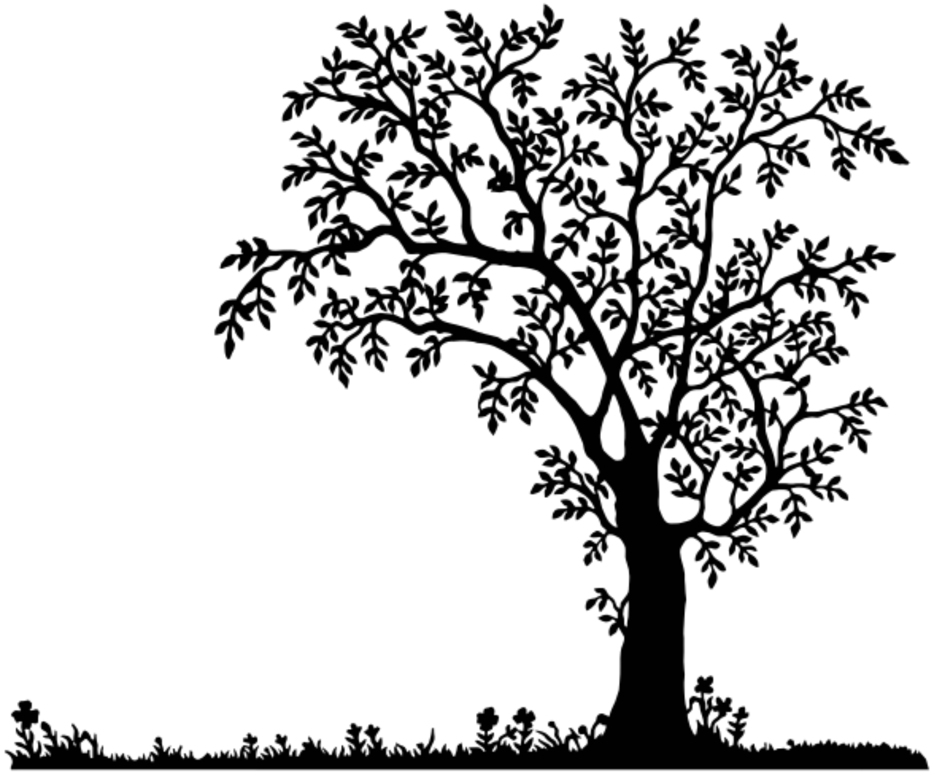
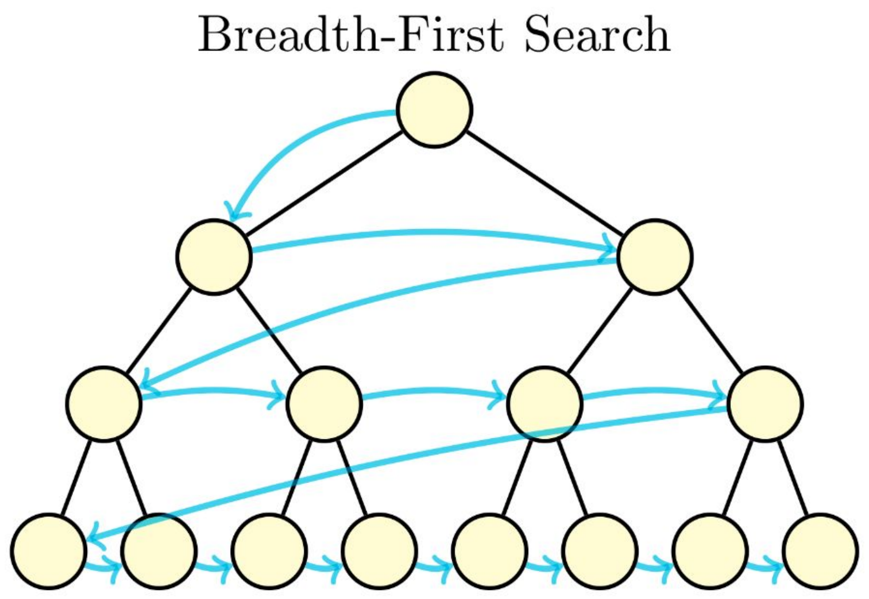
Data Structures and Algorithms (DSA) for AI

Fernanda Madeiral

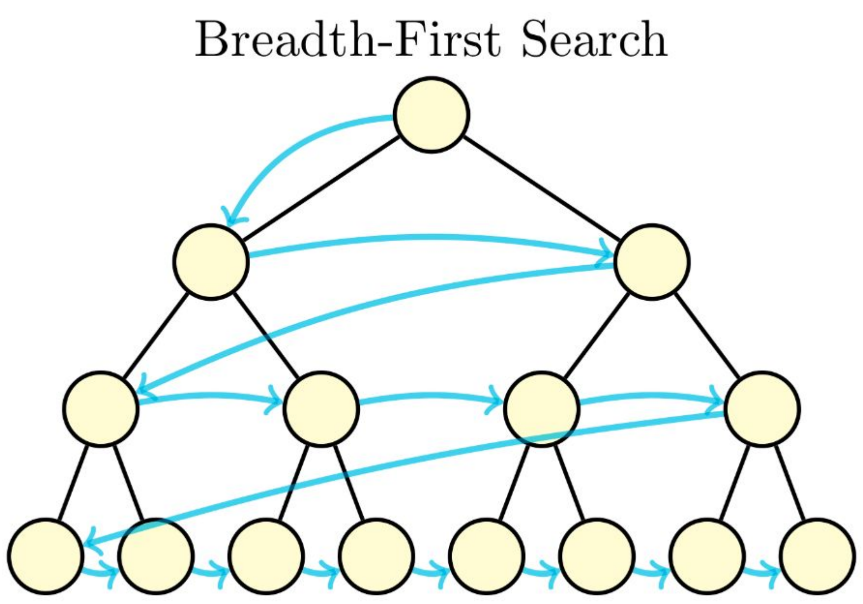


Breadth-first search (BFS)

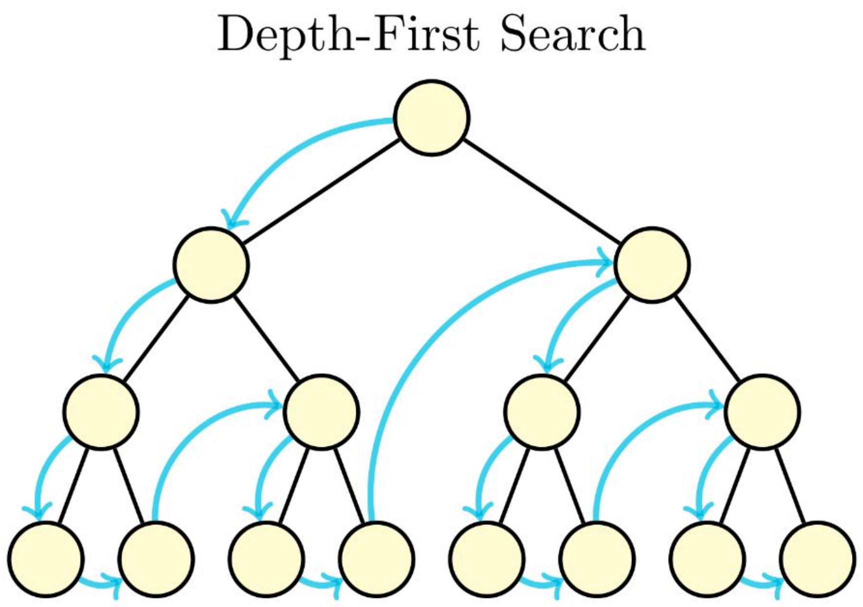


Queue

How to implement depth-first search (DFS)?

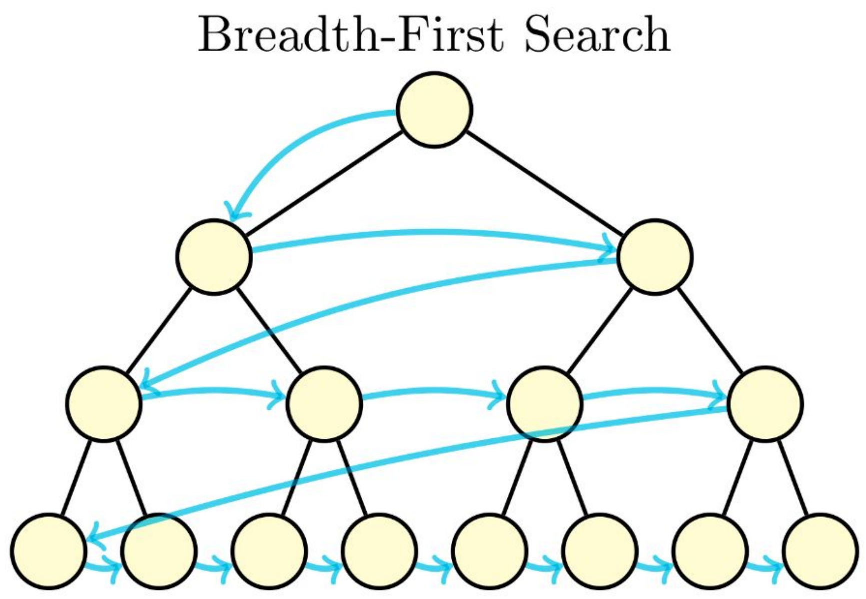


Queue

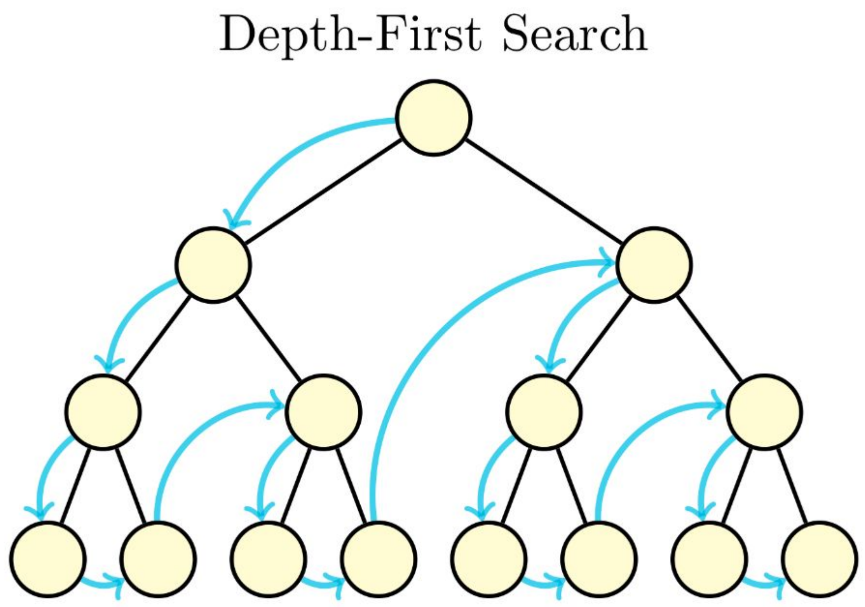


?

How to implement depth-first search (DFS)?

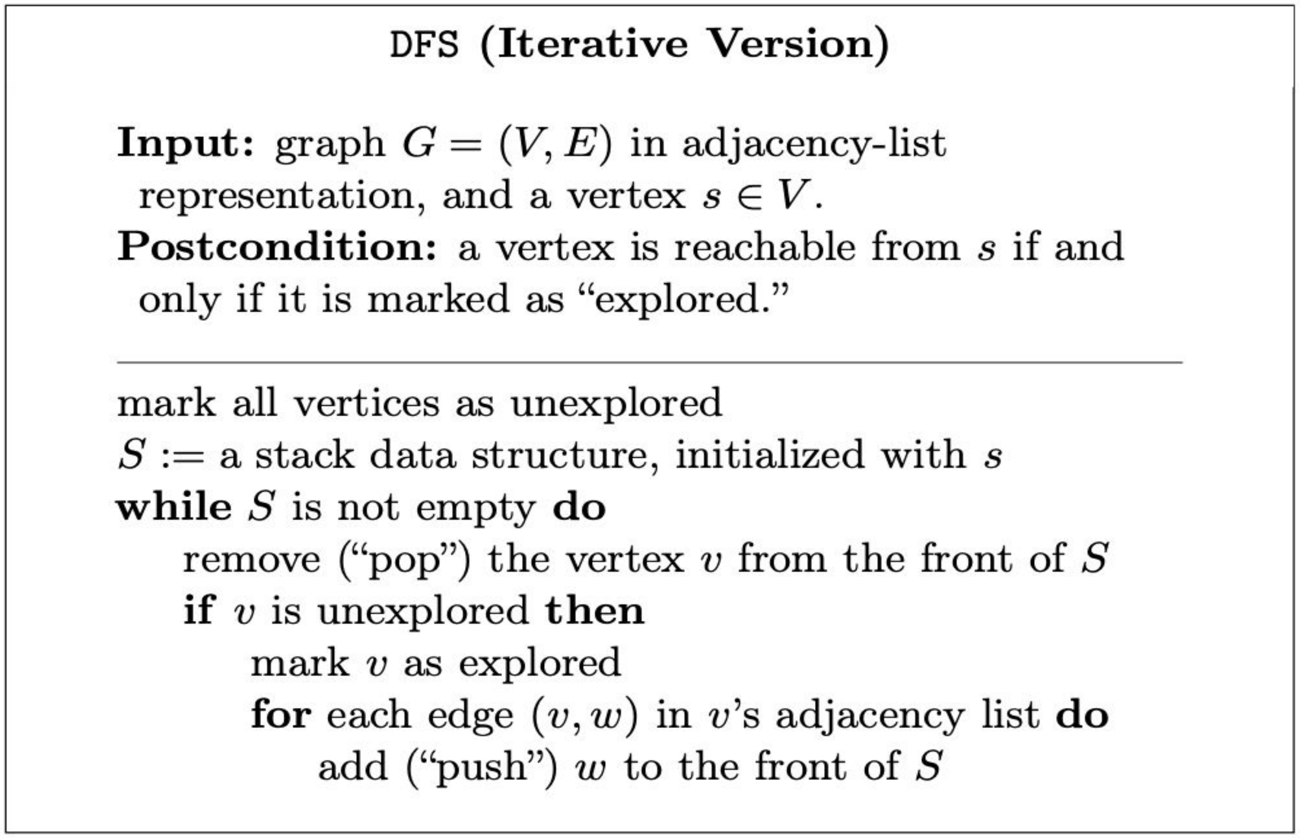


Queue



Stack

Depth-first search (DFS)



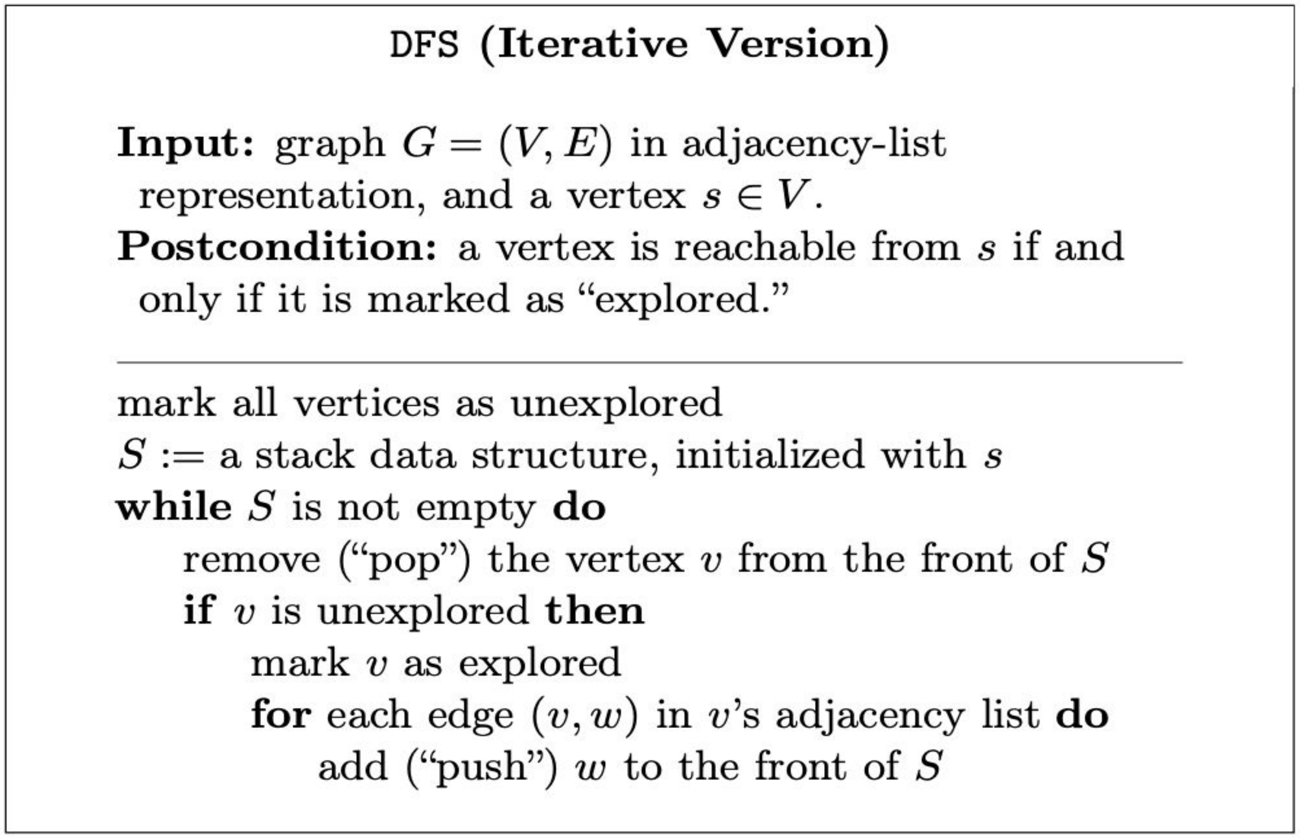
DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.

Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

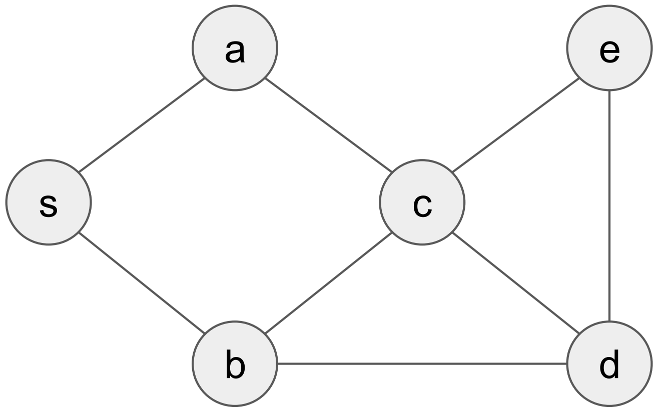
Depth-first search (DFS)



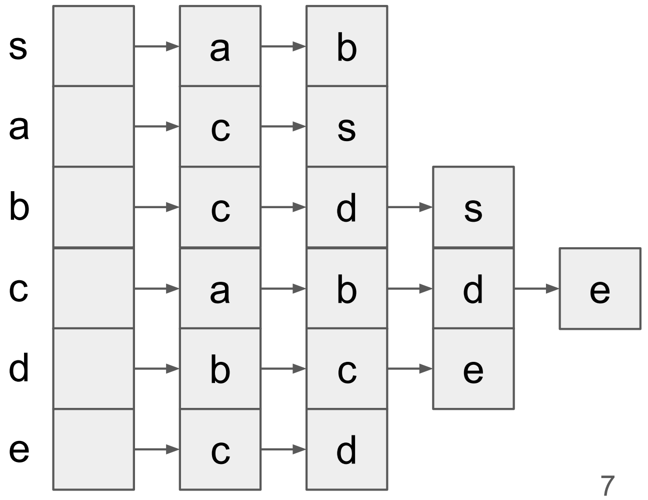
DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

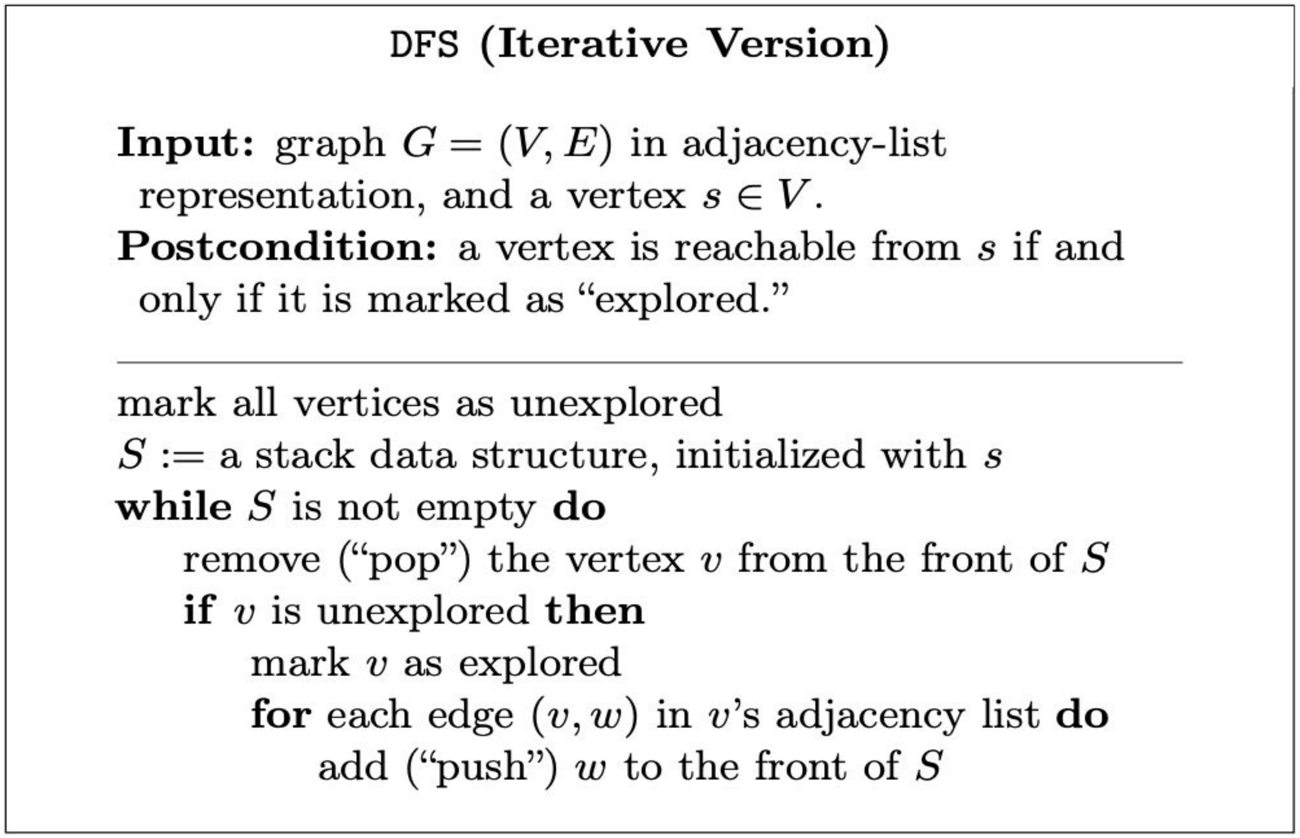
```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```



Let's assume the following adjacency-list:

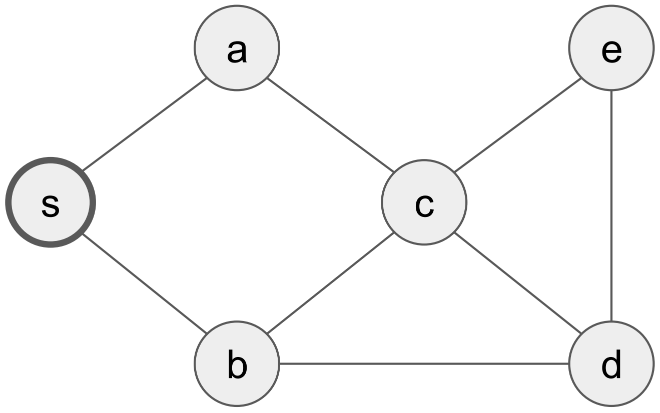


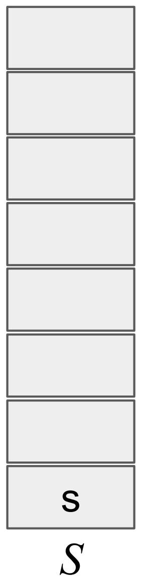
Depth-first search (DFS)



```
DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list
  representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from $s$ if and
  only if it is marked as "explored."
------------------------------------------------------
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```





Explored:

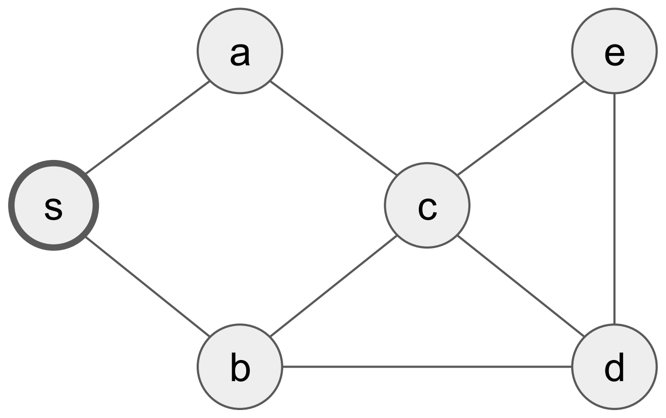
$\varnothing$

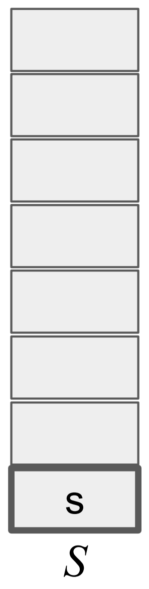
Depth-first search (DFS)

DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

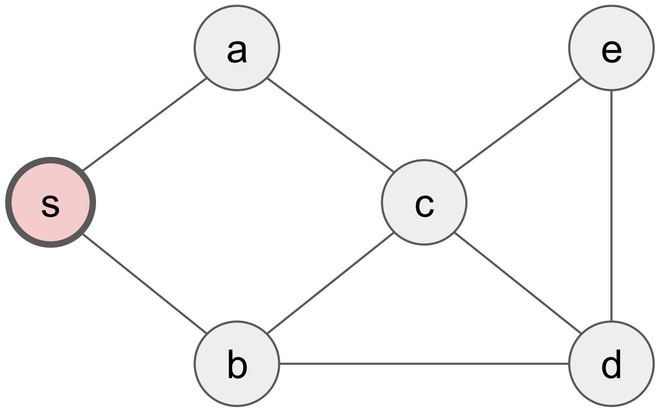
```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```





Explored:
$\varnothing$

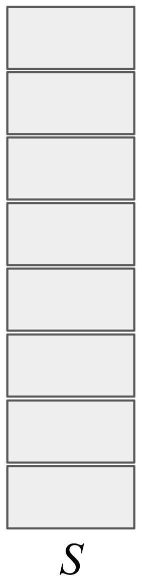
Depth-first search (DFS)



DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```



Explored:
s

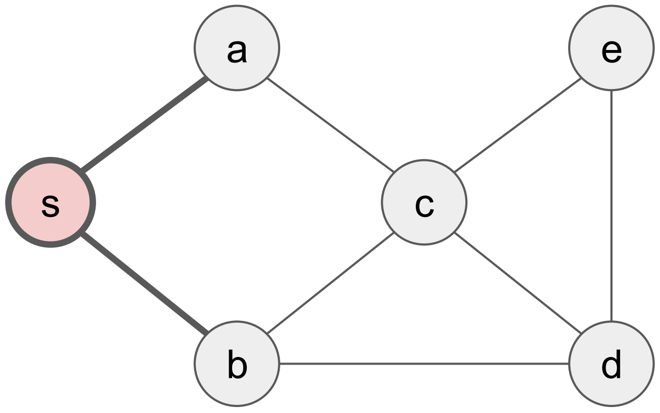
v = s

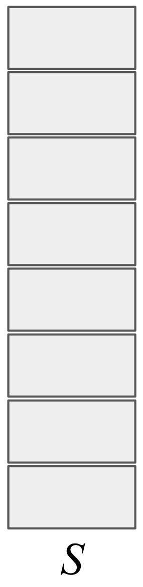
Depth-first search (DFS)

DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

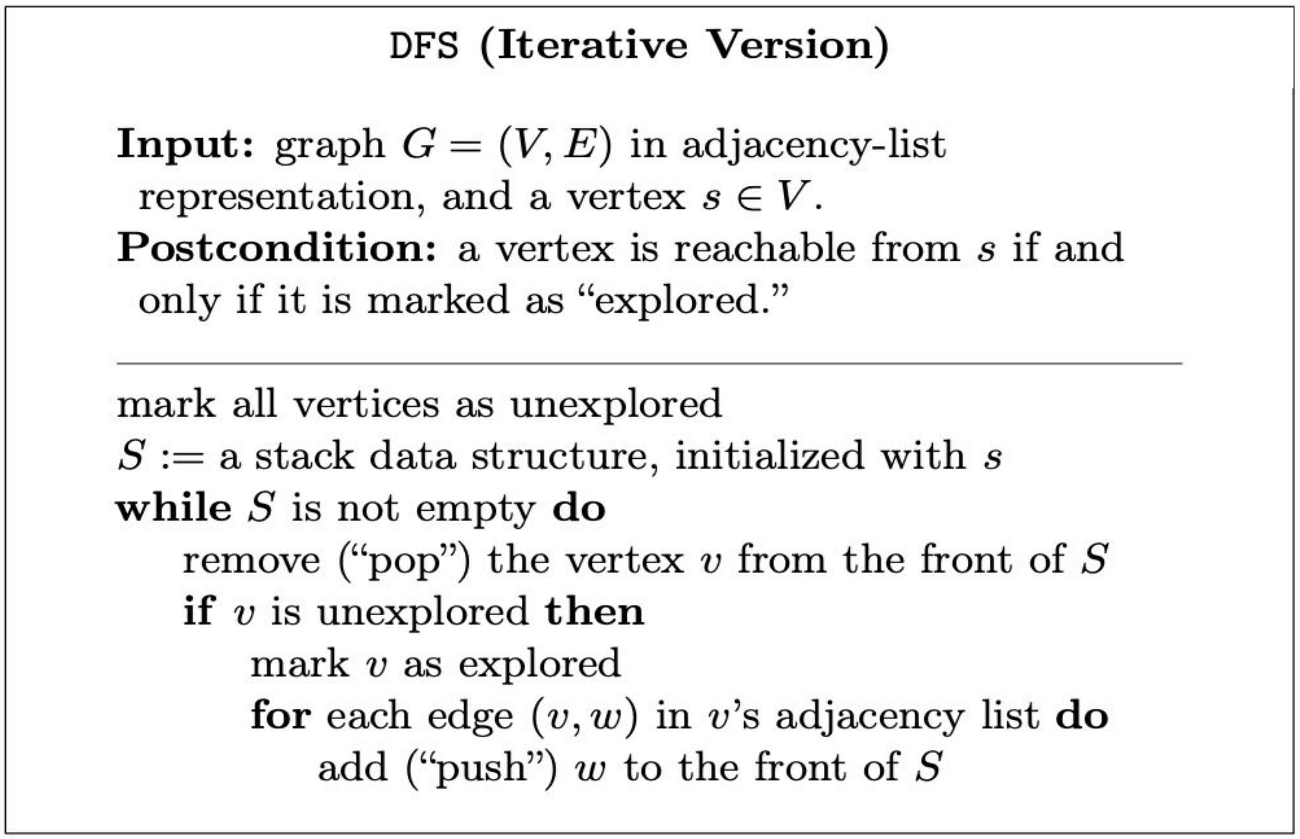




Explored:
s

v = s

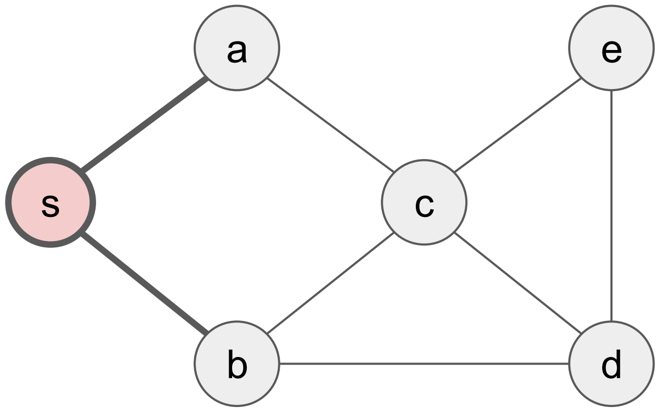
Depth-first search (DFS)

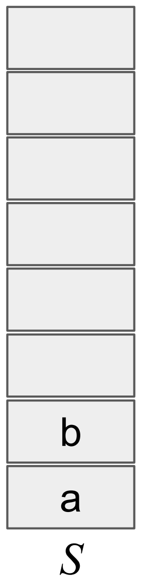


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

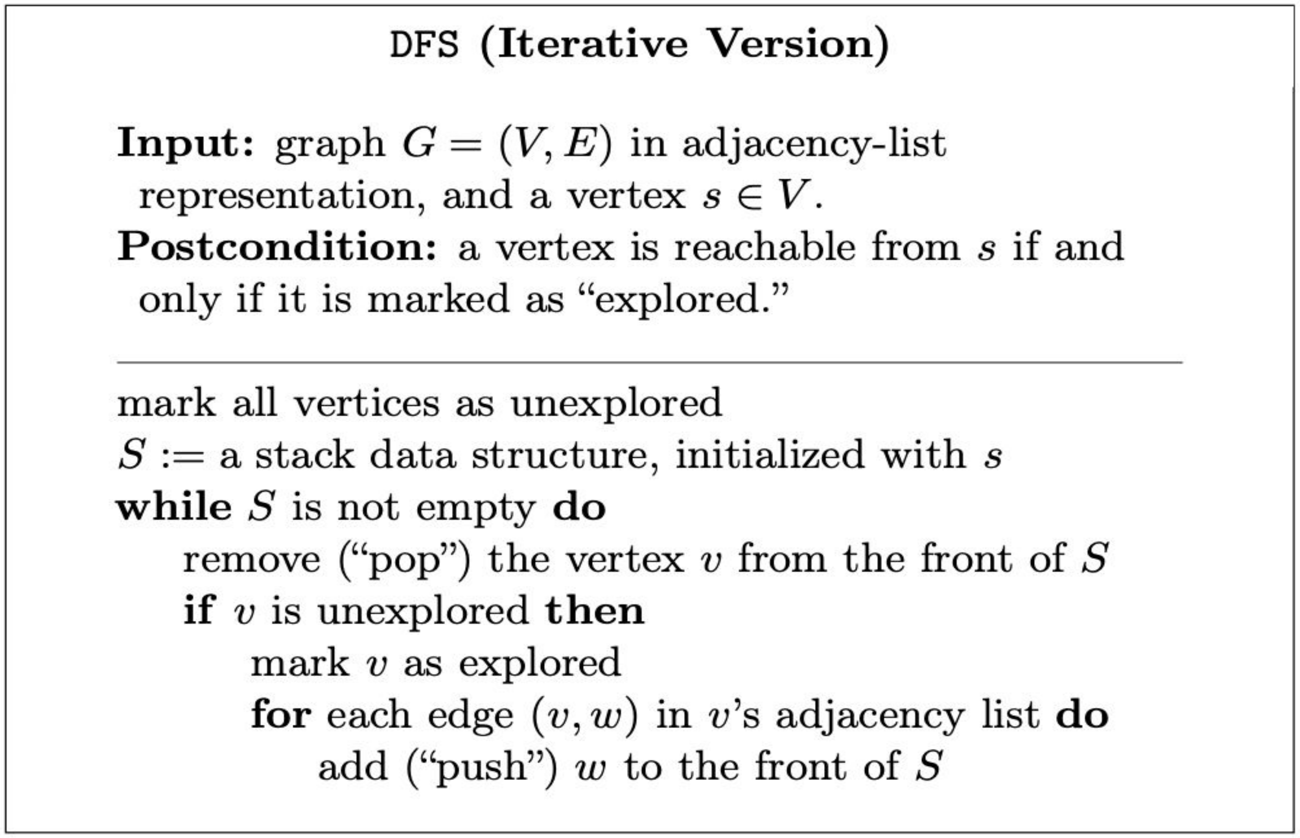




Explored:
s

v = s

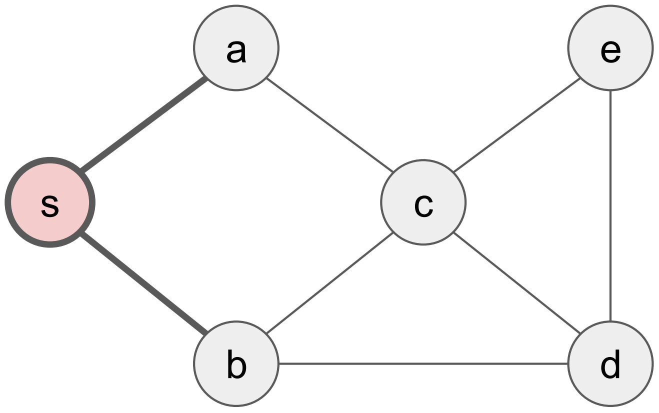
Depth-first search (DFS)

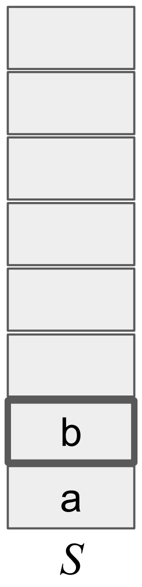


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
**while** $S$ is not empty **do**
    remove ("pop") the vertex $v$ from the front of $S$
    **if** $v$ is unexplored **then**
        mark $v$ as explored
        **for** each edge $(v, w)$ in $v$'s adjacency list **do**
            add ("push") $w$ to the front of $S$
```

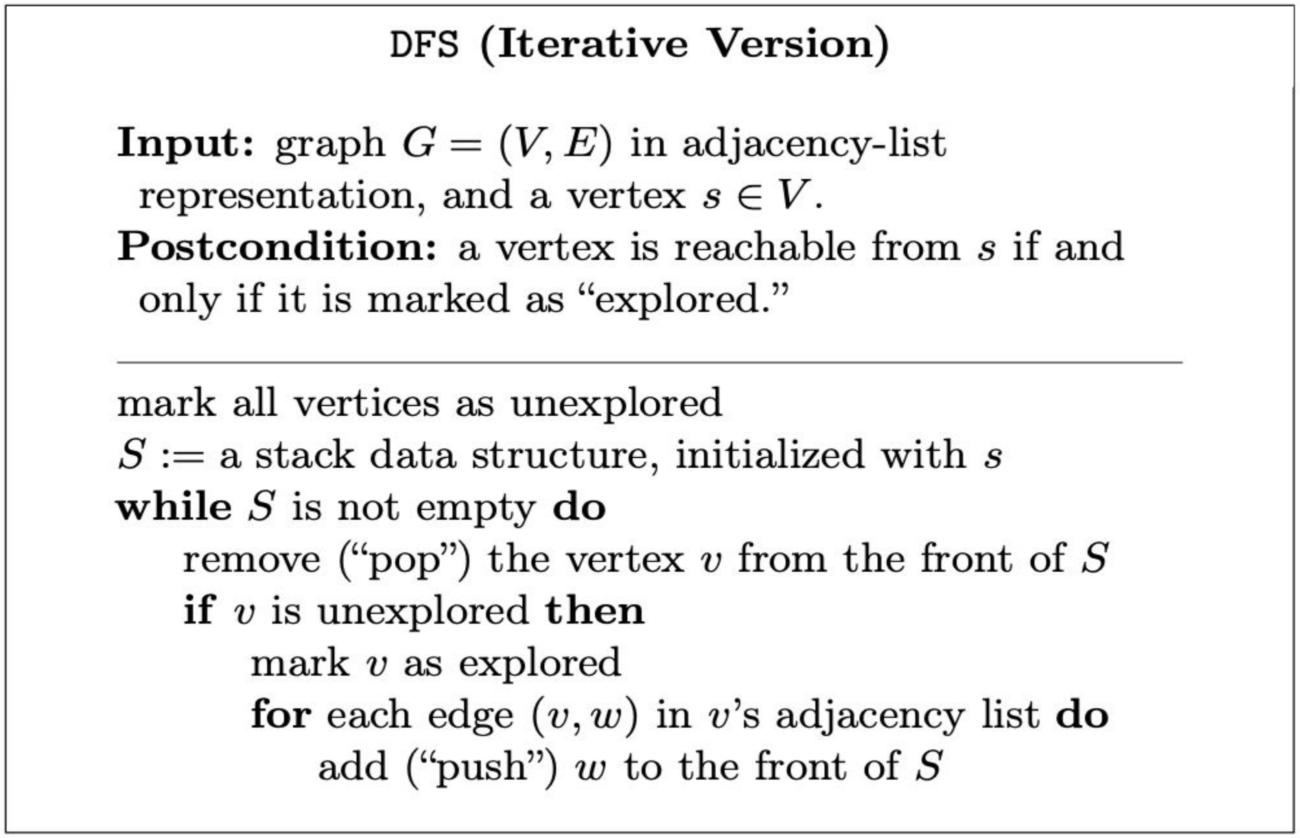




Explored:
s

v = s

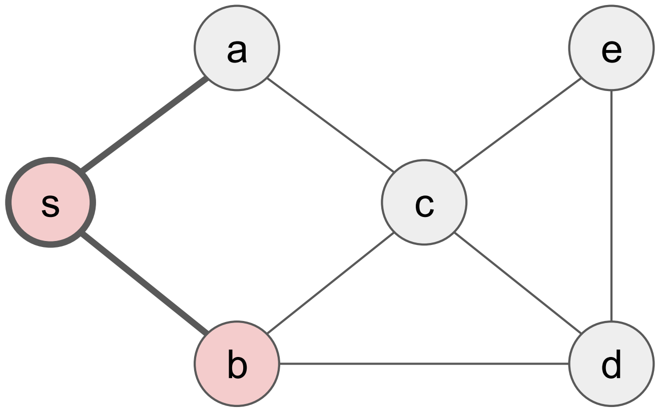
Depth-first search (DFS)

