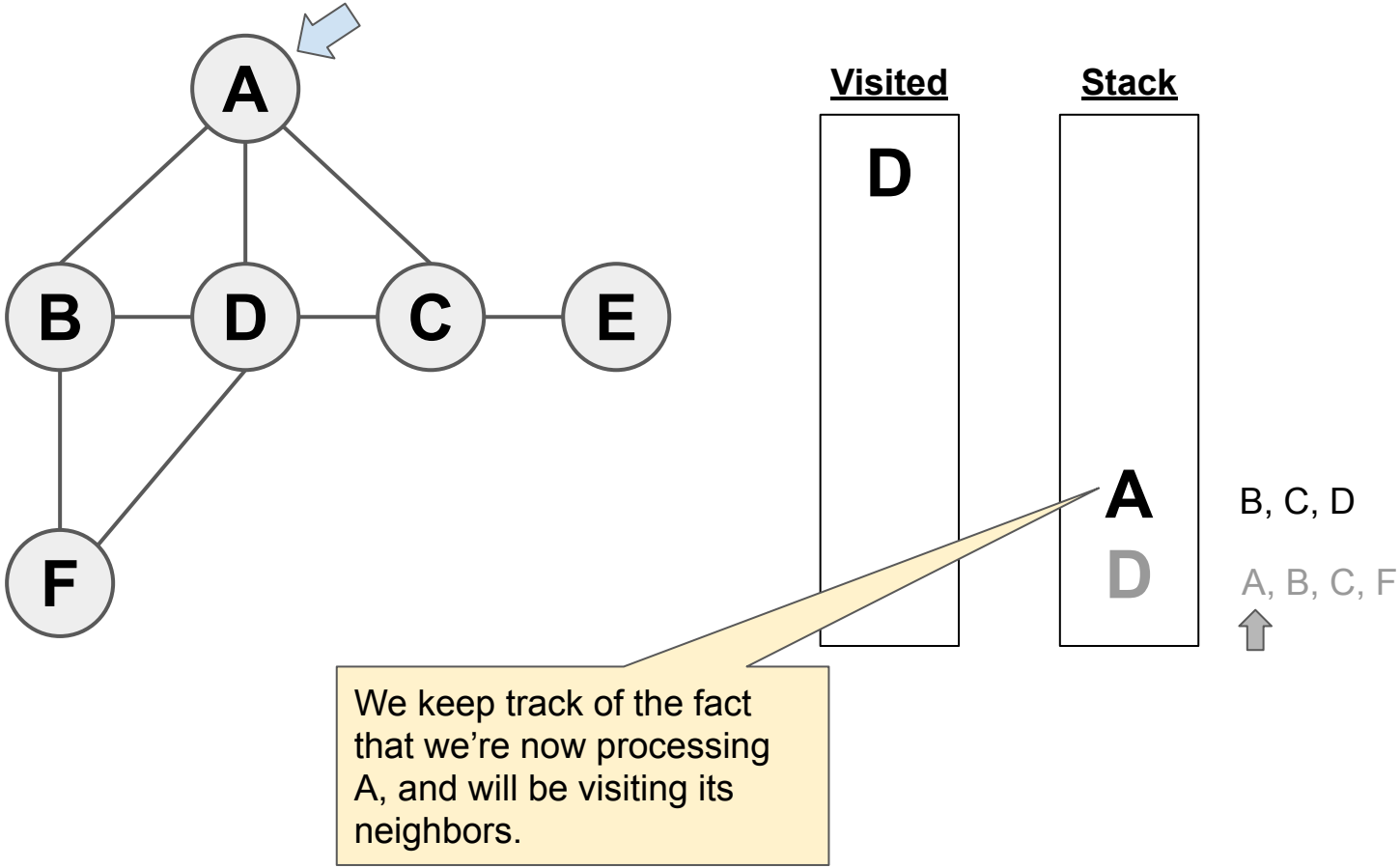


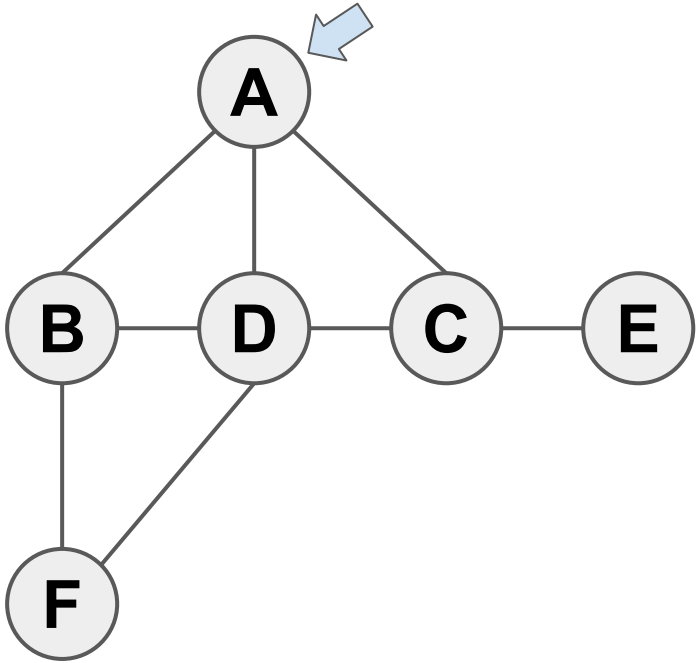
A, B, C, F

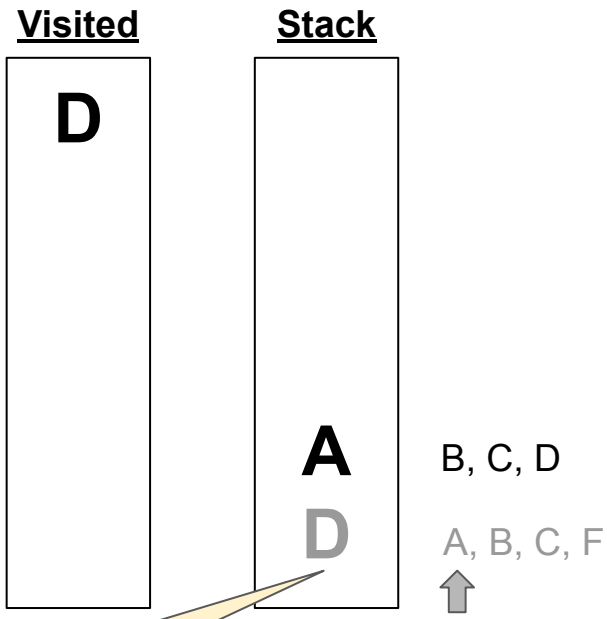
We start iterating over D's neighbors (let's assume we do this in alphabetical order). We start with A, which we haven't visited so…