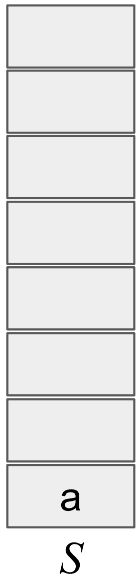


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
**while** $S$ is not empty **do**
    remove ("pop") the vertex $v$ from the front of $S$
    **if** $v$ is unexplored **then**
        mark $v$ as explored
        **for** each edge $(v, w)$ in $v$'s adjacency list **do**
            add ("push") $w$ to the front of $S$
```

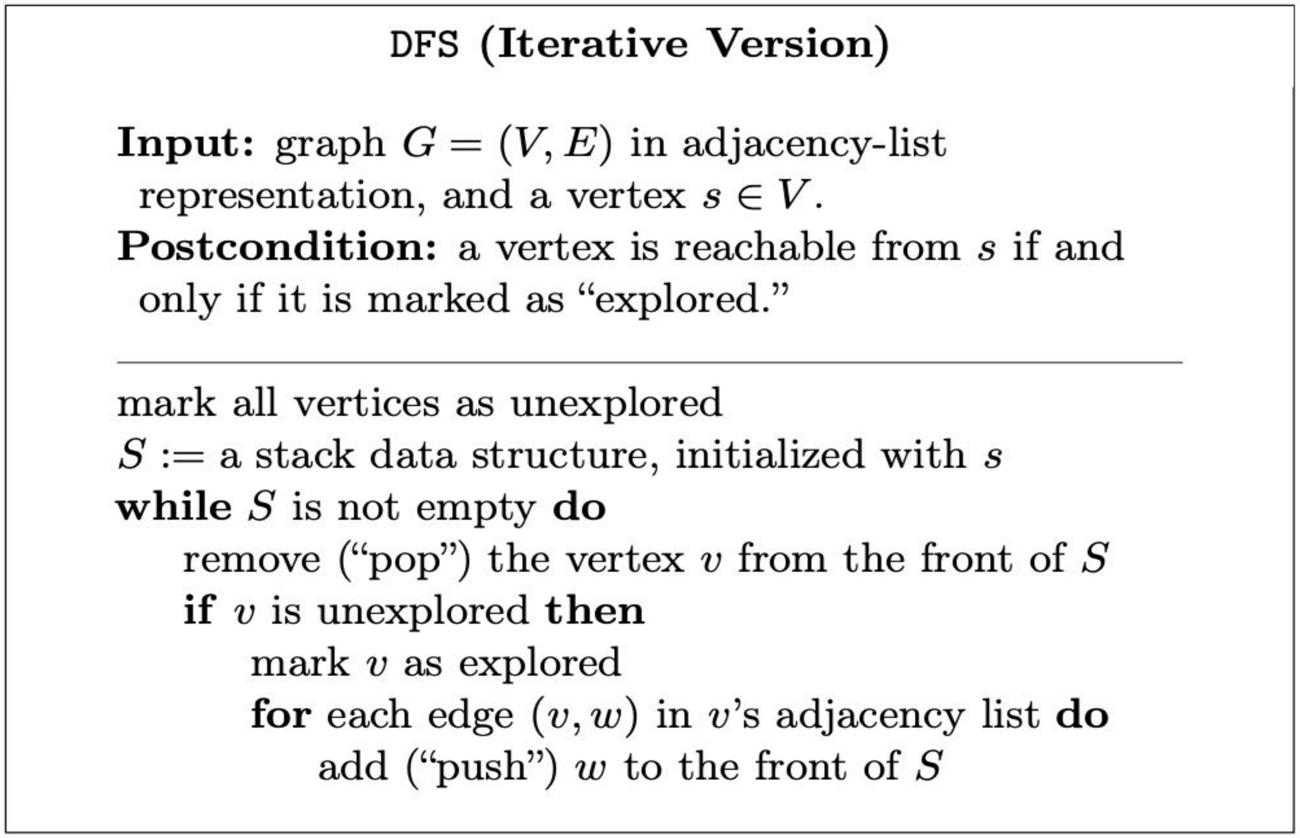




Explored:
s, b

v = b

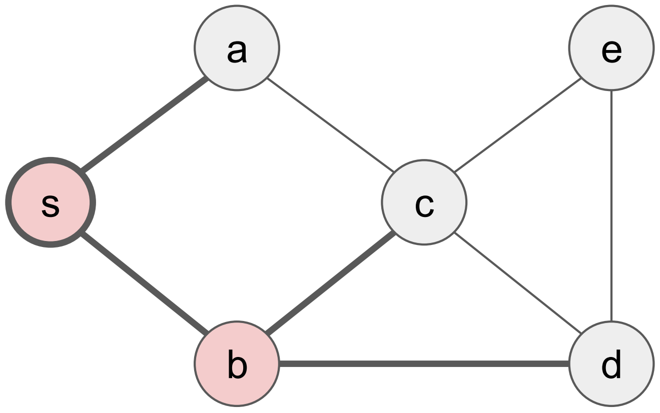
Depth-first search (DFS)

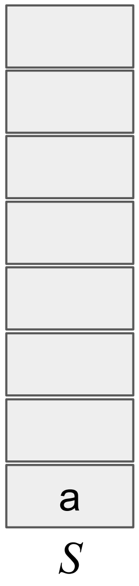


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

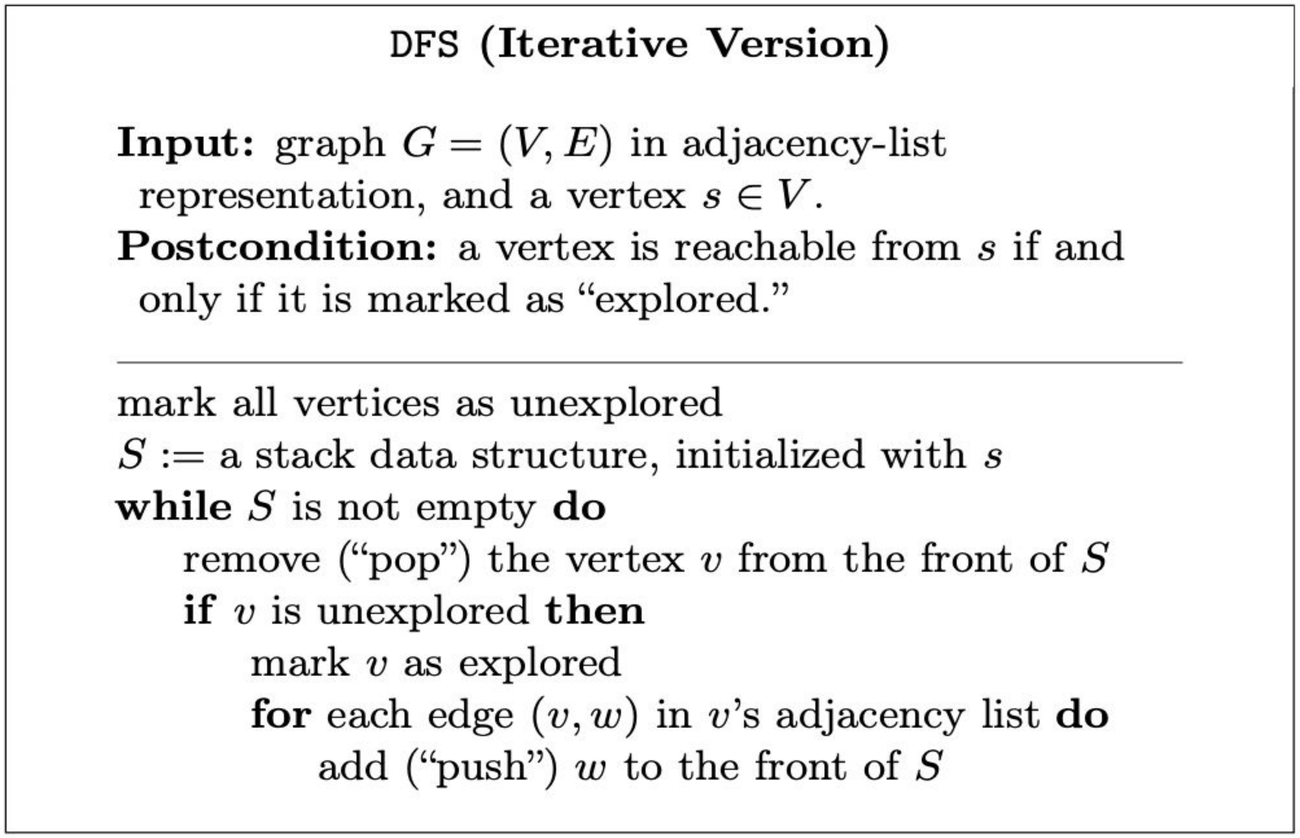




Explored:
s, b

v = b

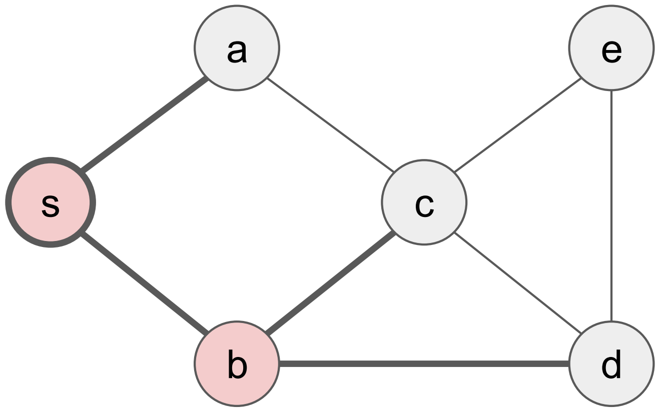
Depth-first search (DFS)

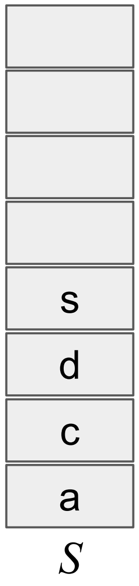


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

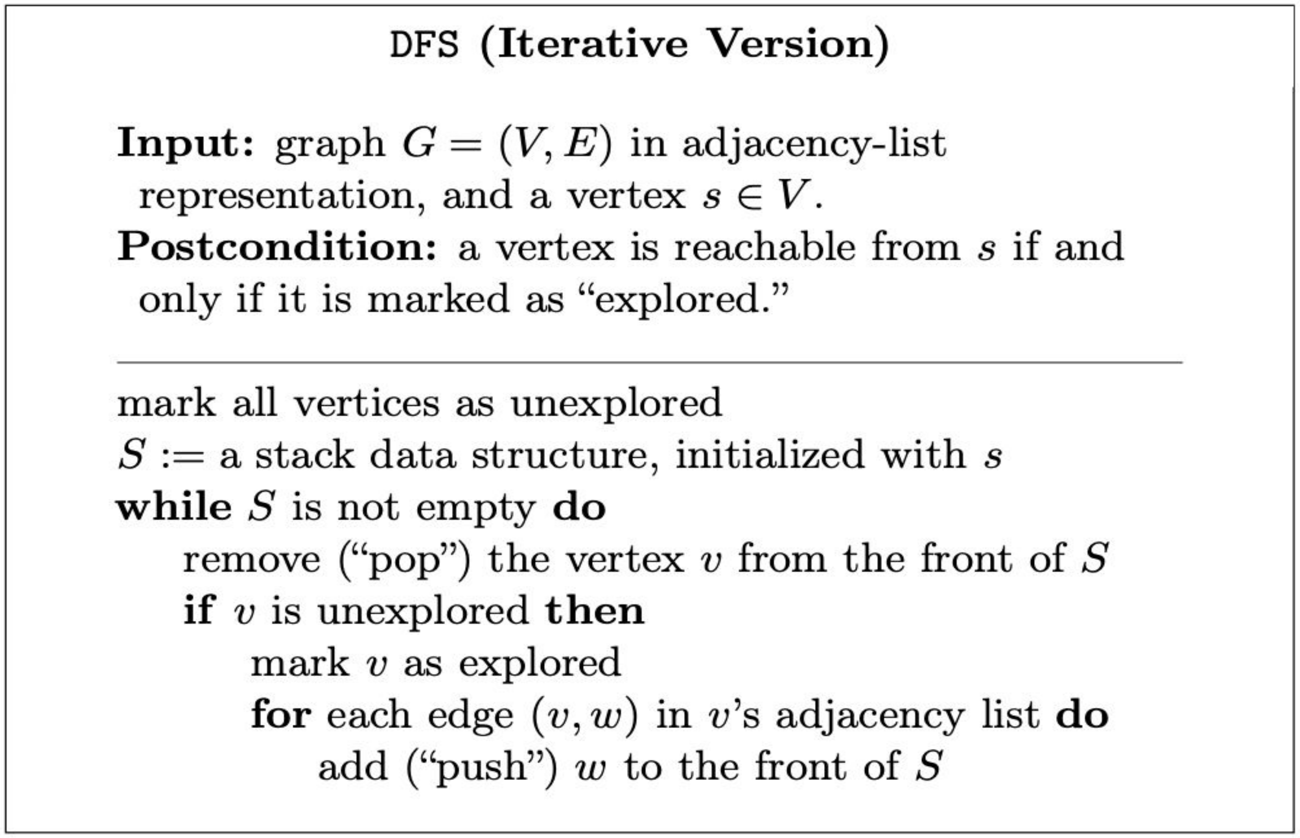




Explored:
s, b

v = b

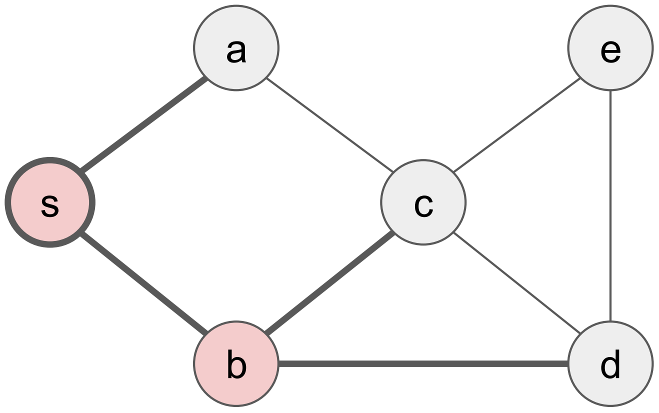
Depth-first search (DFS)

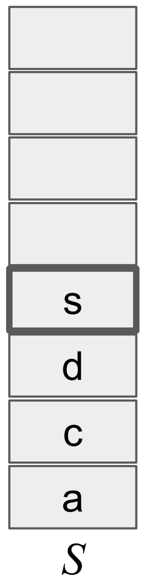


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

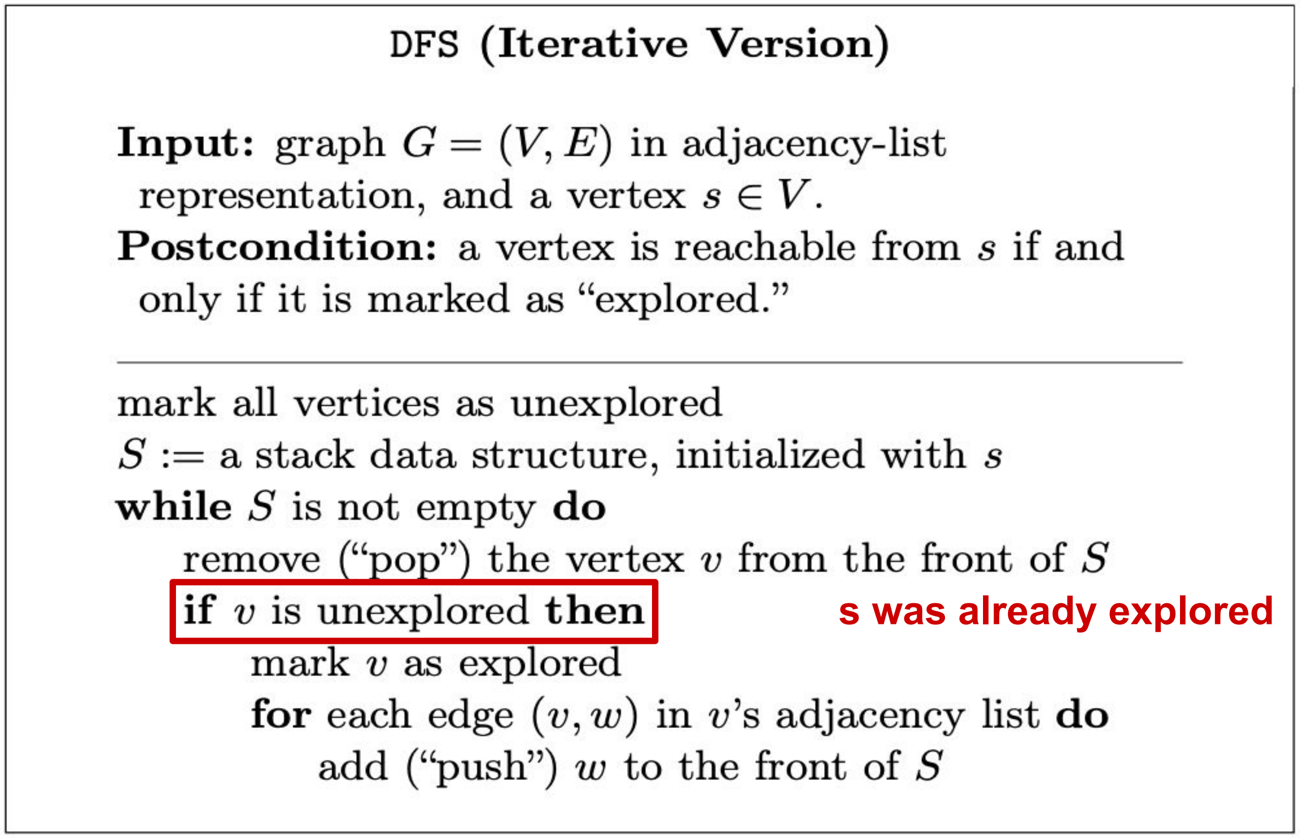




Explored:
s, b

v = b

Depth-first search (DFS)

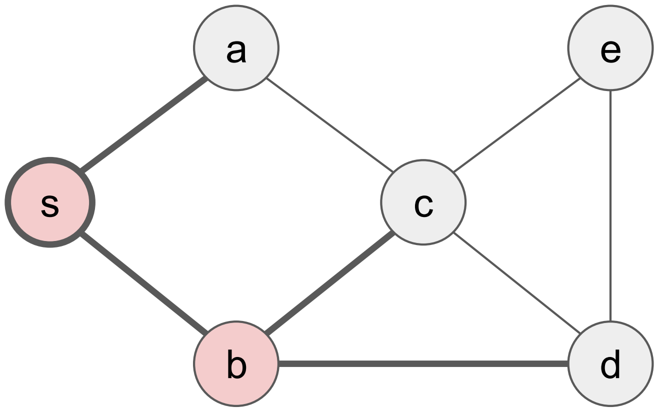


DFS (Iterative Version)

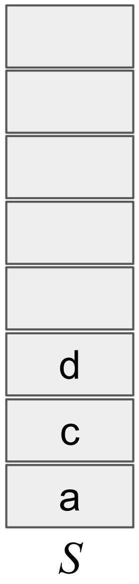
Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

s was already explored

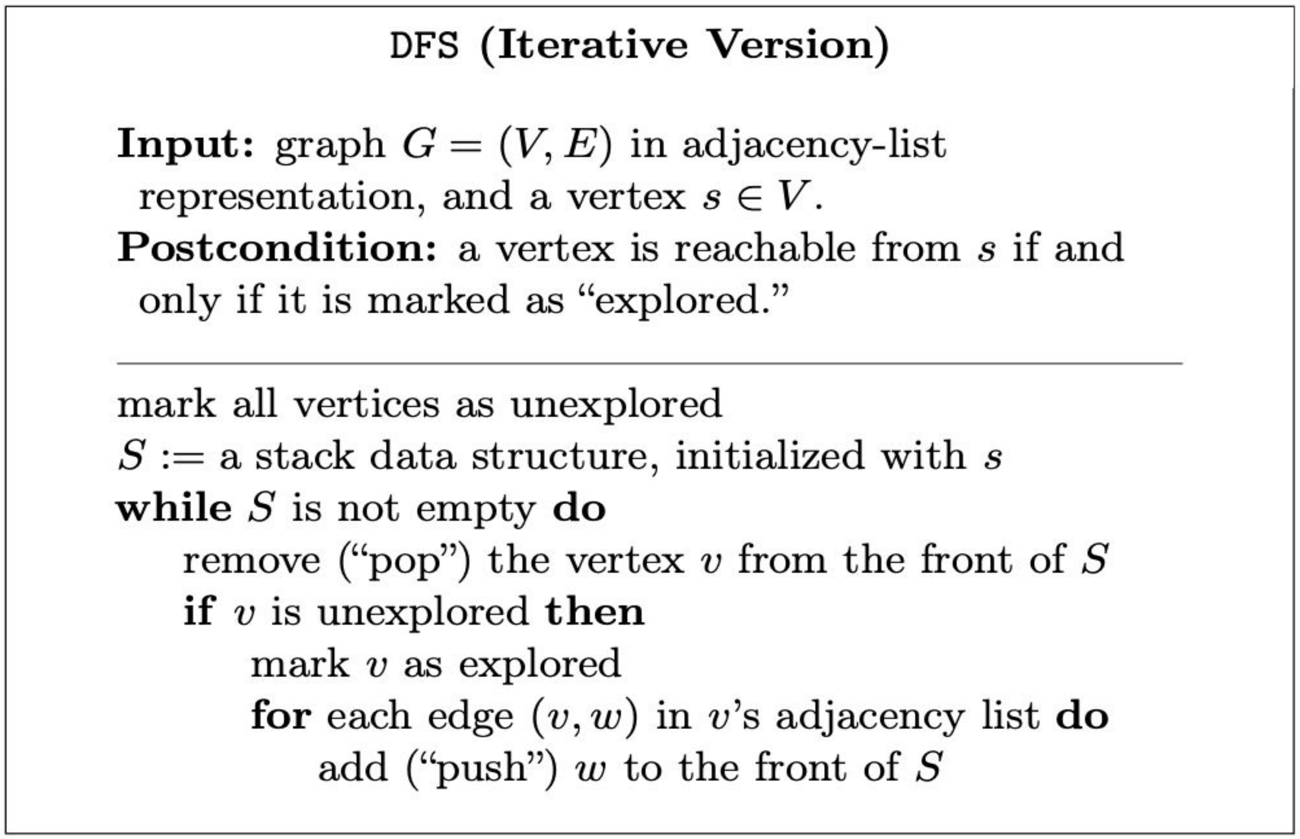


Explored:
s, b



$v =$ s

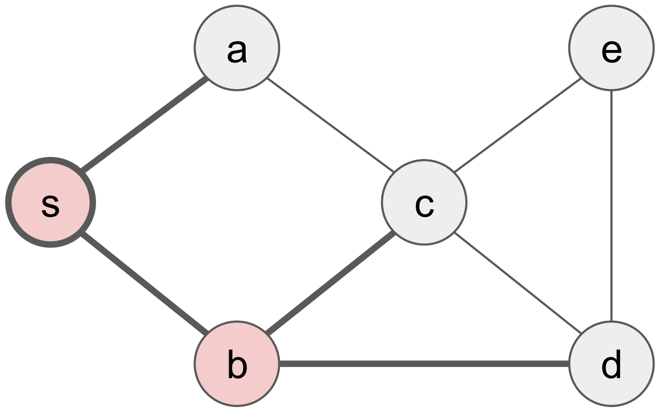
Depth-first search (DFS)

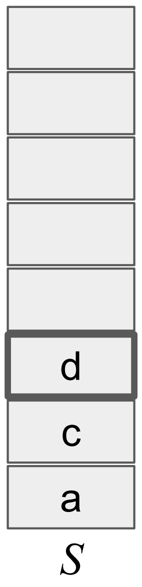


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

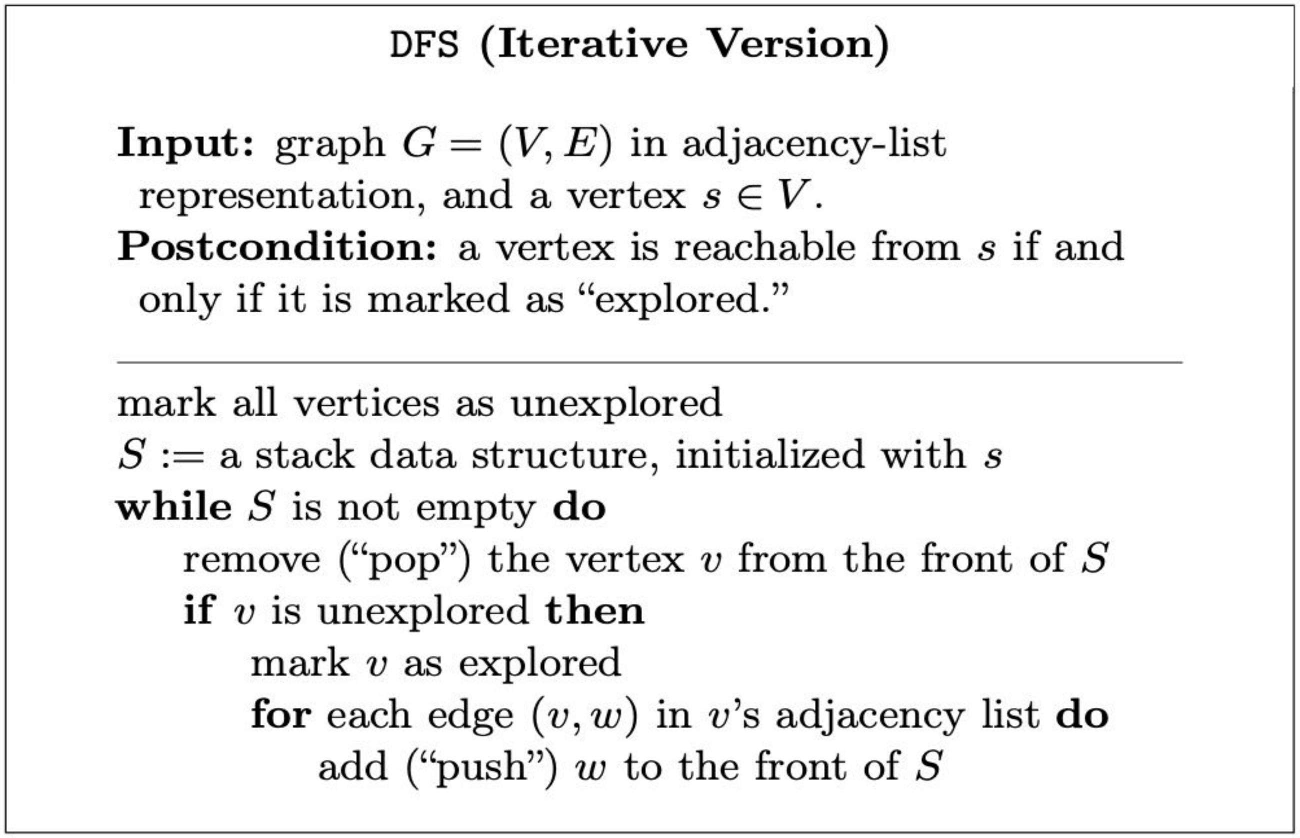




Explored:
s, b

v = s

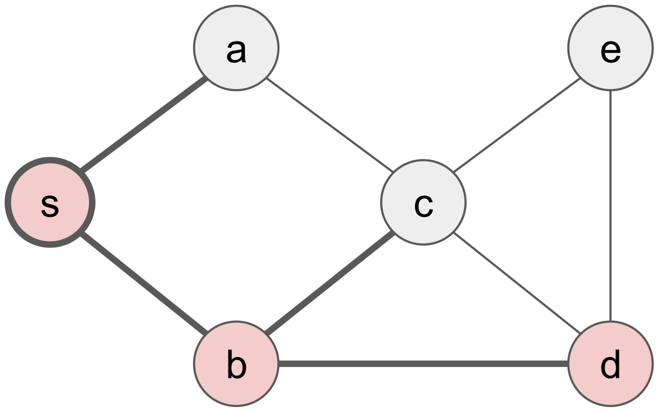
Depth-first search (DFS)

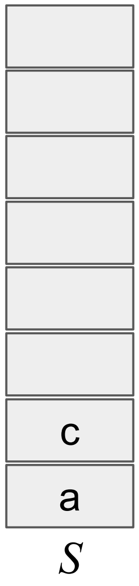


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

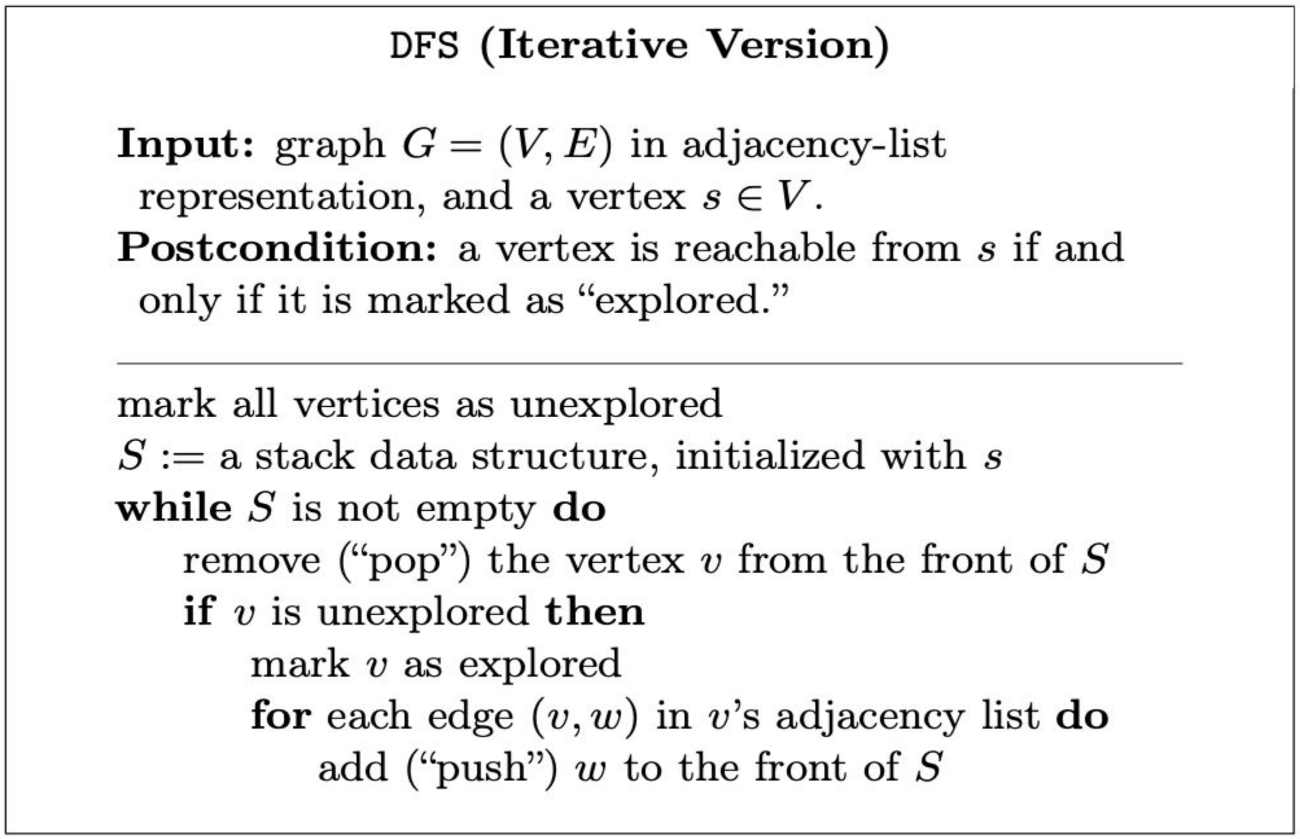




Explored:
s, b, d

v = d

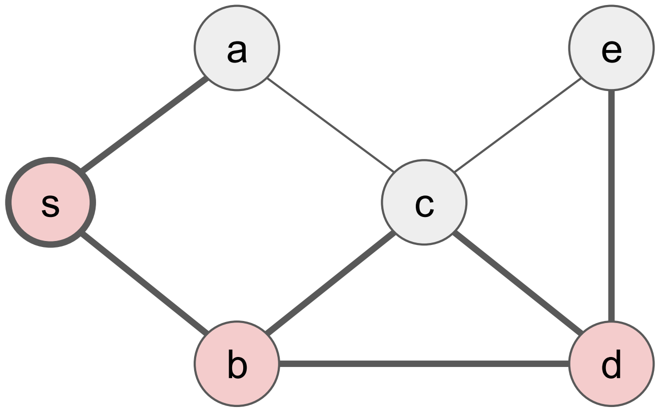
Depth-first search (DFS)

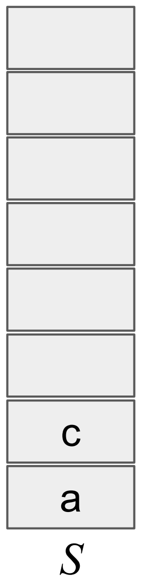


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```





Explored:
s, b, d

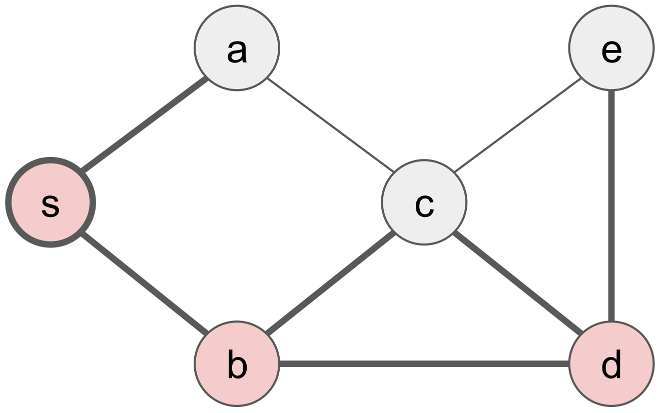
v = d

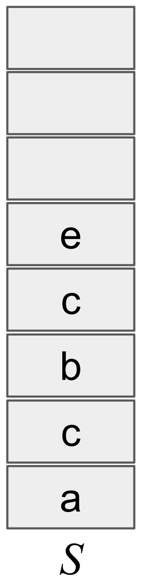
Depth-first search (DFS)

DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

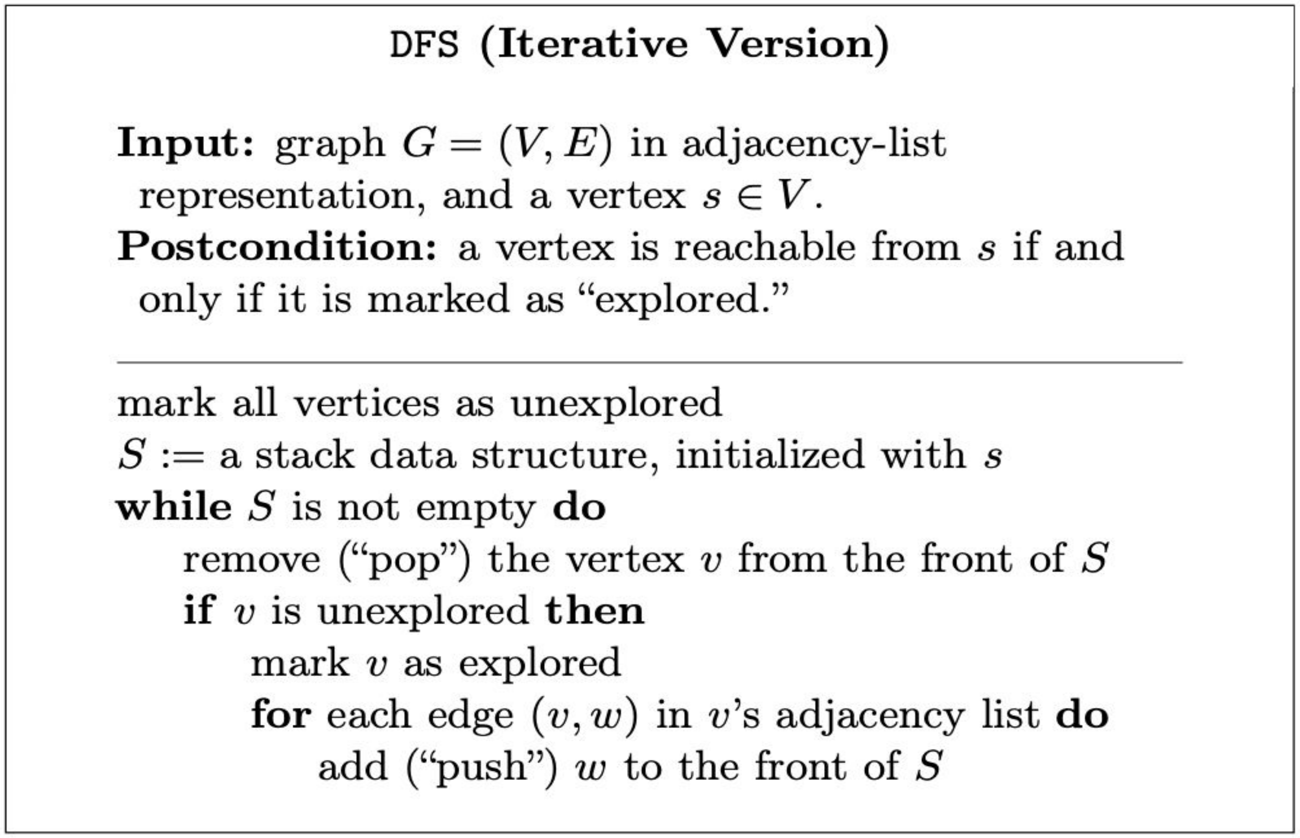




Explored:
s, b, d

v = d

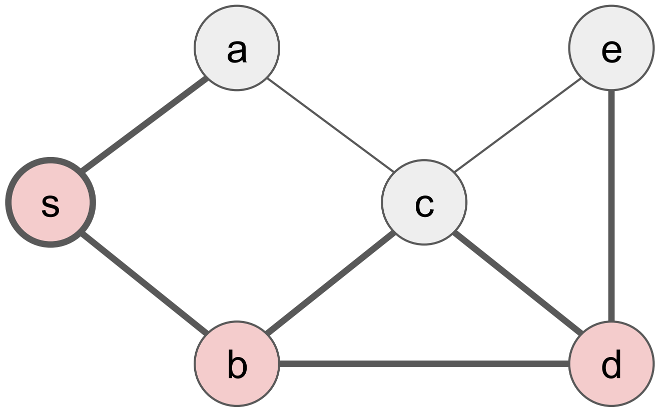
Depth-first search (DFS)

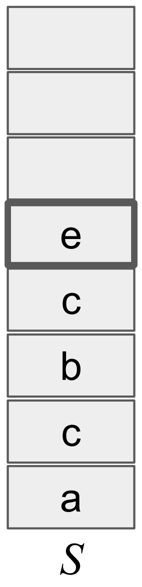


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

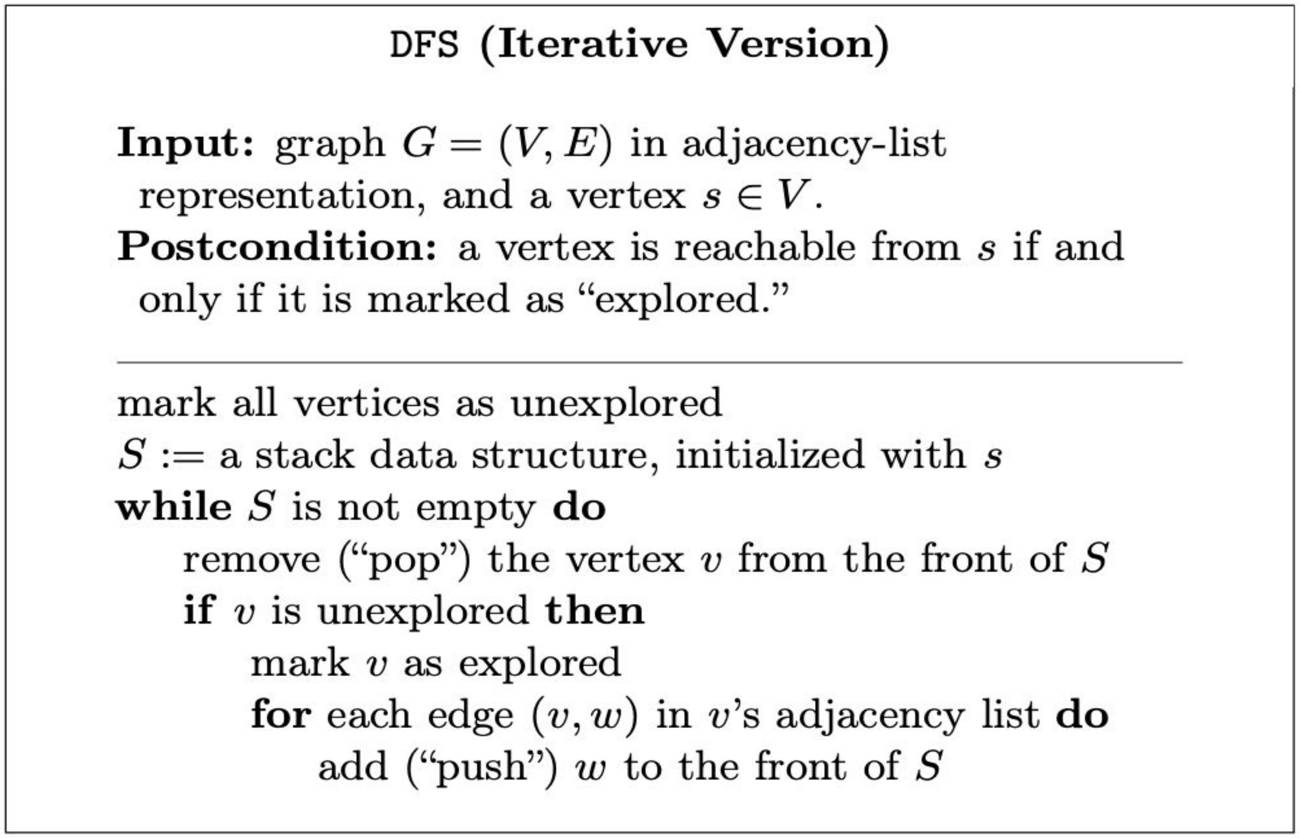




Explored:
s, b, d

v = d