A
B
D
C
E
F
We go “visit” A

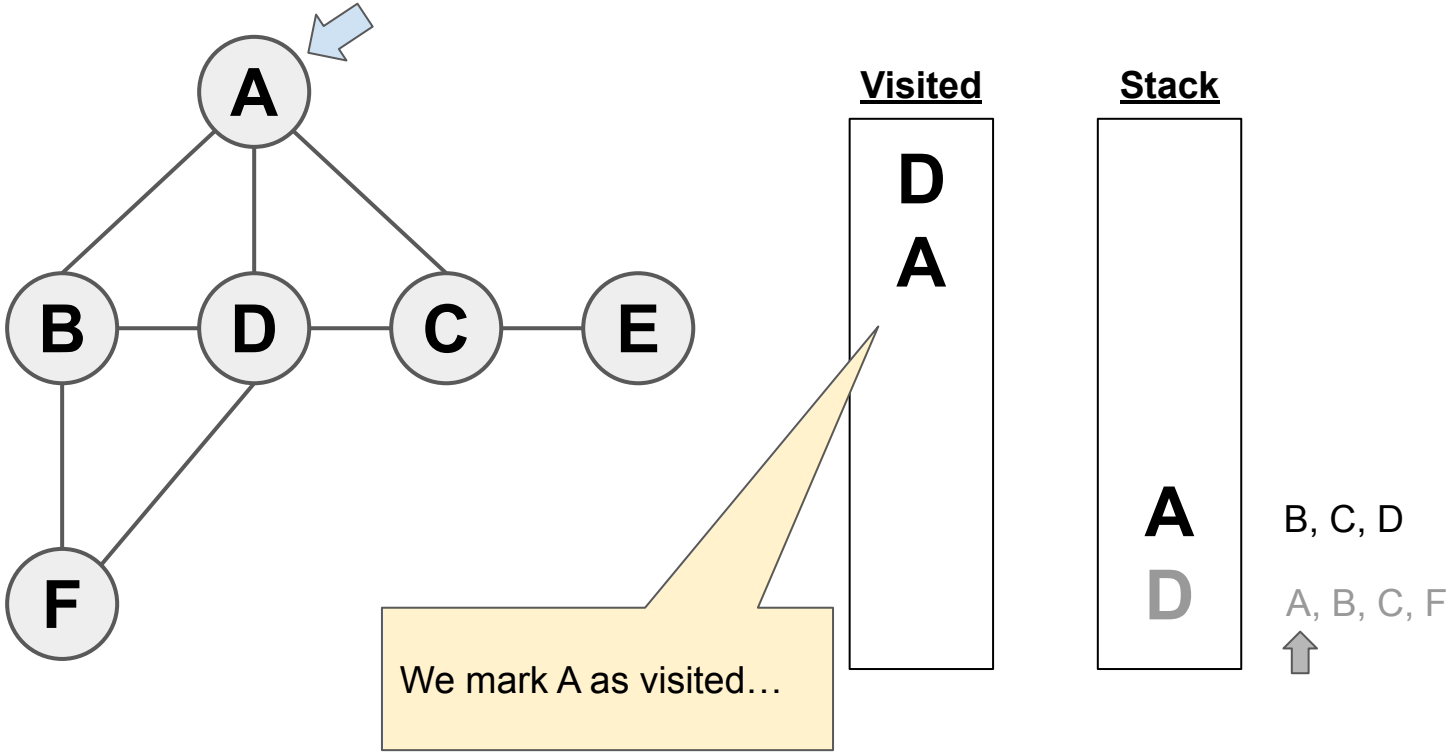

Visited
Stack
D
A
D
B, C, D
A, B, C, F

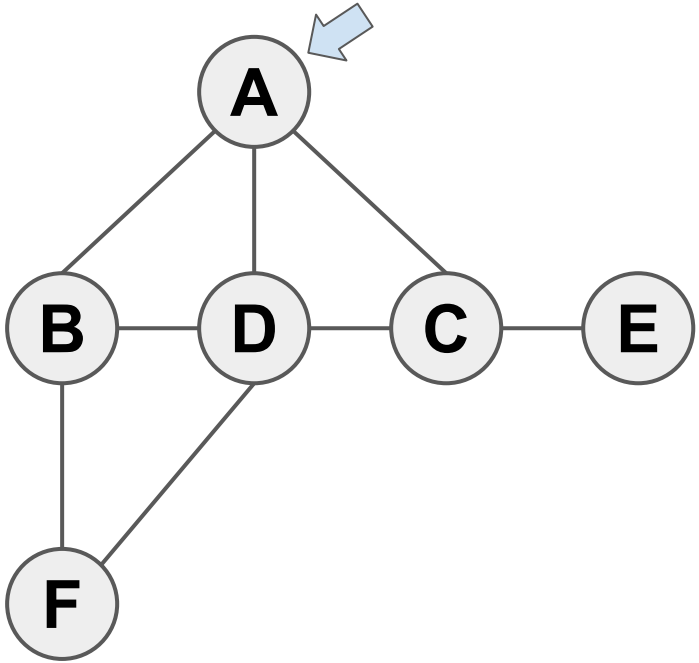

A
B
D
C
E
F
Visited
Stack
D
A
D
B, C, D
A, B, C, F
We keep track of the fact that we're now processing A, and will be visiting its neighbors.

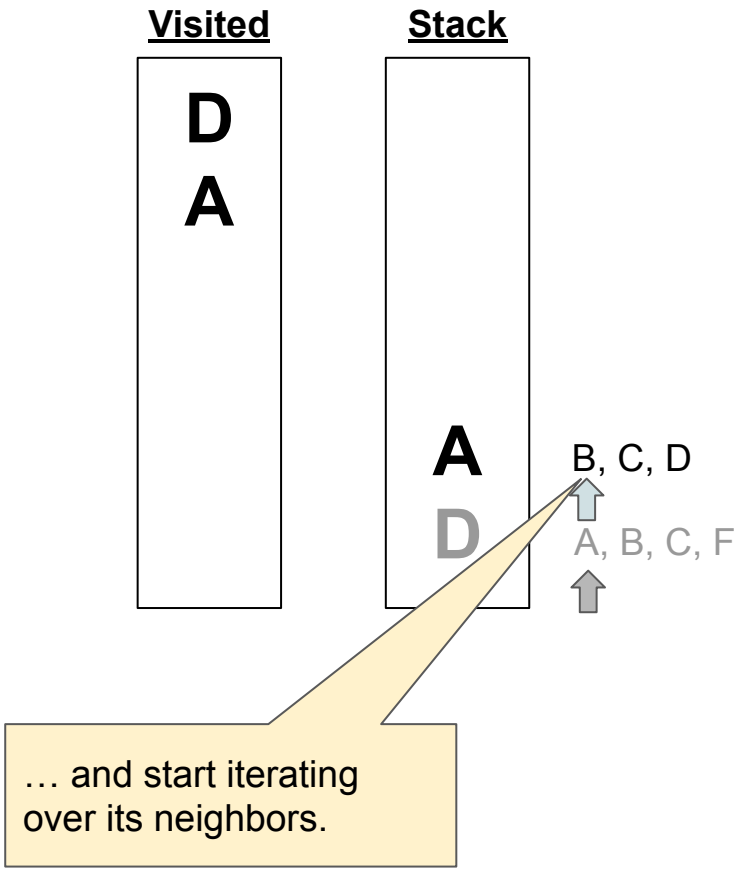


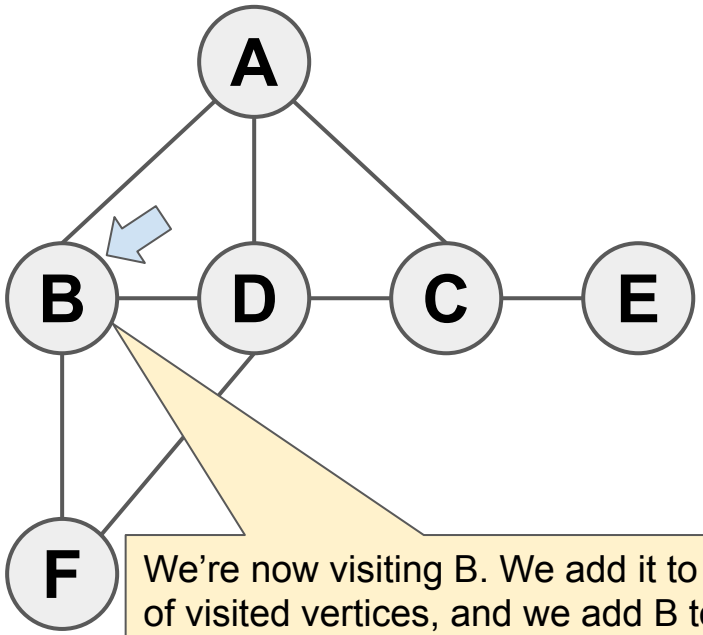


Notice how we've "paused" the processing of D (and its neighbors)


A
B
D
C
E
F
Visited
D
A
Stack
A
D
B, C, D
A, B, C, F
We mark A as visited…


A
B
D
C
E
F


Visited
Stack
D
A
A
D
B, C, D
A, B, C, F
… and start iterating
over its neighbors.



We're now visiting B. We add it to the list of visited vertices, and we add B to the stack. Like before, we've "paused" the processing of A, but we use the stack to keep track that we were processing A before we got to B (and D before we got to A)

Visited

D
A
B

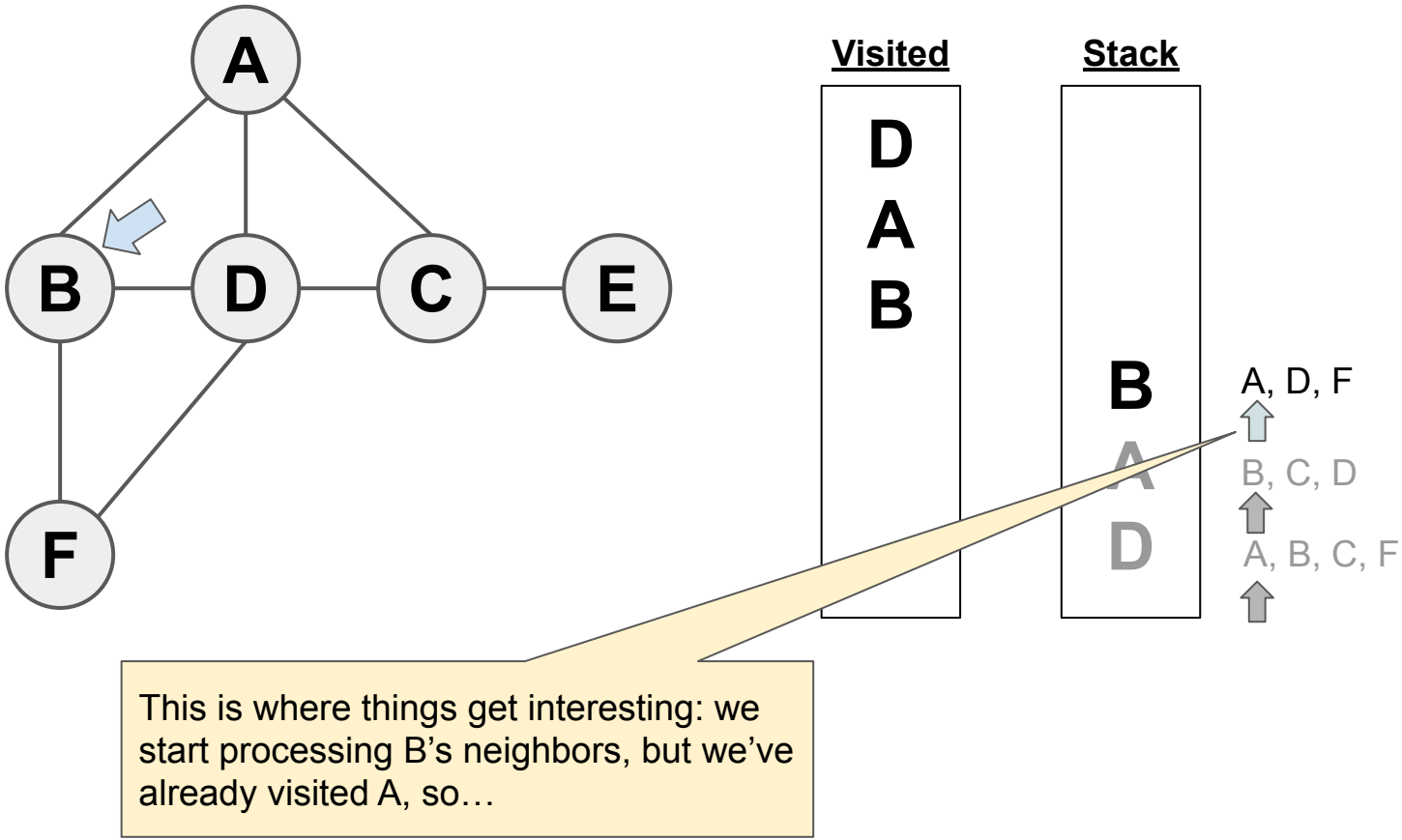
Stack

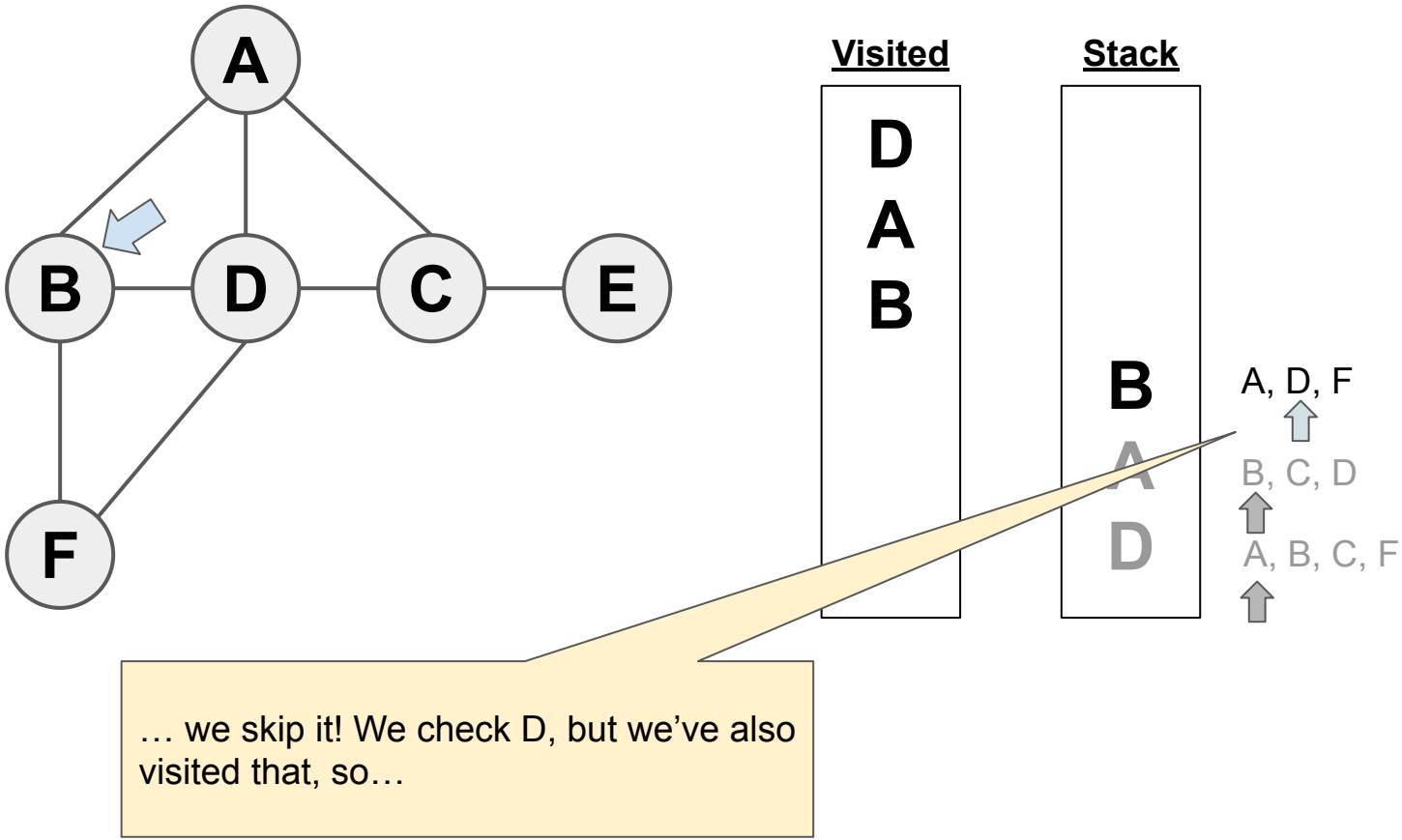
B
A
D

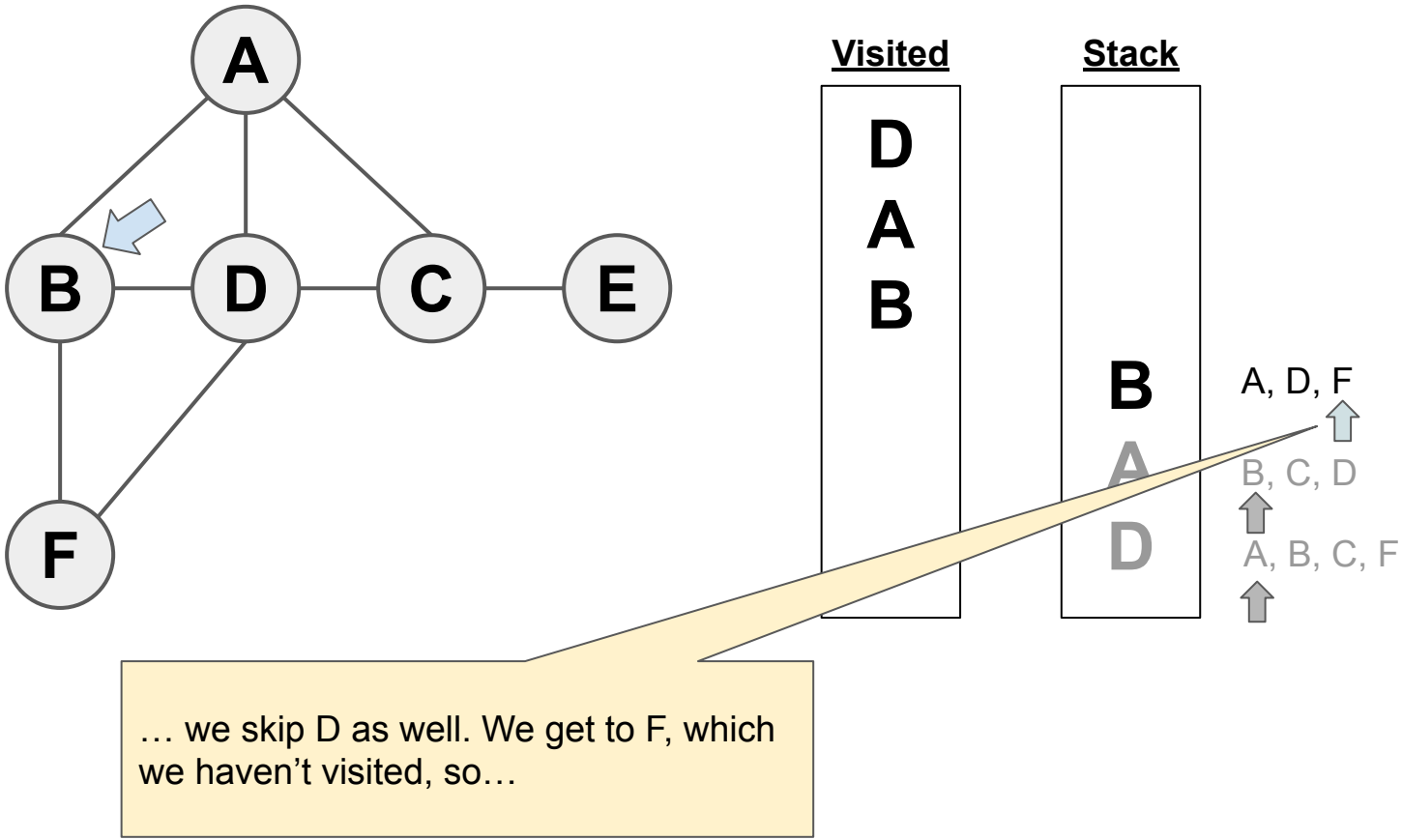
A, D, F

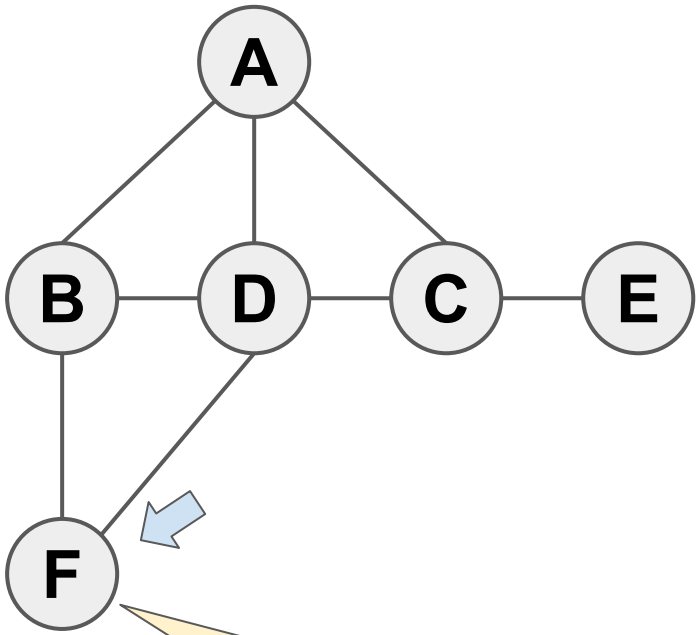
B, C, D

A, B, C, F


A
B
D
C
E
F
Visited
D
A
B
Stack
B
A
D
A, D, F
B, C, D
A, B, C, F
This is where things get interesting: we start processing B's neighbors, but we've already visited A, so…