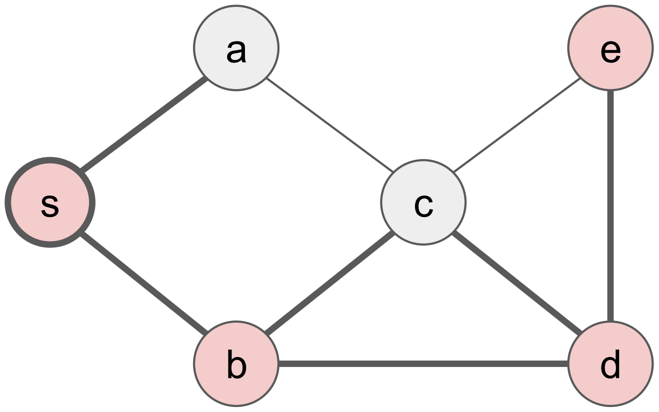
Depth-first search (DFS)

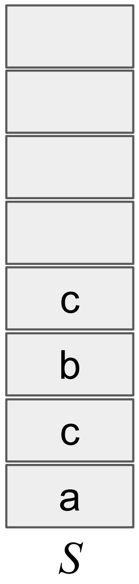


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

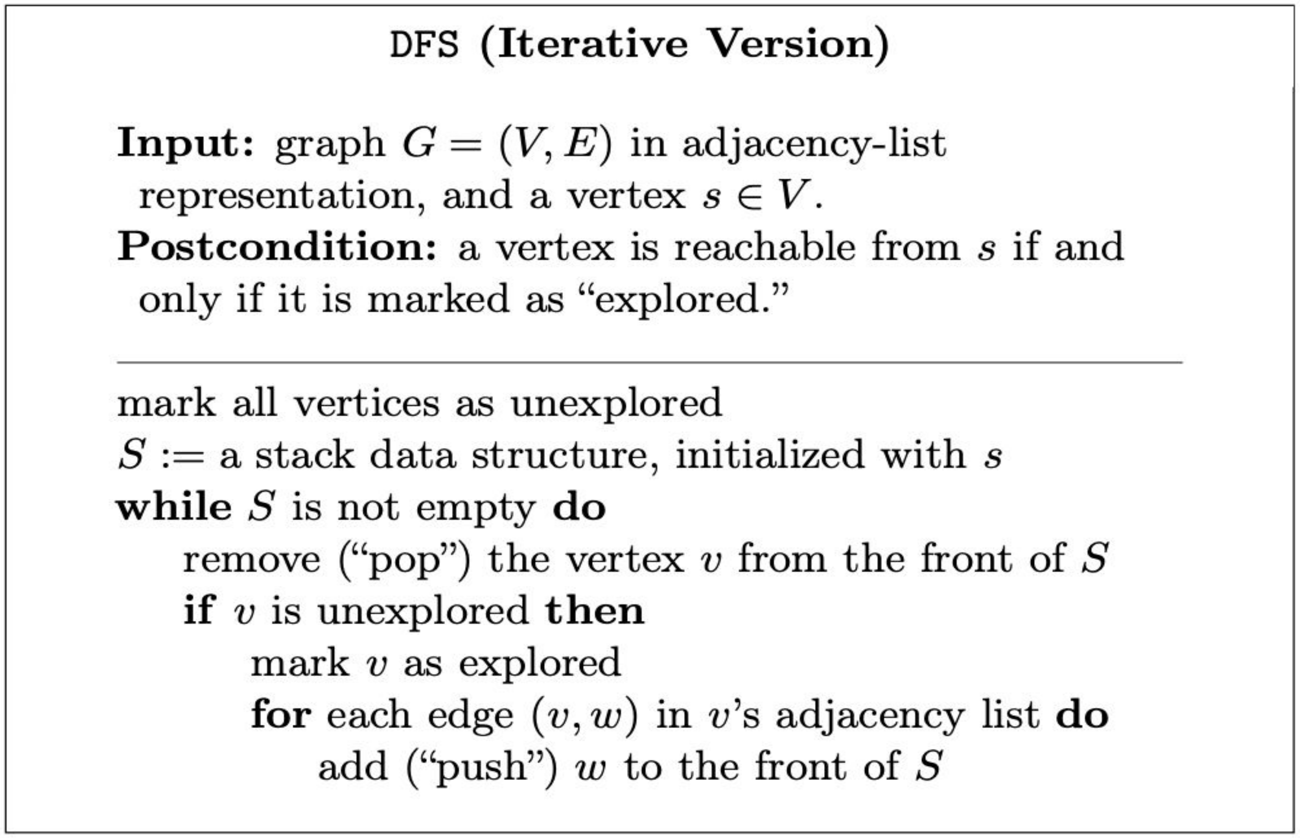




Explored:
s, b, d, e

v = e

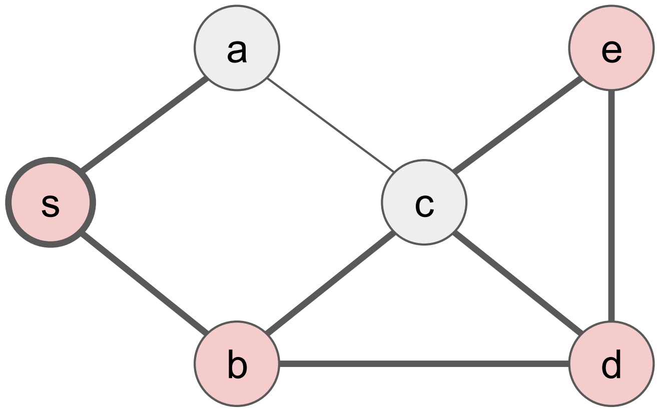
Depth-first search (DFS)

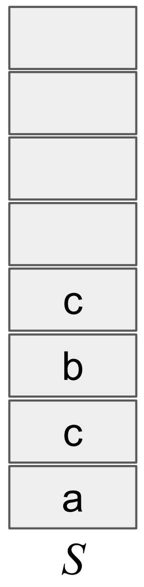


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

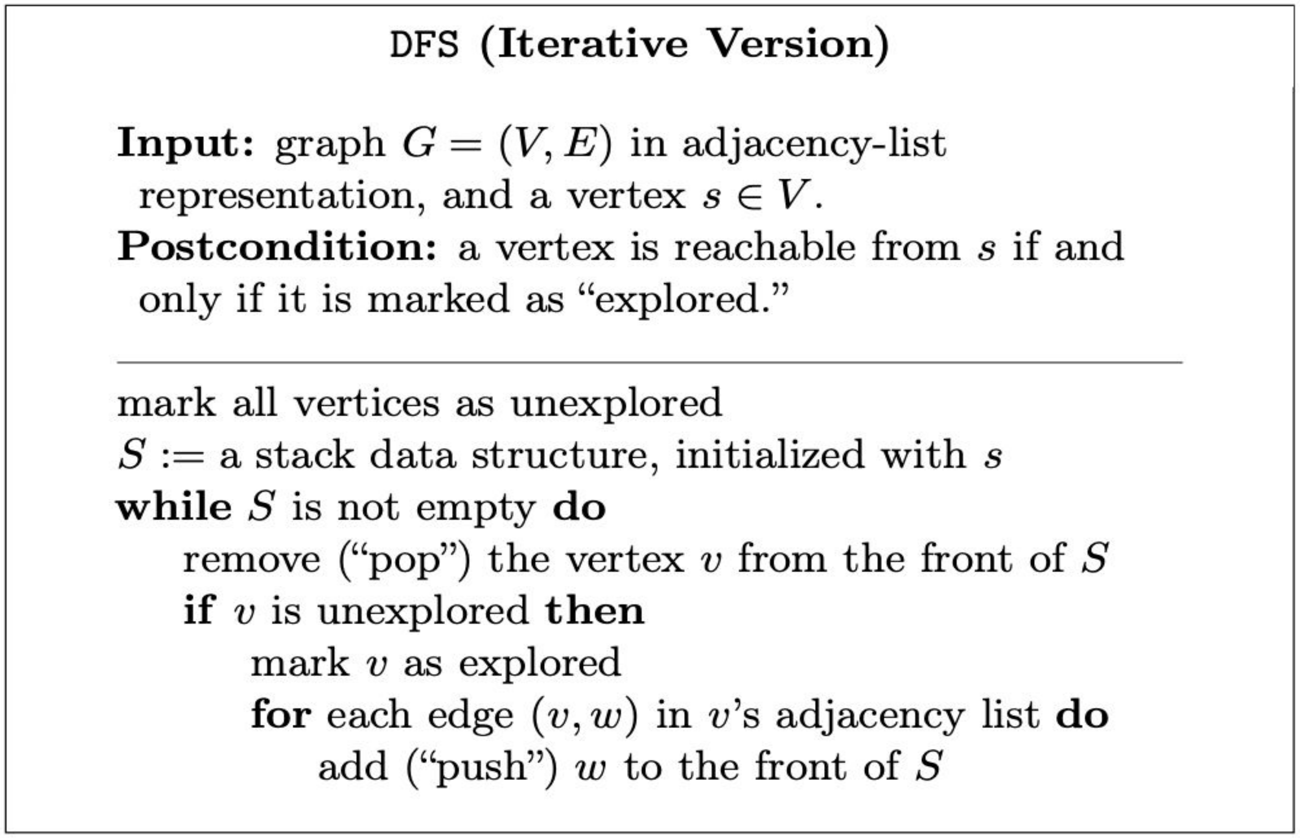




Explored:
s, b, d, e

v = e

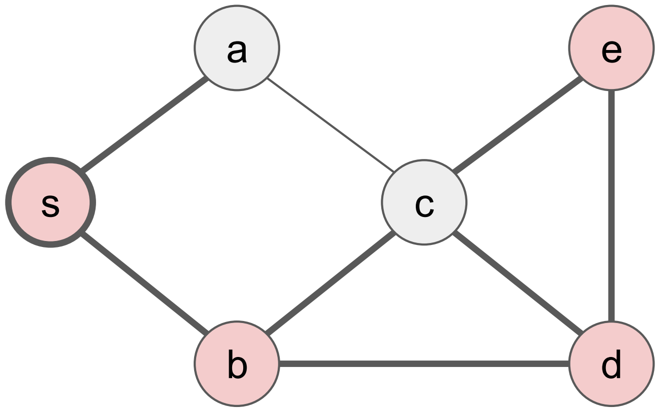
Depth-first search (DFS)

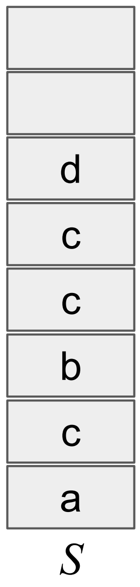


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

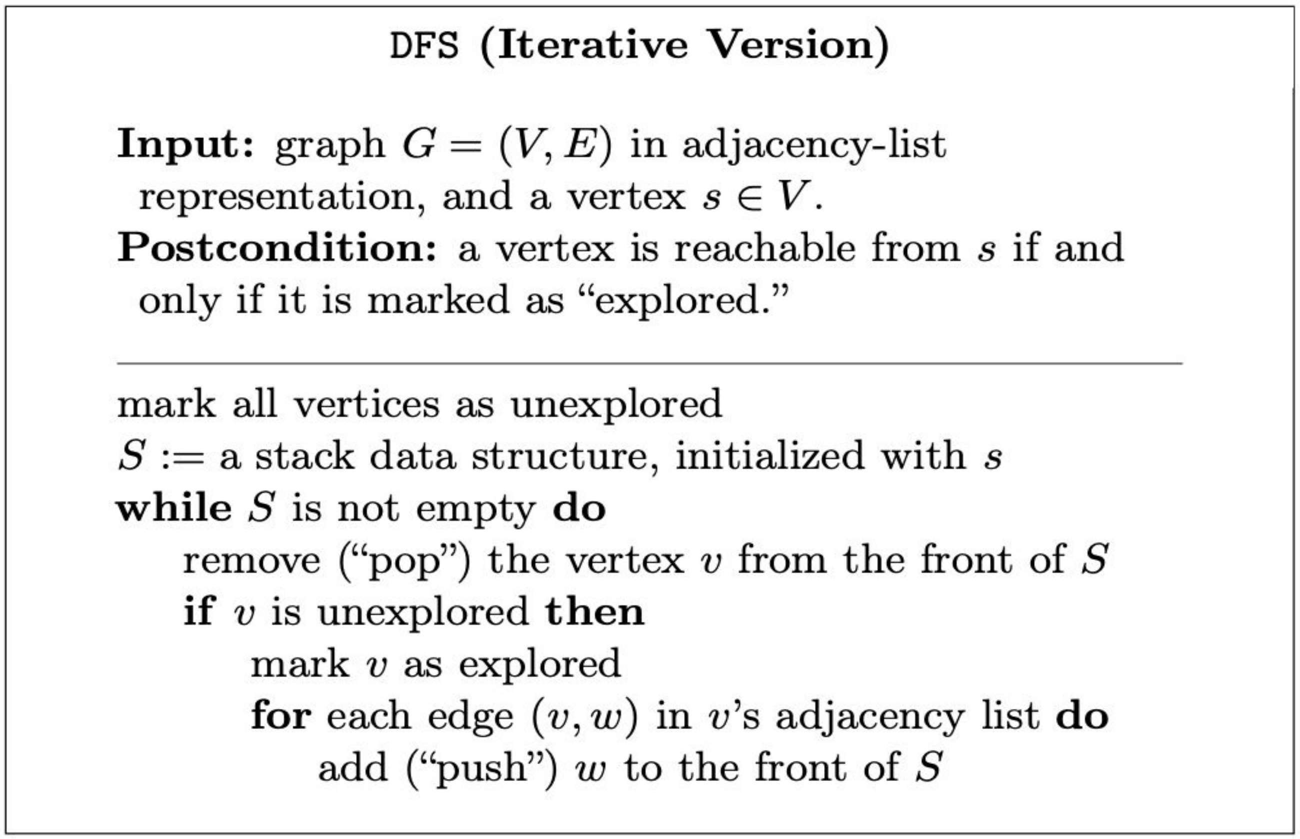




Explored:
s, b, d, e

v = e

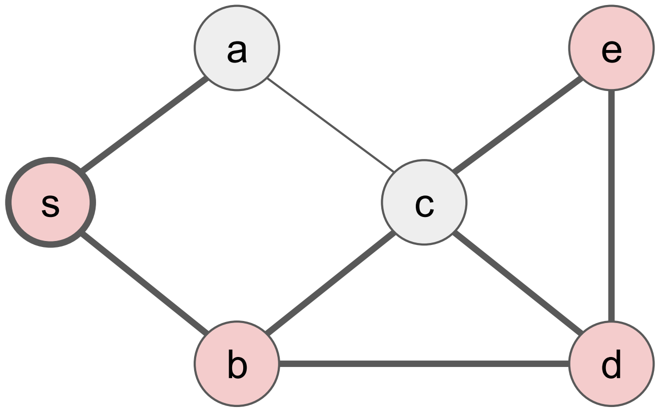
Depth-first search (DFS)

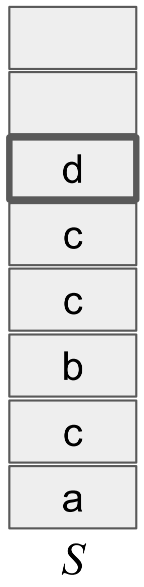


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

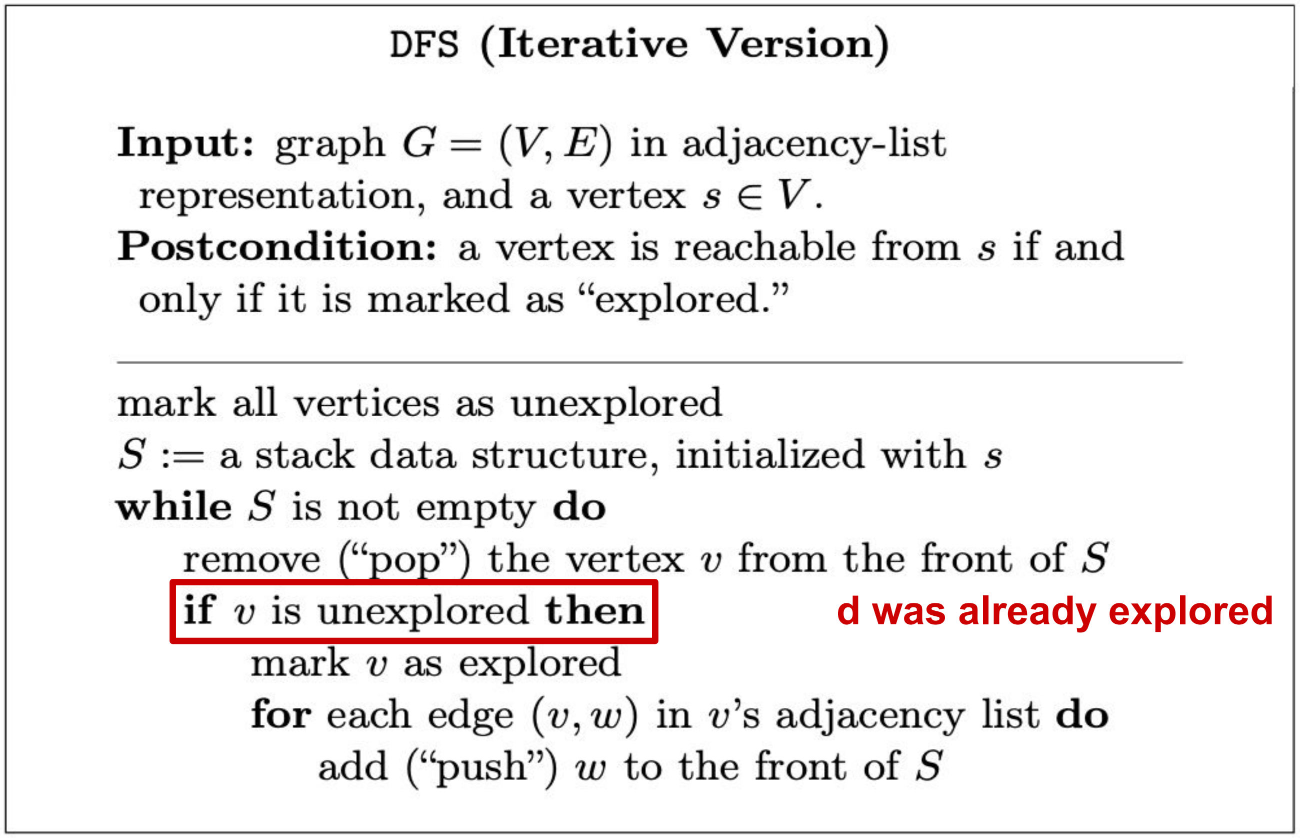




Explored:
s, b, d, e

v = e

Depth-first search (DFS)

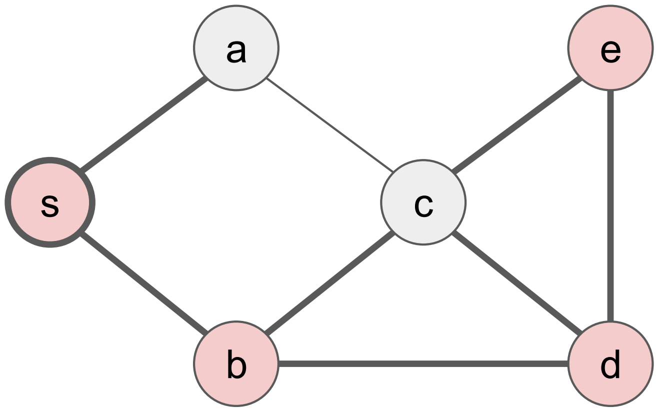


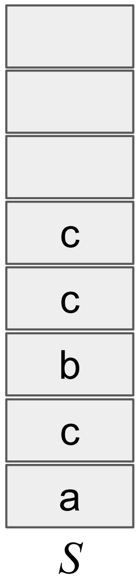
DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

d was already explored

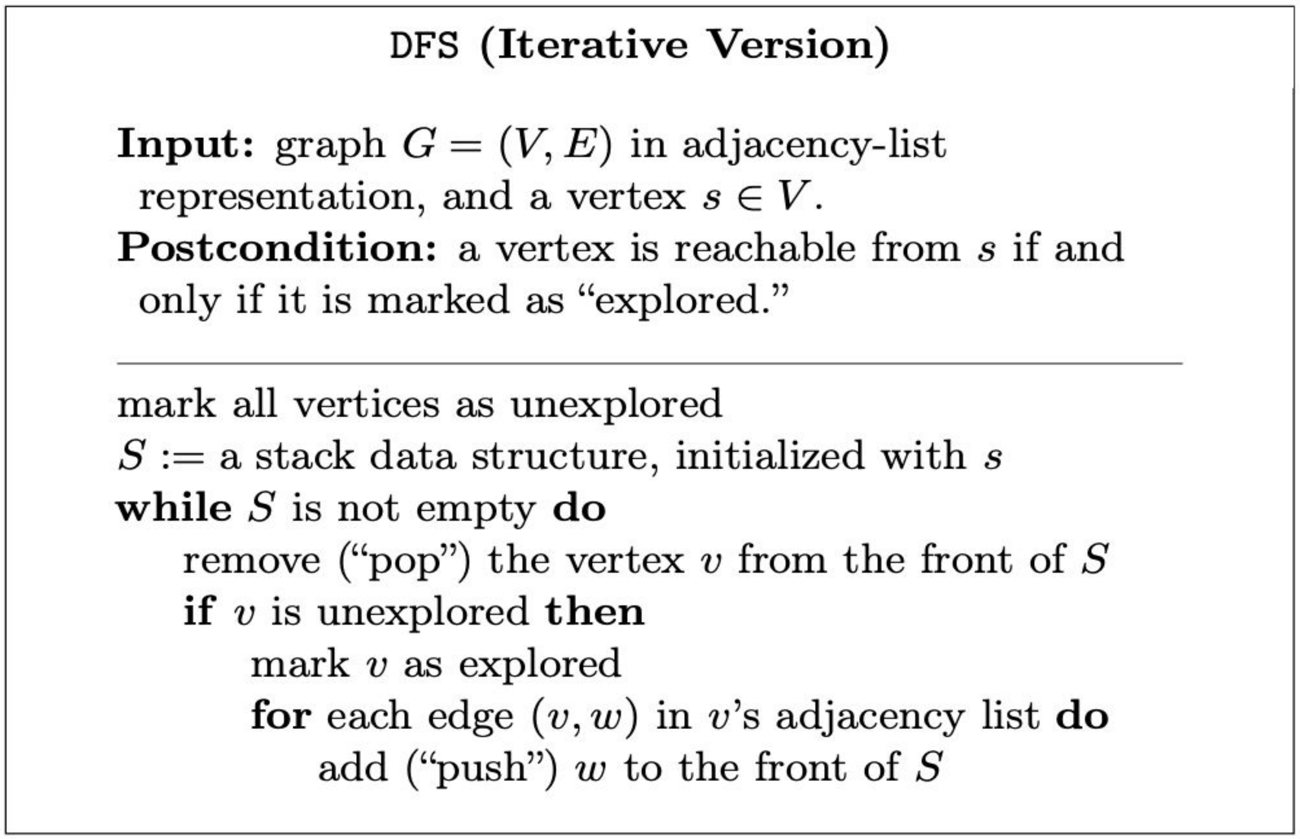




Explored:
s, b, d, e

v = d

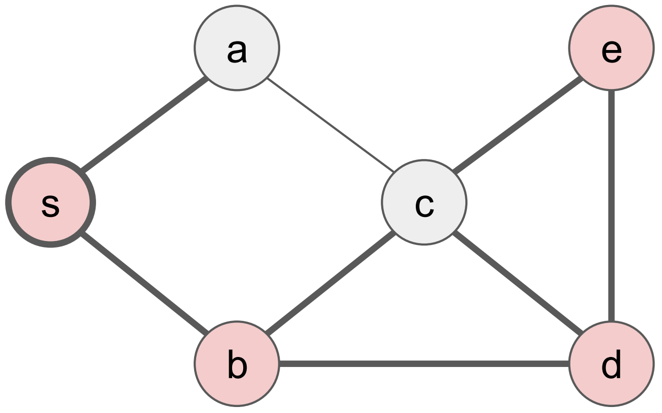
Depth-first search (DFS)