A
B
D
C
E
F
Visited
D
A
B
Stack
B
A
D
A, D, F
B, C, D
A, B, C, F
… we skip it! We check D, but we've also visited that, so…


A
B
D
C
E
F
Visited
D
A
B
Stack
B
A
D
A, D, F
B, C, D
A, B, C, F
… we skip D as well. We get to F, which we haven't visited, so…



… we go visit F (and we update the visited list and stack accordingly)

Visited

D
A
B
F

Stack

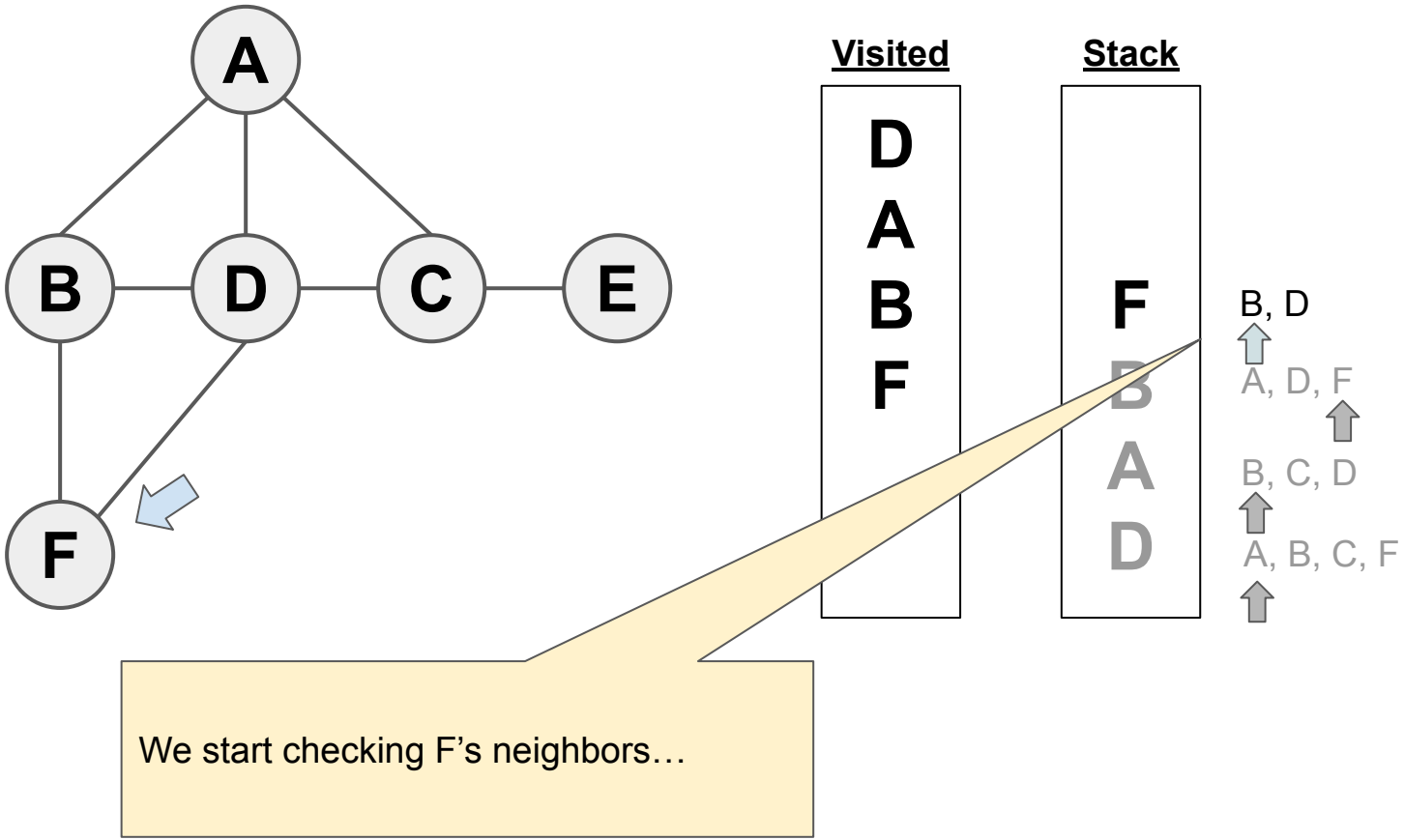
F
B
A
D

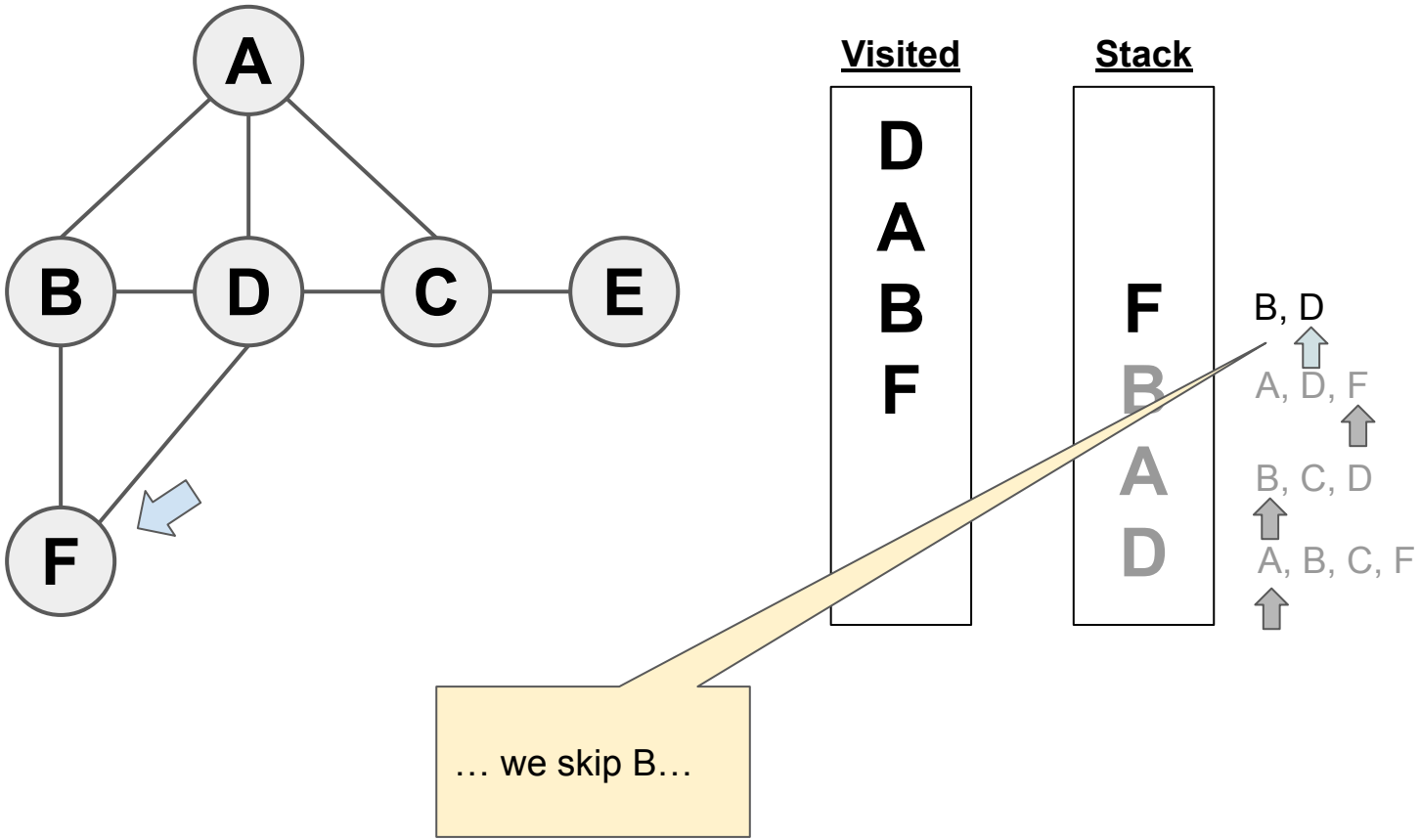
B, D

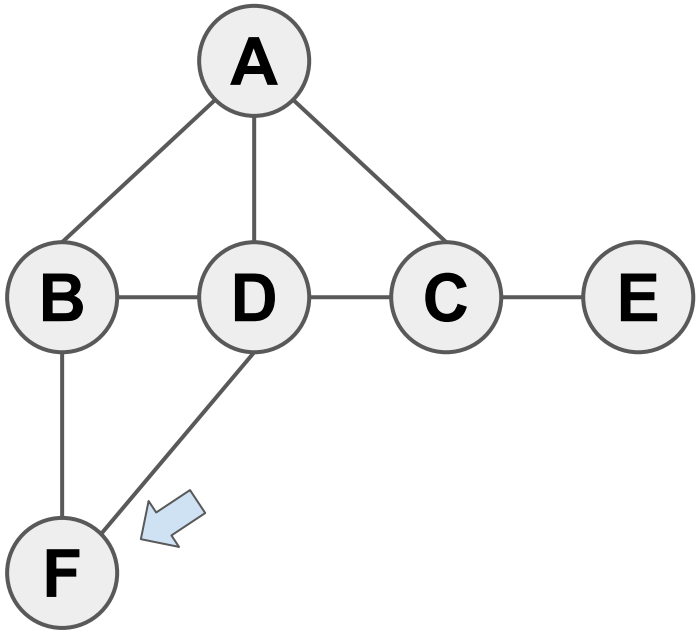
A, D, F

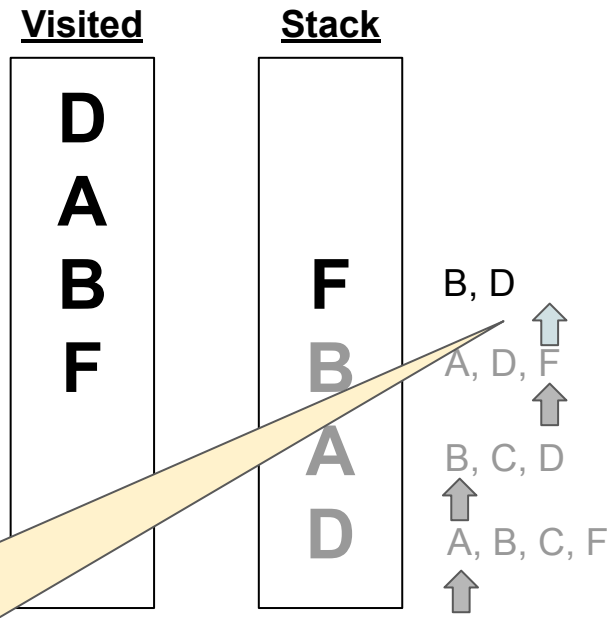
B, C, D

A, B, C, F

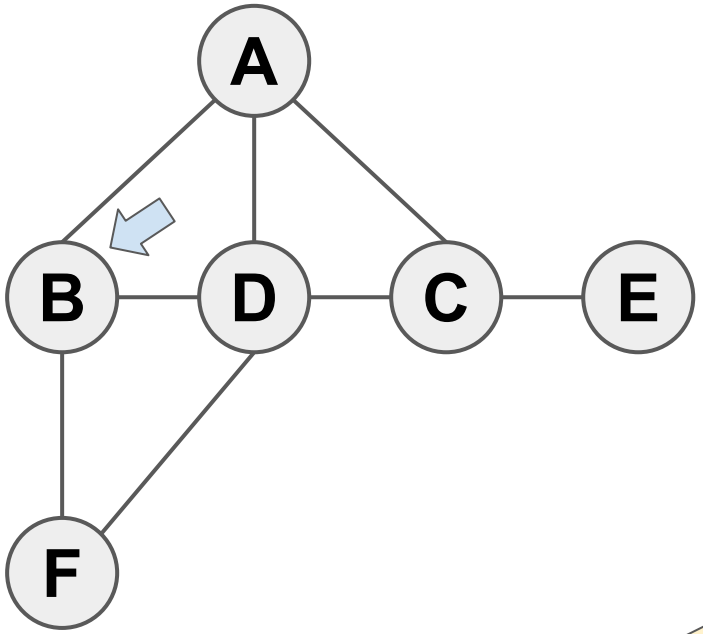

A
B
D
C
E
F
Visited
D
A
B
F
Stack
F
B
A
D
B, D
A, D, F
B, C, D
A, B, C, F
We start checking F’s neighbors…


A
B
D
C
E
F
Visited
Stack
D
A
B
F
F
B
A
D
B, D
A, D, F
B, C, D
A, B, C, F
… we skip B…





… as well as D. At this point, we've fully processed F: we've visited it, and we've gone through its list of neighbors (but we didn't go visit them, because they had previously been visited)



Visited

D
A
B
F

Stack

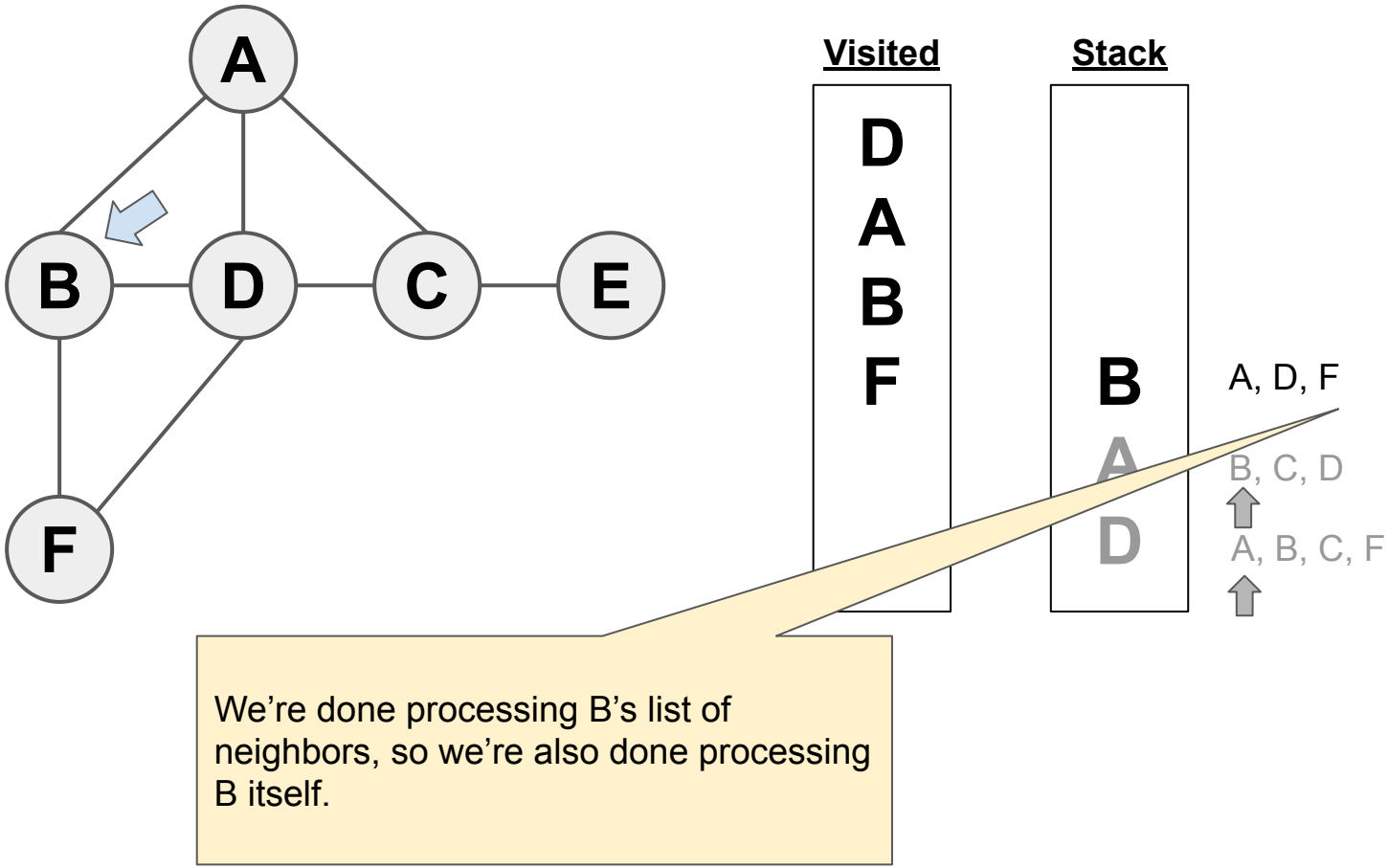
B
A
D

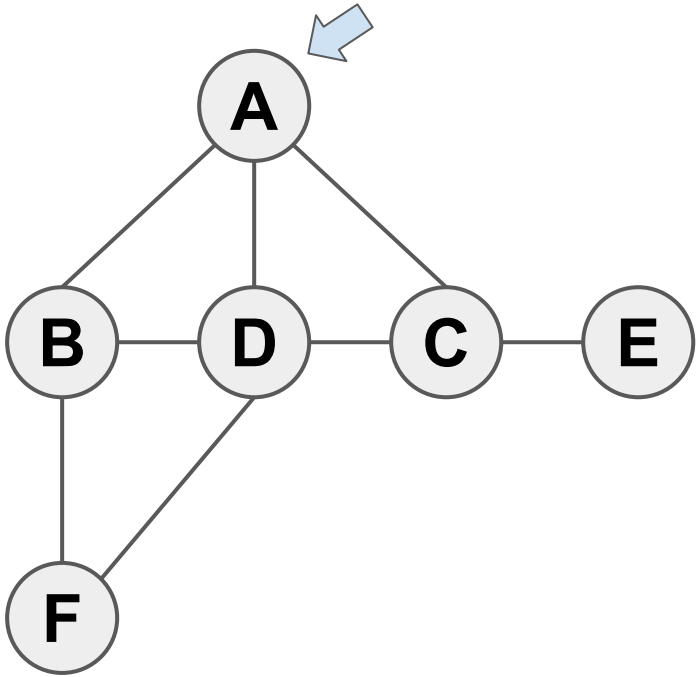
A, D, F

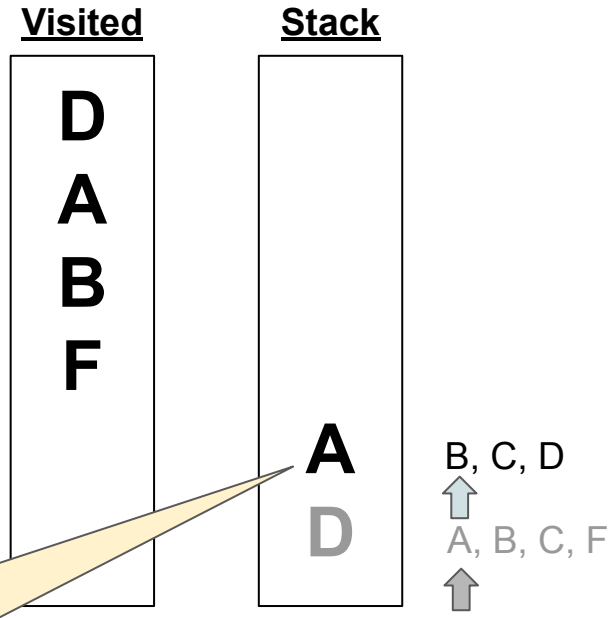
B, C, D

A, B, C, F

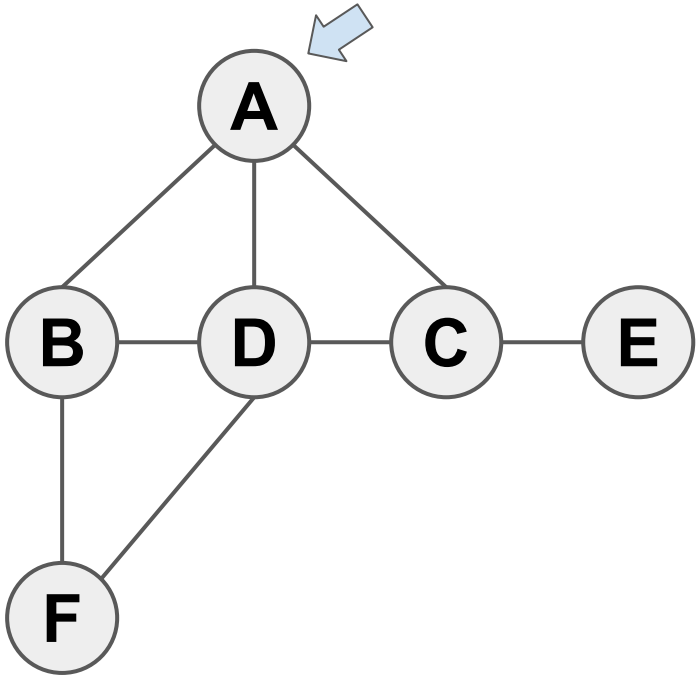
So we remove F's entry from the stack, and we're back to processing B.

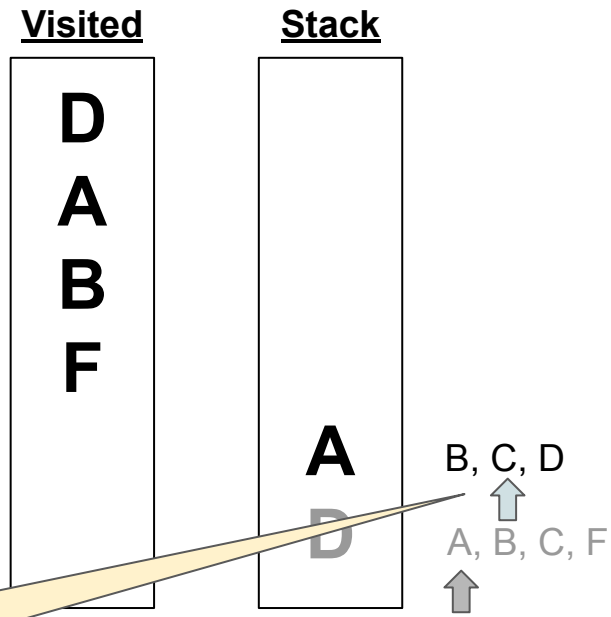

A
B
D
C
E
F
Visited
D
A
B
F
Stack
B
A
D
A, D, F
B, C, D
A, B, C, F
We're done processing B's list of neighbors, so we're also done processing B itself.



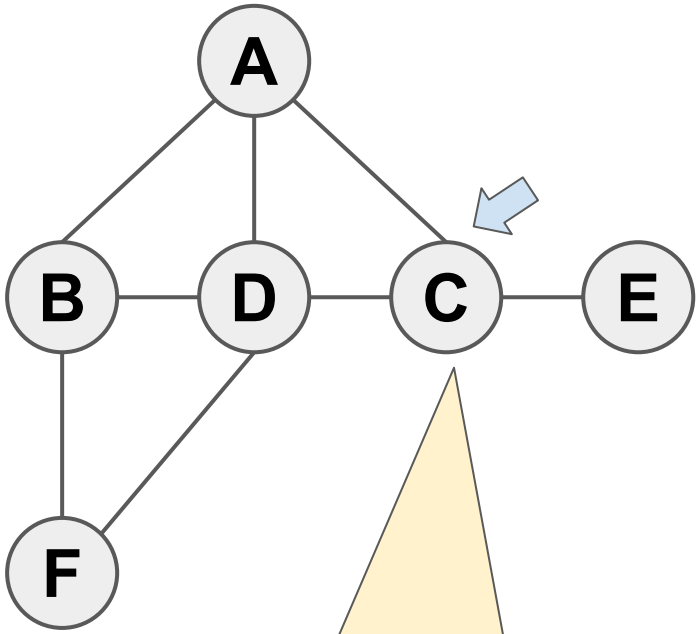


We remove B from the stack, and we're back to processing A. We resume where we left things off: we had gone to visit B…