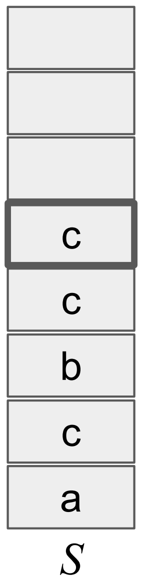


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

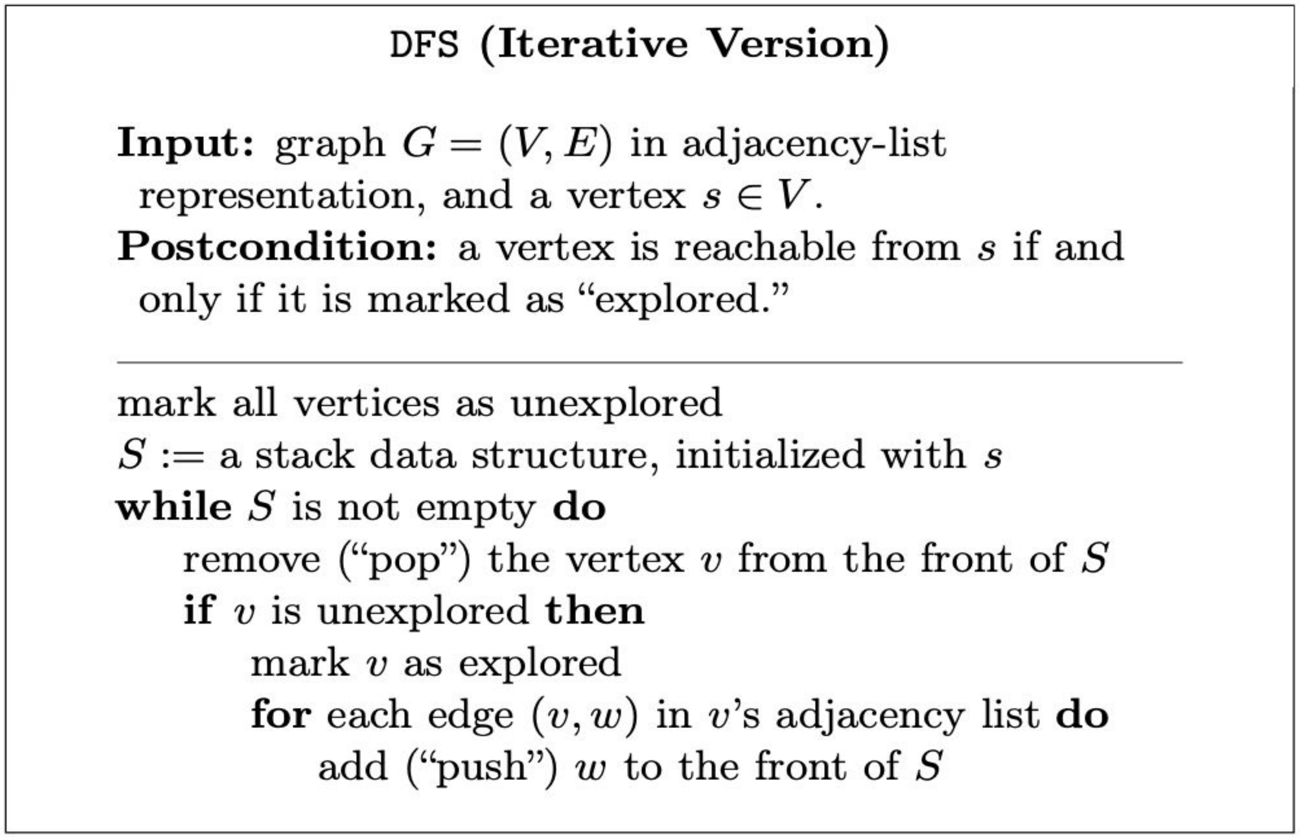




Explored:
s, b, d, e

v = d

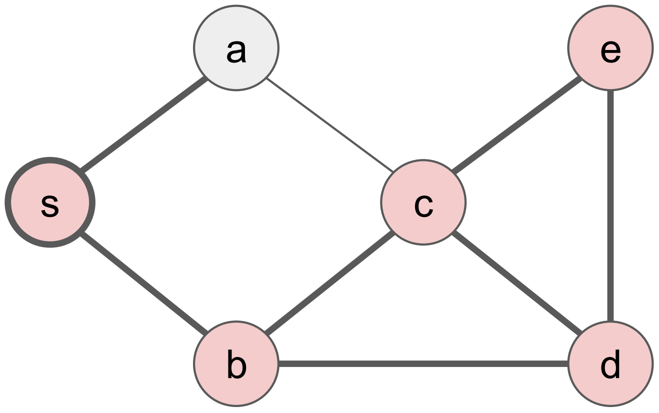
Depth-first search (DFS)

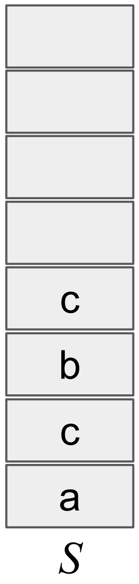


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

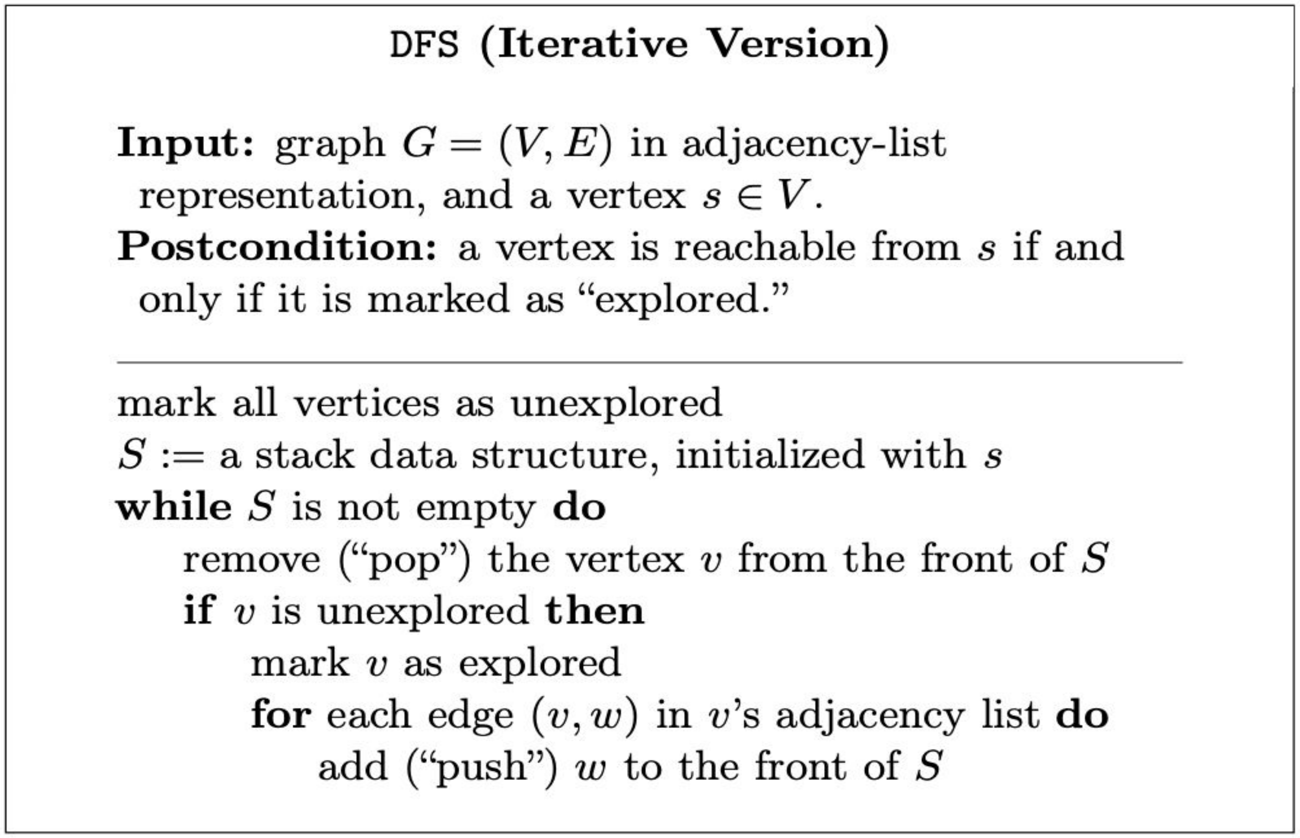




Explored:
s, b, d, e,
c

v = c

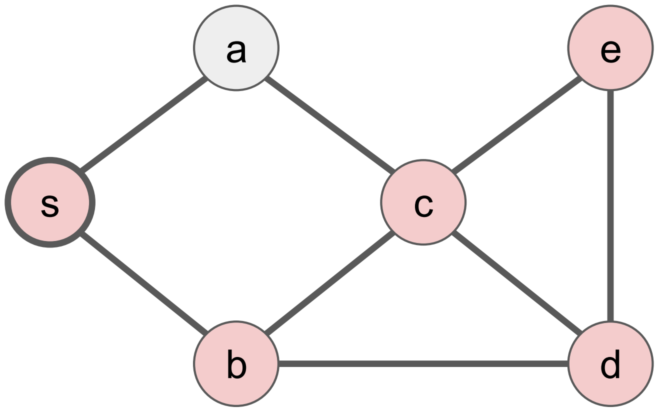
Depth-first search (DFS)

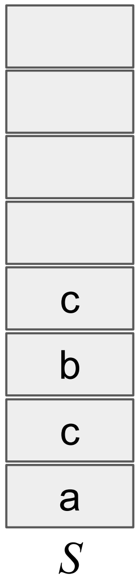


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```





Explored:
s, b, d, e,
c

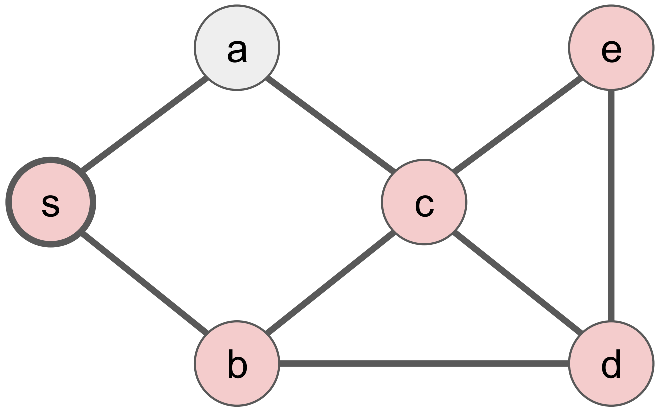
v = c

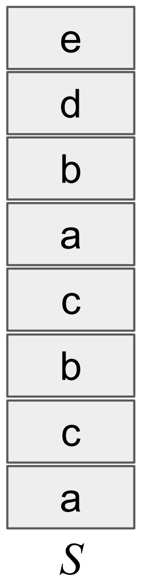
Depth-first search (DFS)

DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

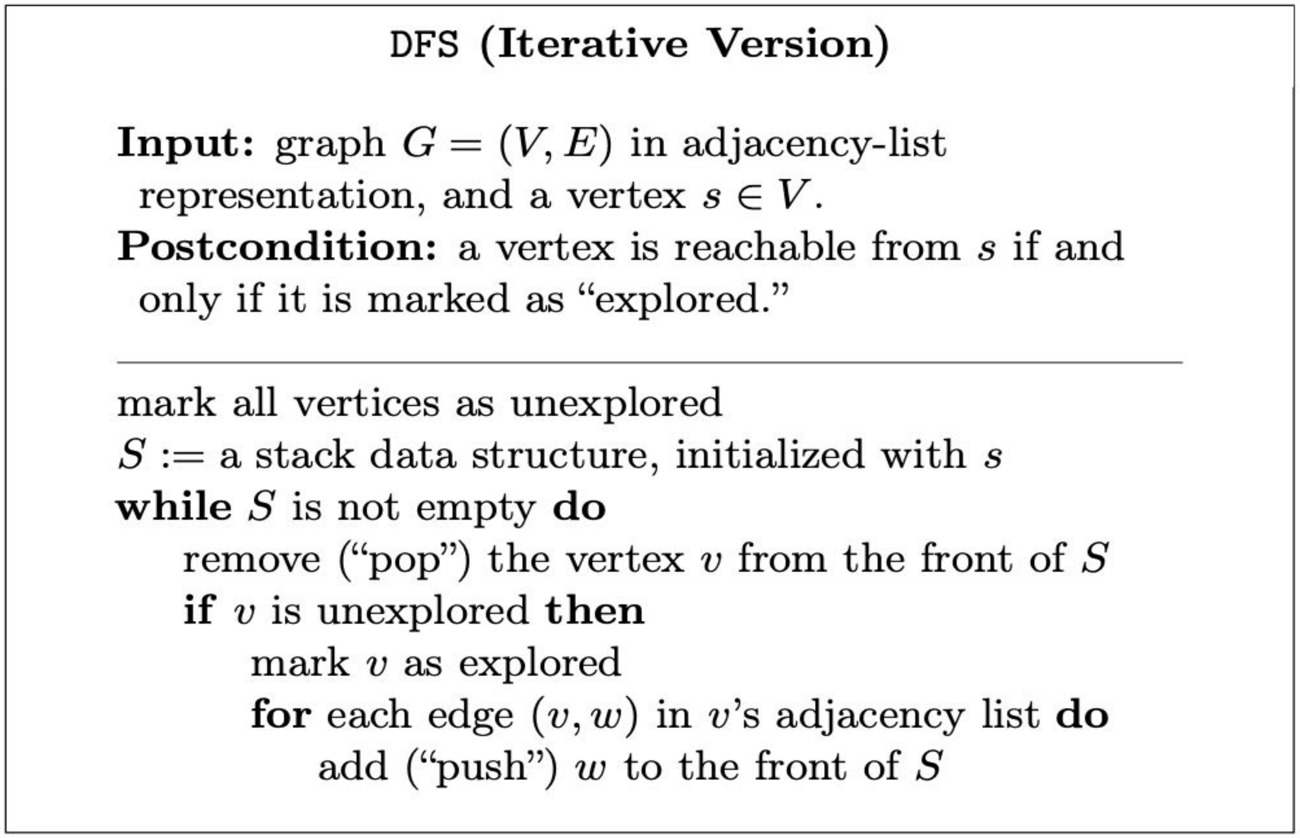




Explored:
s, b, d, e,
c

v = c

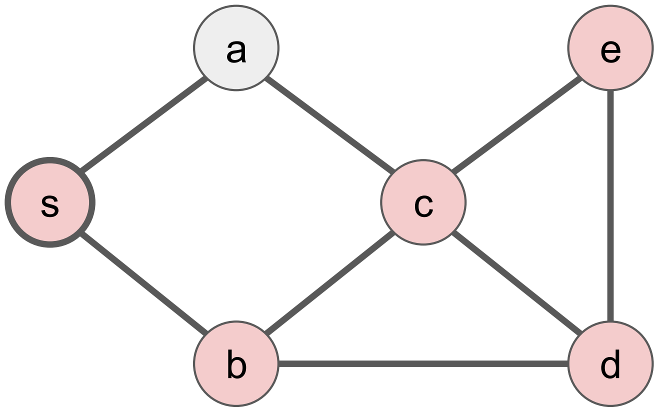
Depth-first search (DFS)

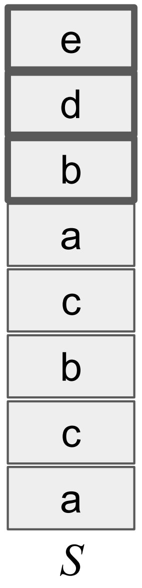


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

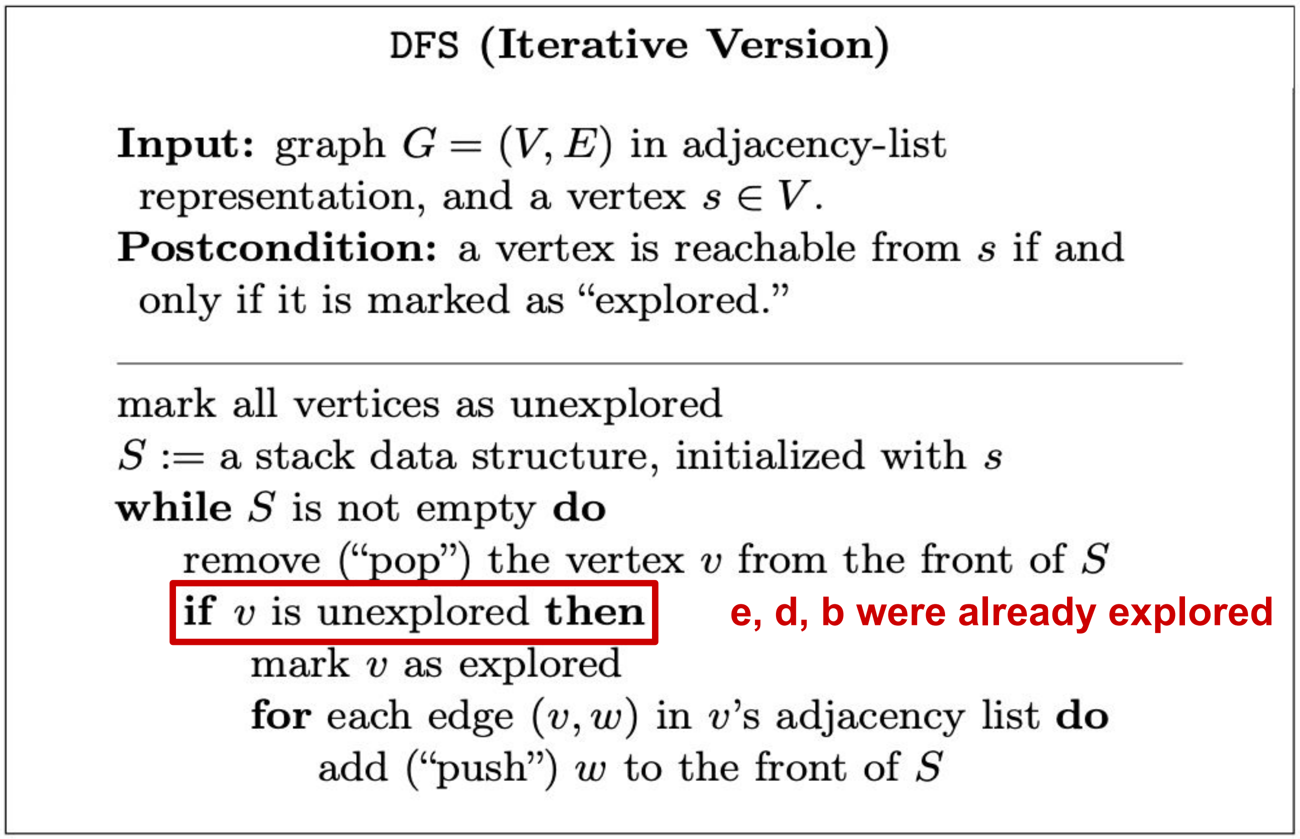




Explored:
s, b, d, e,
c

v = c

Depth-first search (DFS)

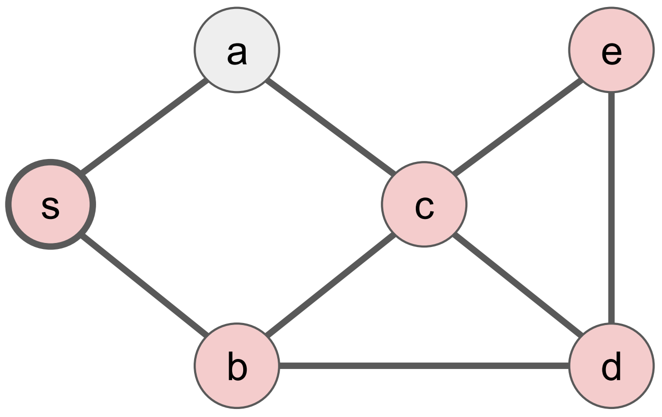


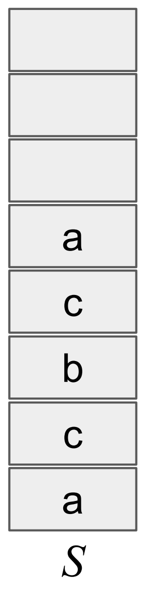
DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

e, d, b were already explored