… so we check the next neighbor, C. We haven't visited that vertex before, so…



Visited

D
A
B
F
C

Stack

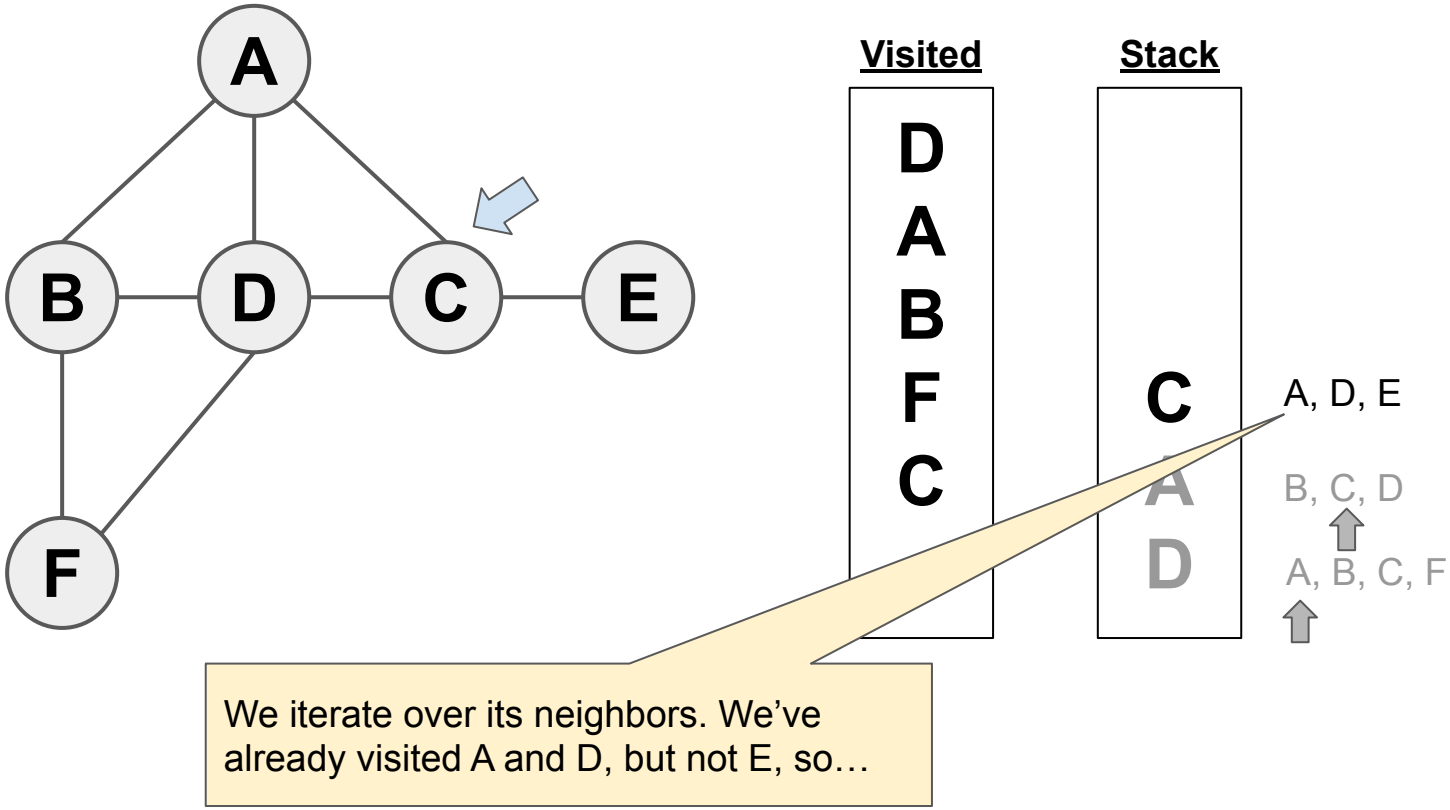
C
A
D

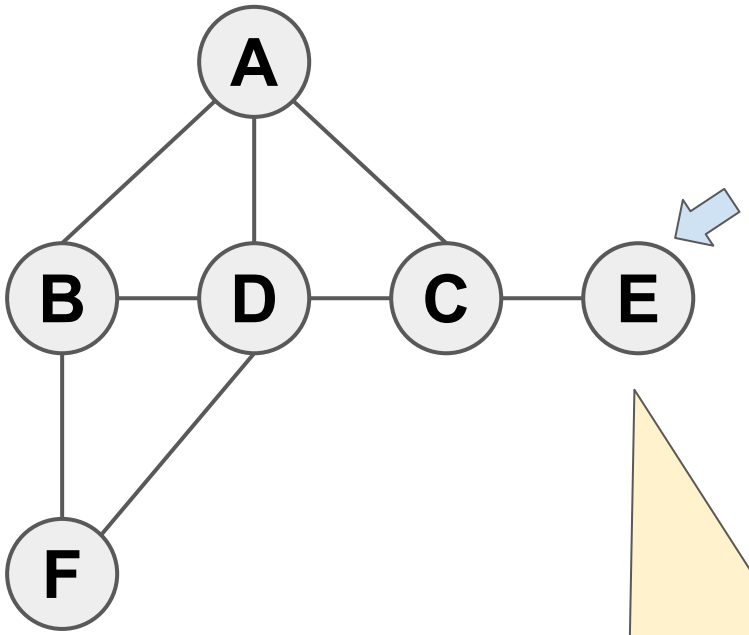
A, D, E

B, C, D

A, B, C, F

We visit C and update the visited list and stack.


A
B
D
C
E
F
Visited
Stack
D
A
B
F
C
C
A
D
A, D, E
B, C, D
A, B, C, F
We iterate over its neighbors. We've already visited A and D, but not E, so…



Visited

D
A
B
F
C
E

Stack

E
C
A
D

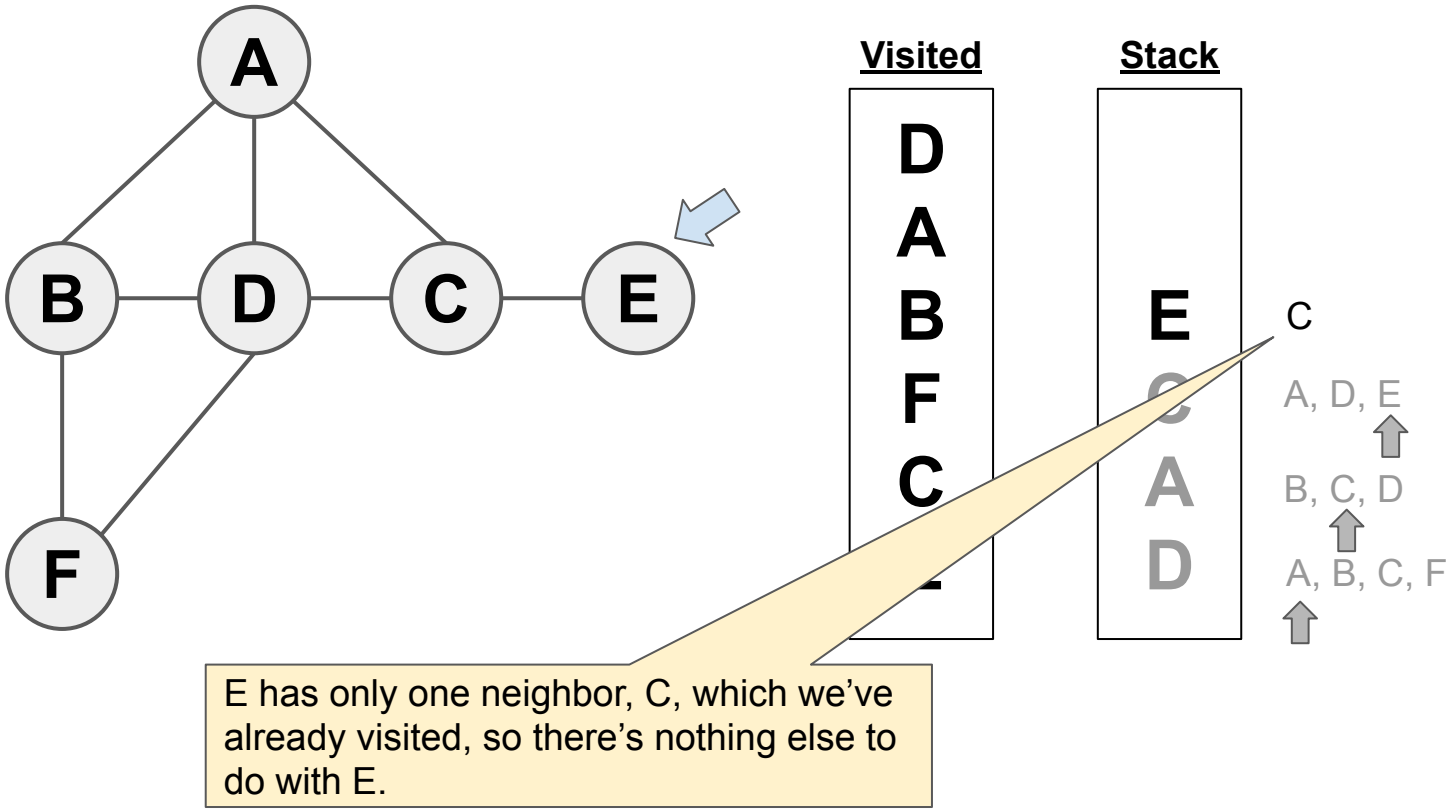
C

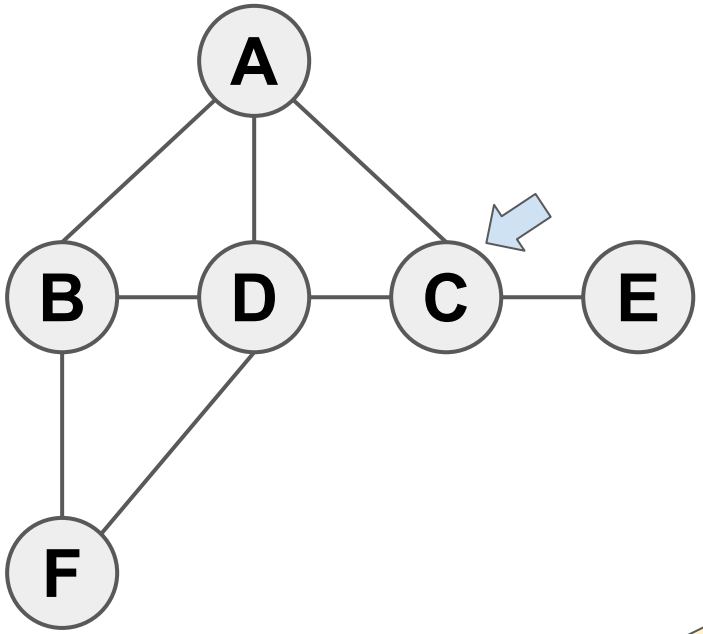
A, D, E

B, C, D

A, B, C, F

We go visit E, and update the list of visited lists and the stack.


A
B
D
C
E
F
Visited
D
A
B
F
C
Stack
E
C
A
D
C
A, D, E
B, C, D
A, B, C, F
E has only one neighbor, C, which we've already visited, so there's nothing else to do with E.



Visited

D
A
B
F
C

Stack

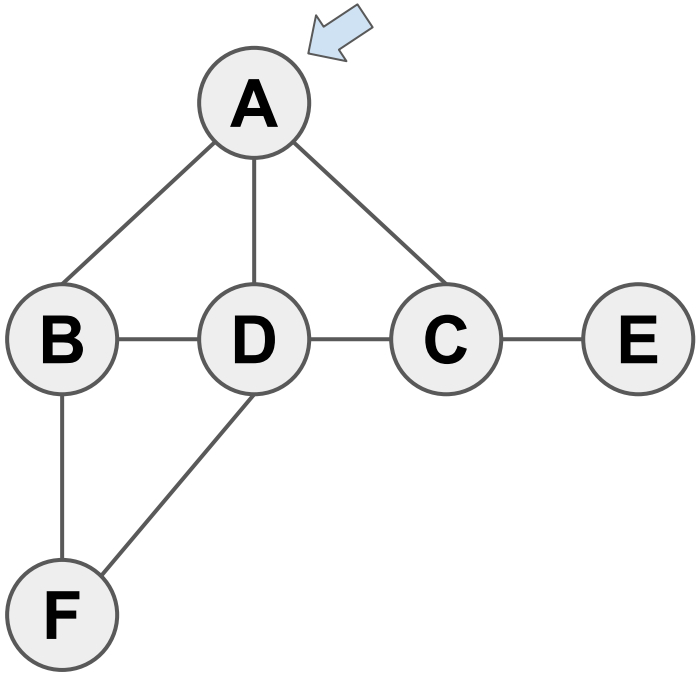
C
A
D

A, D, E

B, C, D

A, B, C, F

We're back to processing C, but we're also done with its list of neighbors, so we're also done with C.