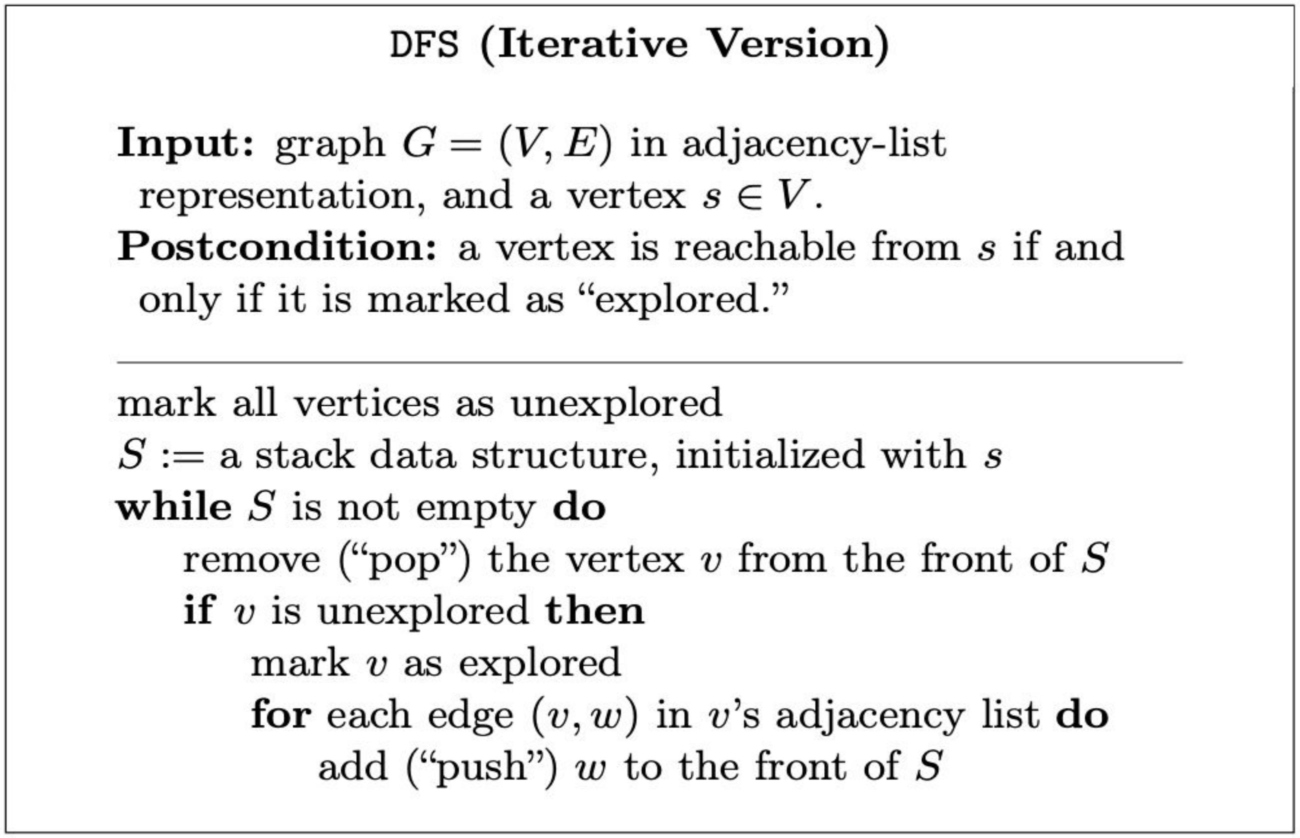




Explored:
s, b, d, e,
c

$v =$
e…d…b

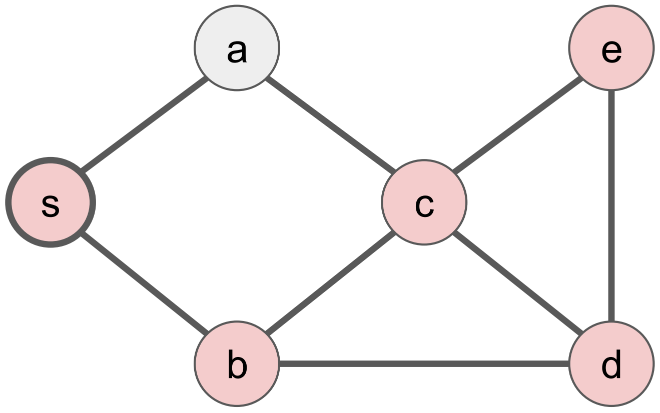
Depth-first search (DFS)

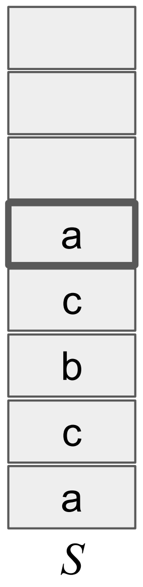


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

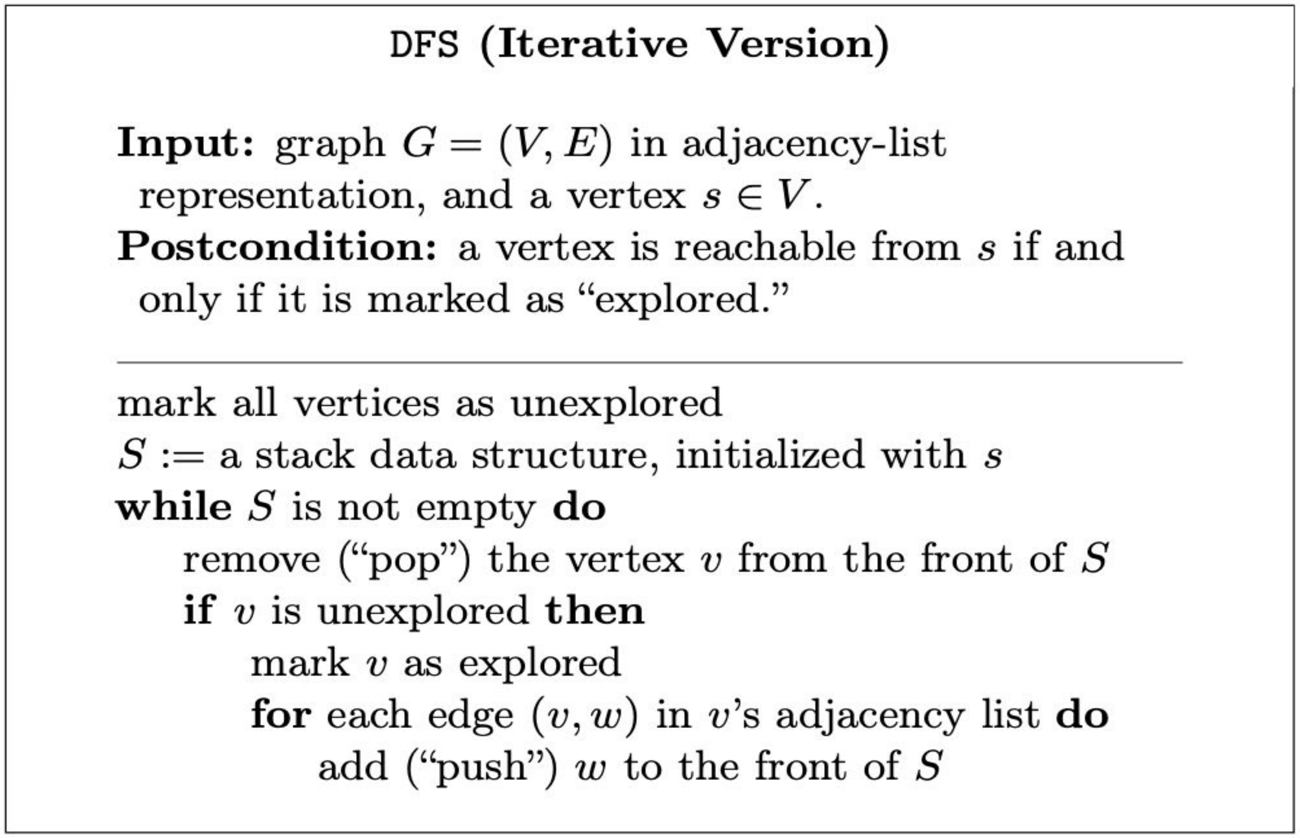




Explored:
s, b, d, e,
c

$v =$
e…d…b

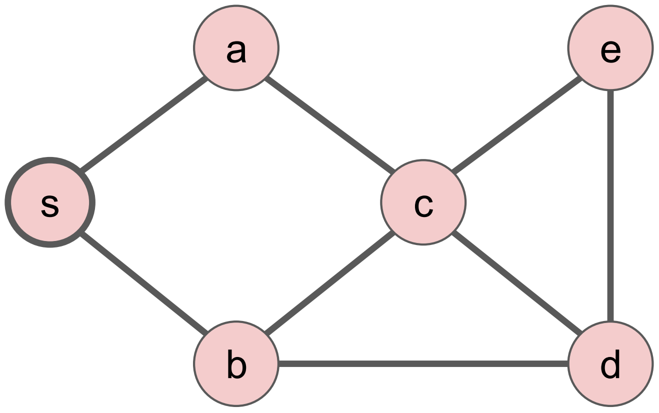
Depth-first search (DFS)

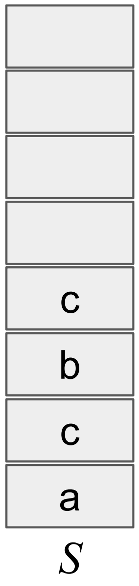


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

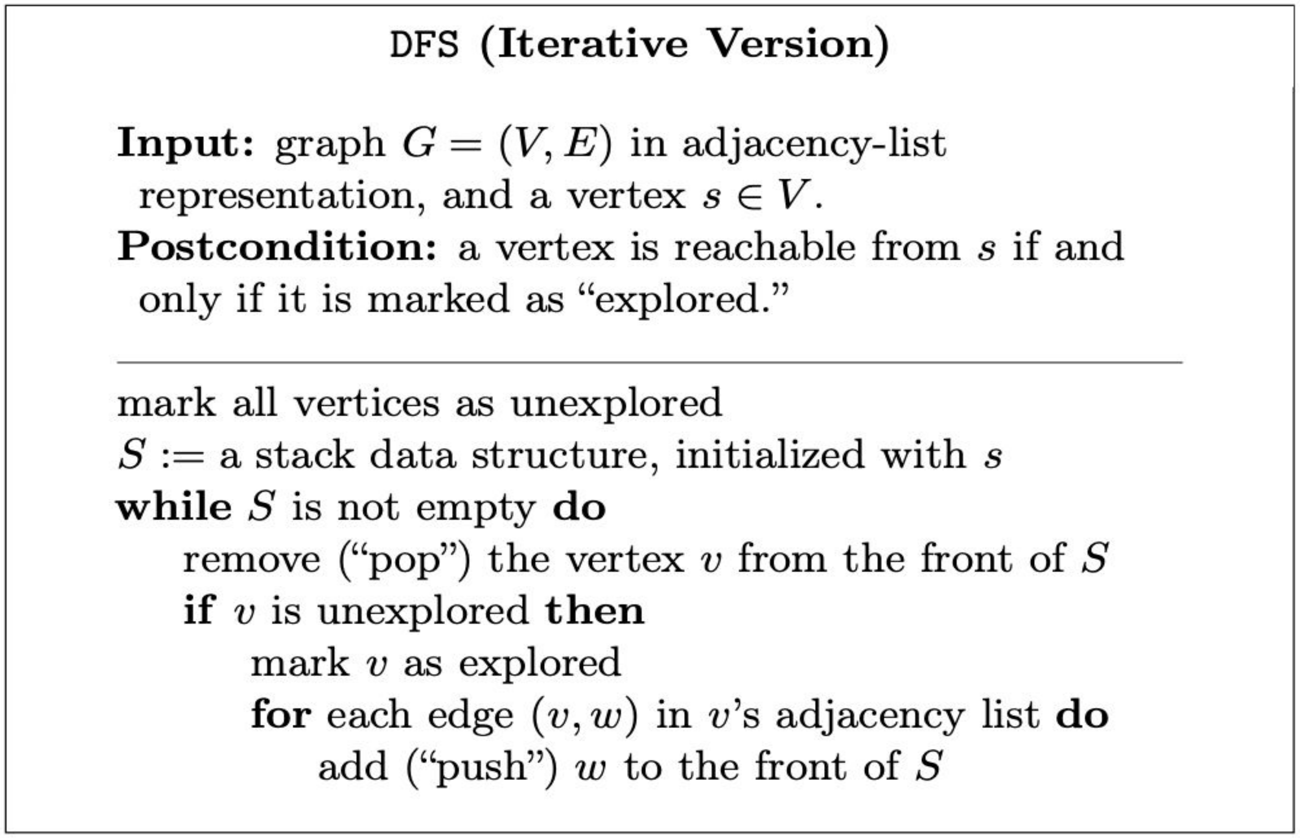




Explored:
s, b, d, e,
c, a

$v =$ a

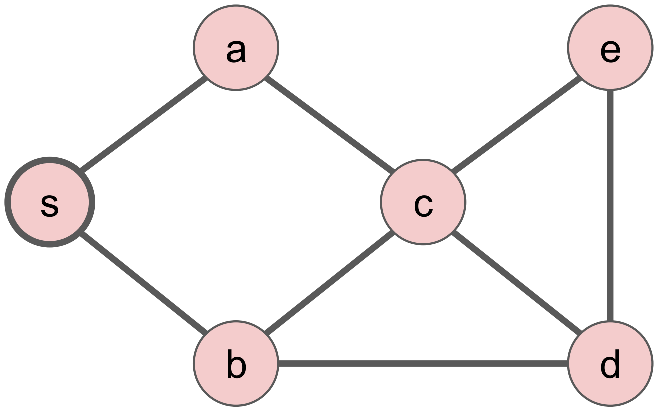
Depth-first search (DFS)

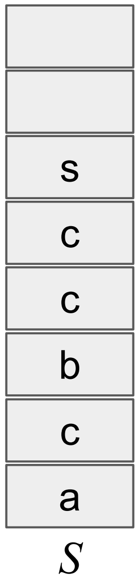


DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

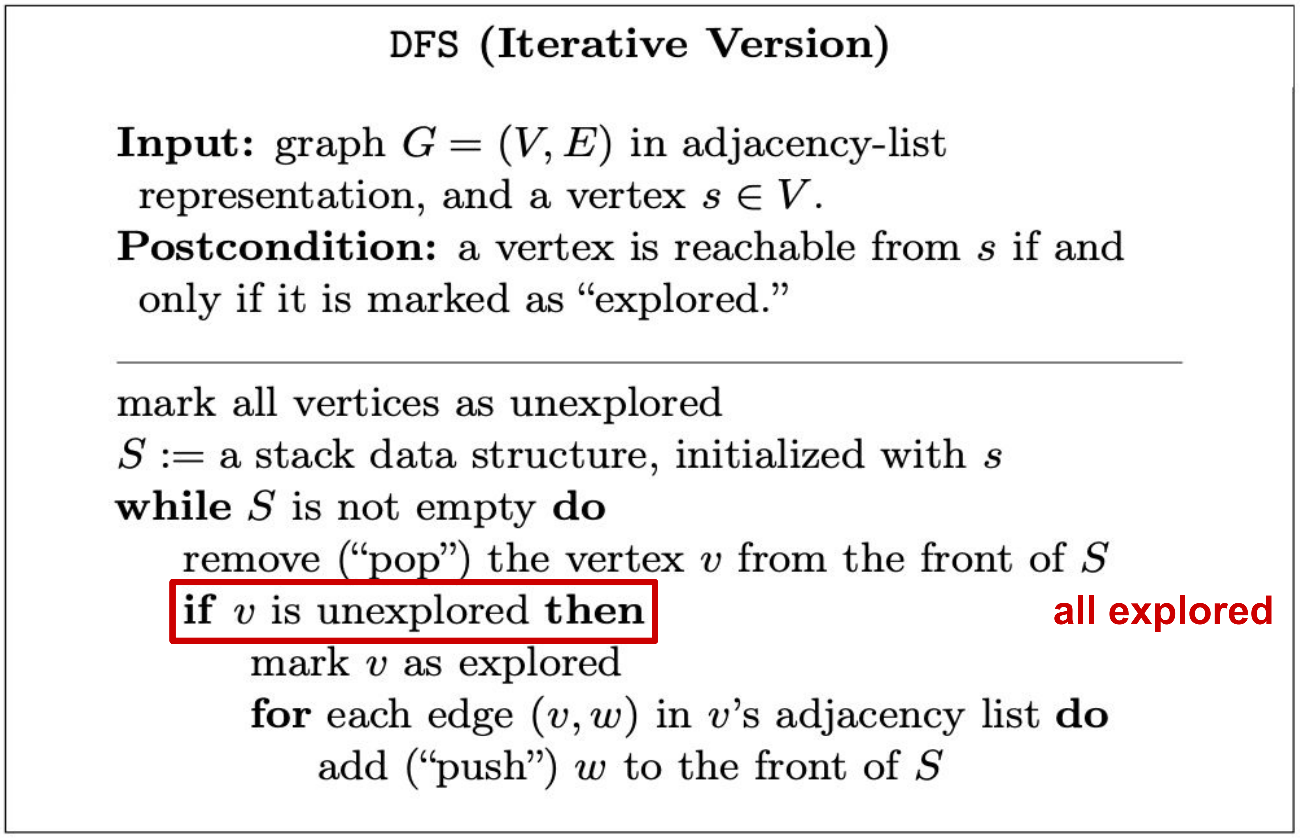




Explored:
s, b, d, e,
c, a

v = a

Depth-first search (DFS)

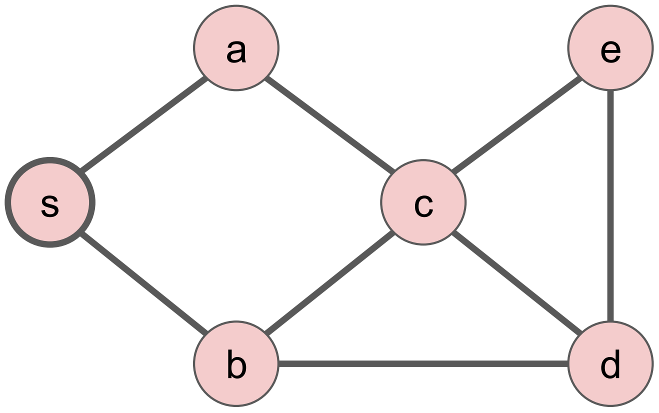


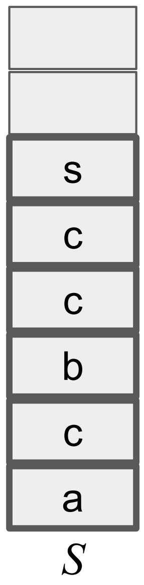
DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```

all explored

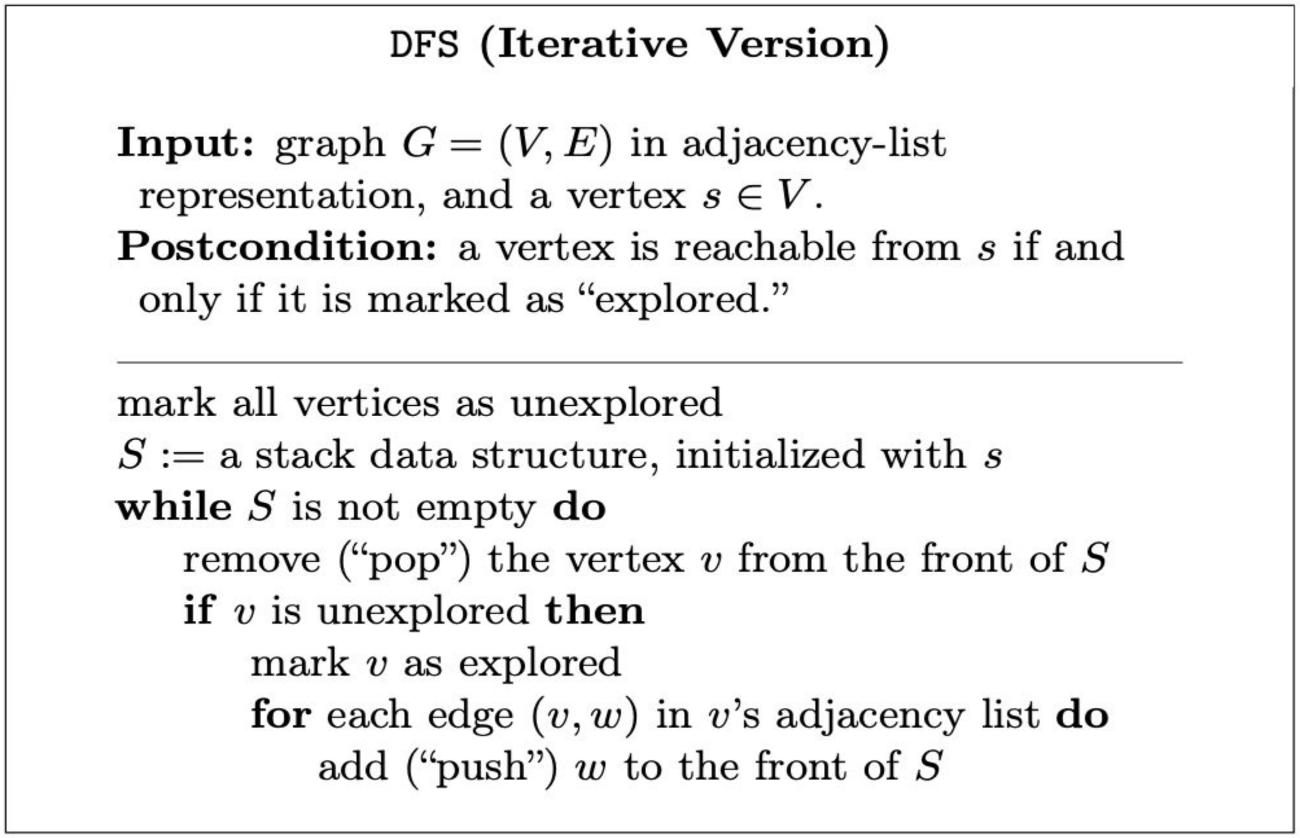




Explored:
s, b, d, e,
c, a

v = each
at a time

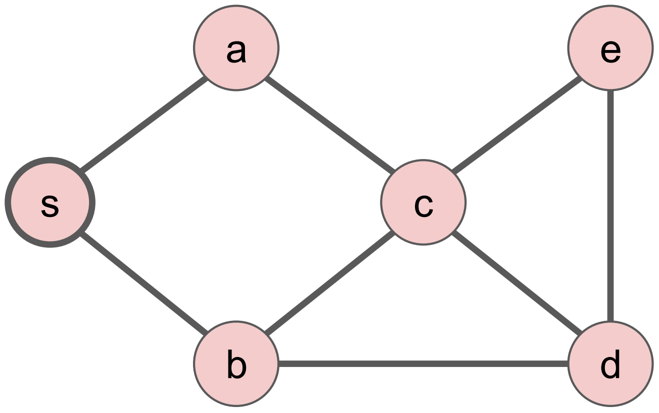
Depth-first search (DFS)



DFS (Iterative Version)

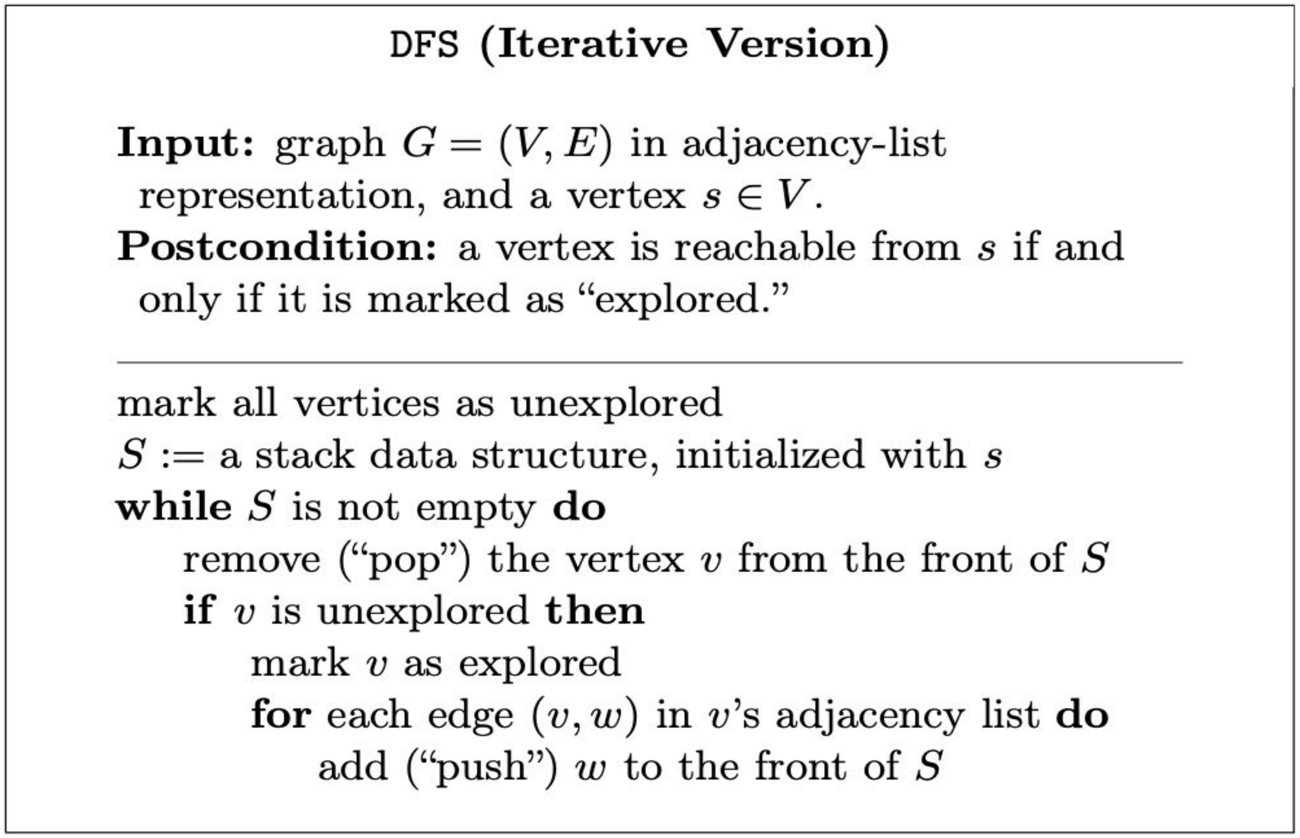
Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```