Visited

D
A
B
F
C
E

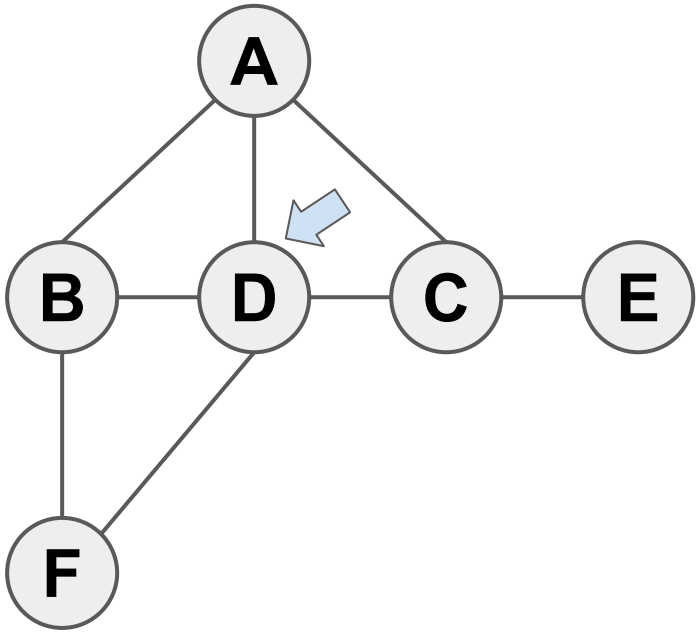
Stack

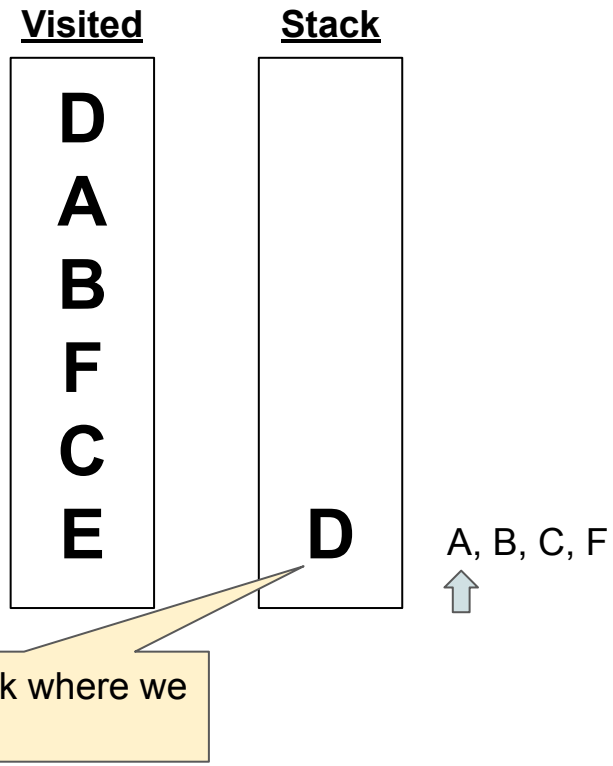
A
D

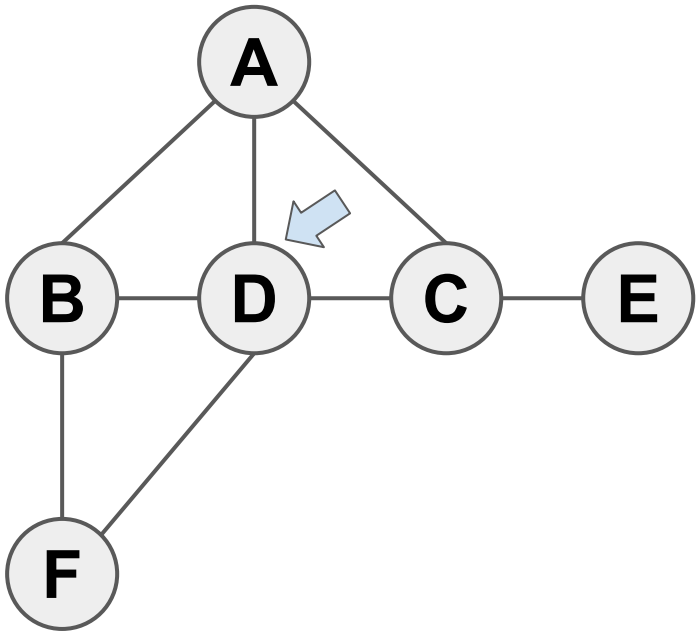
B, C, D

A, B, C, F

And we're back to processing A. We're still not done with its list of neighbors, but the remaining neighbor (D) has already been processed, so we're also done with A


A
B
D
C
E
F


Visited
Stack
D
A
B
F
C
E
D
A, B, C, F
So we're back where we started: D.



Visited

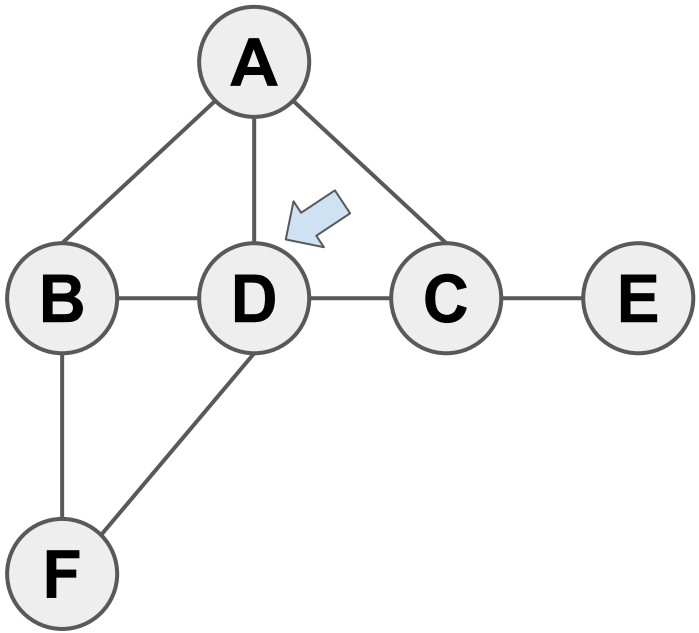
D
A
B
F
C
E

Stack

D

A, B, C, F

When we went off to visit A (then B, followed by F, C, and E), we still had three neighbors left to process: B, C, and F



Visited

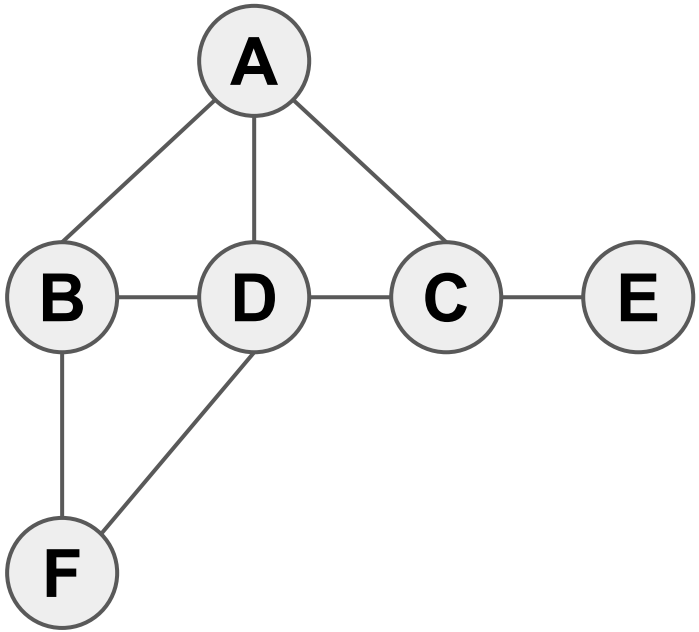
D
A
B
F
C
E

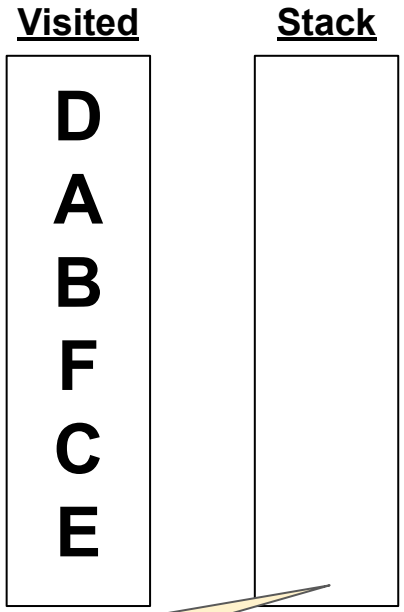
Stack

D

A, B, C, F

We've already visited those neighbors, so we skip them. There is nothing left to do with D…





And the stack is now empty, which means we are done with our depth-first traversal.