Explored:
s, b, d, e,
c, a

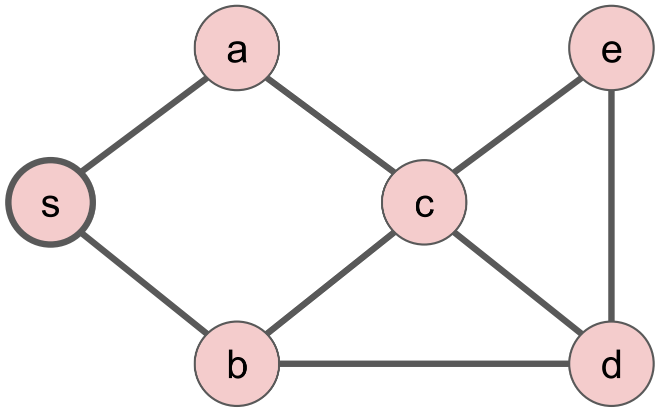
Depth-first search (DFS)



DFS (Iterative Version)

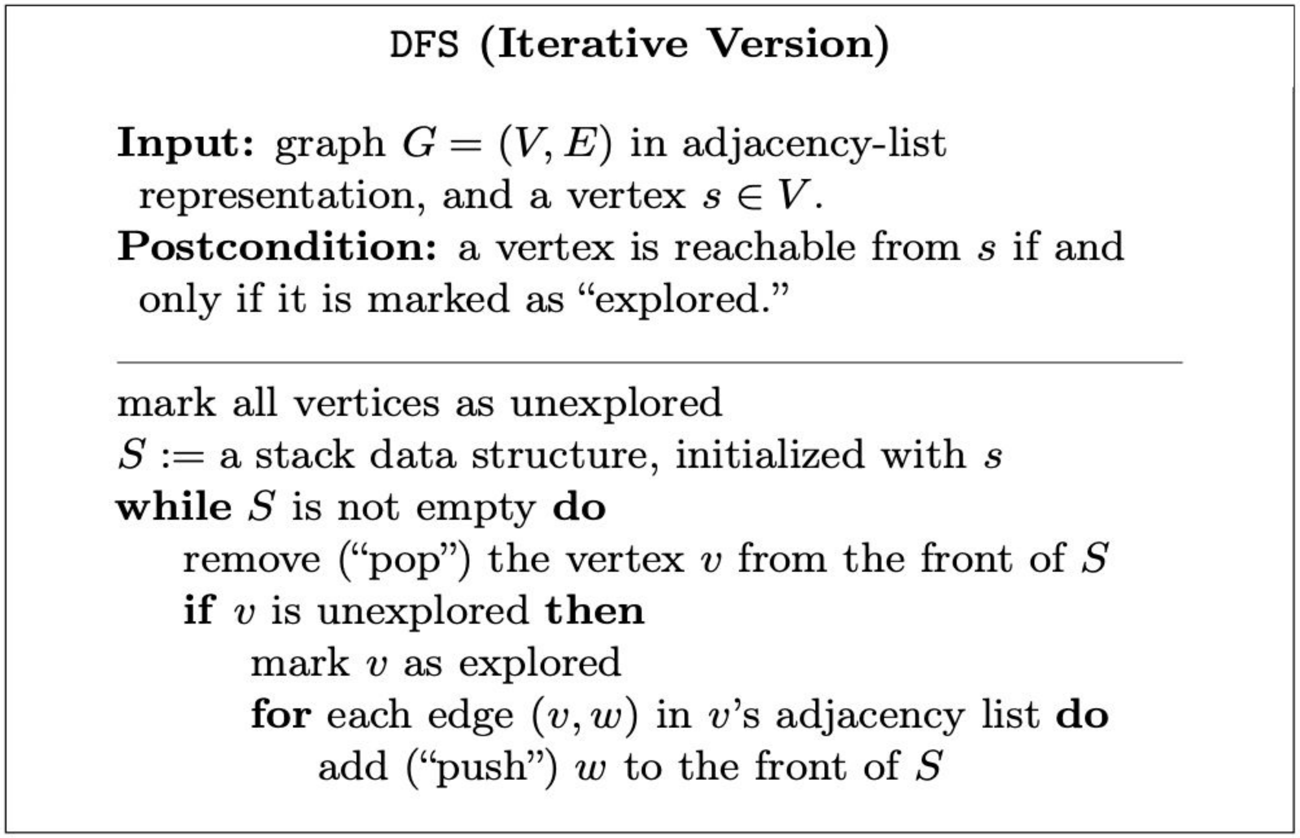
Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```



Time complexity?

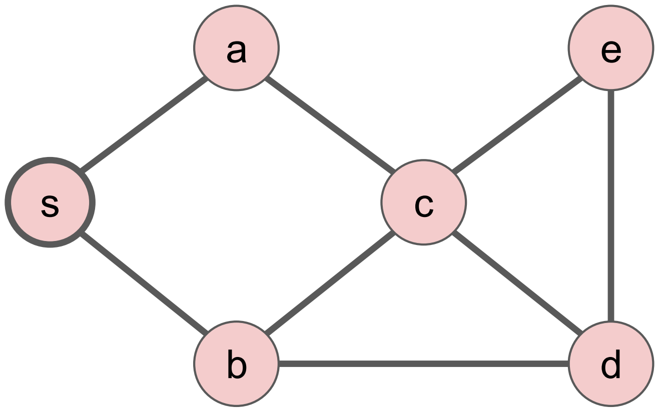
Depth-first search (DFS)



DFS (Iterative Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

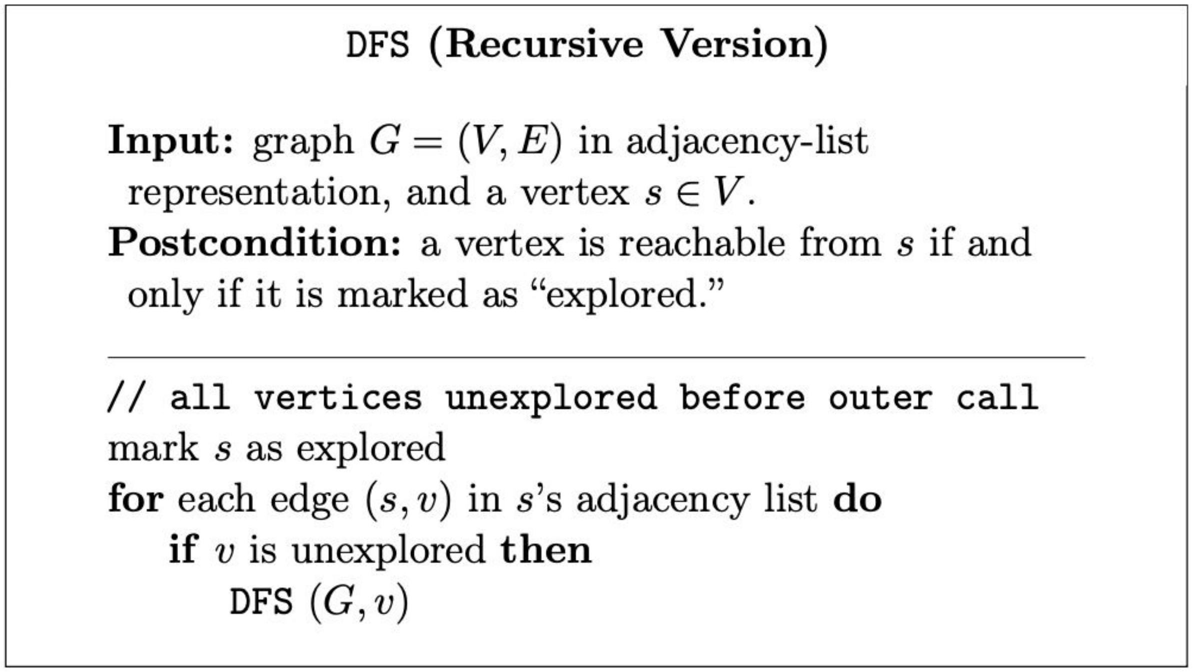
```
mark all vertices as unexplored
$S$ := a stack data structure, initialized with $s$
while $S$ is not empty do
    remove ("pop") the vertex $v$ from the front of $S$
    if $v$ is unexplored then
        mark $v$ as explored
        for each edge $(v, w)$ in $v$'s adjacency list do
            add ("push") $w$ to the front of $S$
```



Time complexity
O(V + E)

Depth-first search (DFS) without the stack?
How do you think it is possible?

Recursive depth-first search (DFS)



DFS (Recursive Version)

Input: graph $G = (V, E)$ in adjacency-list representation, and a vertex $s \in V$.
Postcondition: a vertex is reachable from s if and only if it is marked as "explored."

```
// all vertices unexplored before outer call
mark $s$ as explored
for each edge $(s, v)$ in $s$'s adjacency list do
    if $v$ is unexplored then
        DFS $(G, v)$
```