Do Now Exercise

To prepare you for the lecture today, please do the following exercise.

Try searching (walking) routes

from Halligan Hall

to Museum of Fine Arts, Boston

using Google Maps.

COMP15: Data Structures

Week 10, Summer 2019

Admin

T9: (Please check the course page for details)

Due by 6pm on **Tuesday, July 30**

T9: In-class Presentation

on Week 11 (Wednesday, July 31)

Questions about T9 and the presentation?

Feedbacks on **P4** from Matt

P5: **Word Frequency Database**

Project Due by 6pm on Sunday, August 4

Questions about P5?

Hash Tables

The running time of
put(), get(), remove() performed on a hash table using chaining?

Worst-case: $O(n)$
Best-case: $O(1)$
(Average-case: $O(1)$)

(Note: Plus the cost of generating a hash code and of the compression operation.
Worst-case: With an assumption that the put operation checks duplicates.
Average-case: With an assumption that a "good" hash function is used.)

Load factor and rehashing

Discussion:
What happens when the load factor is high?

Other ways to handling collisions

A sneak peek preview (Comp 160, Algorithms)

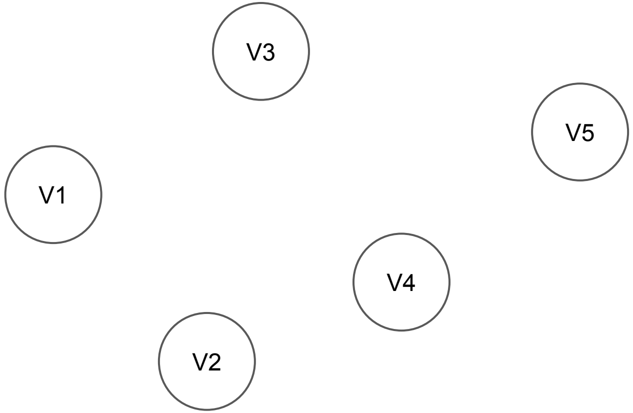
Open addressing

- Linear probing
- Quadratic probing
- Double hashing

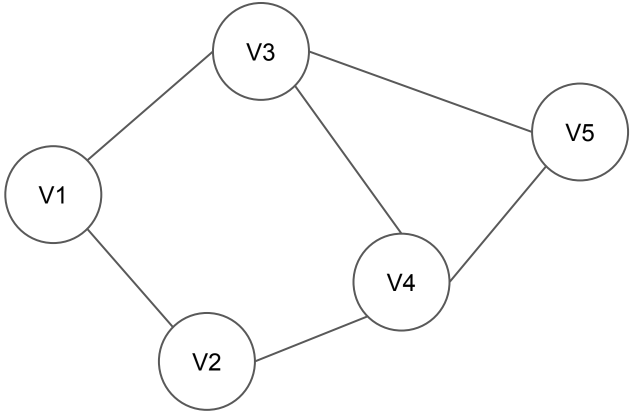
Graph

Some Terminologies

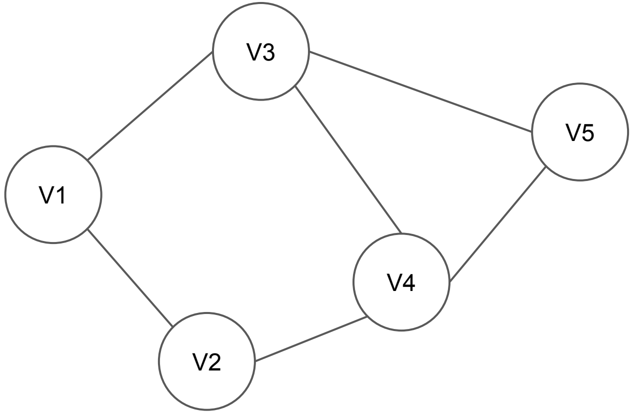
Vertices (Nodes)



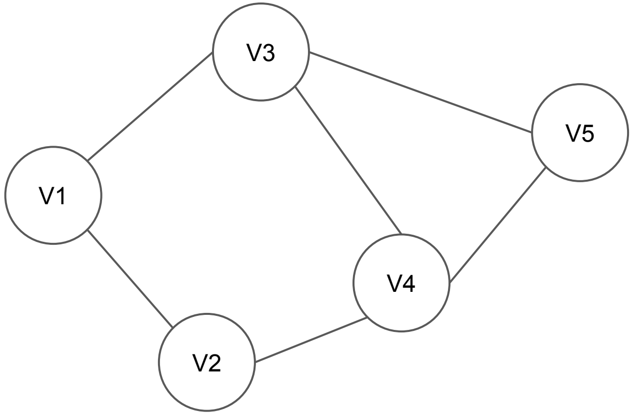
Edges



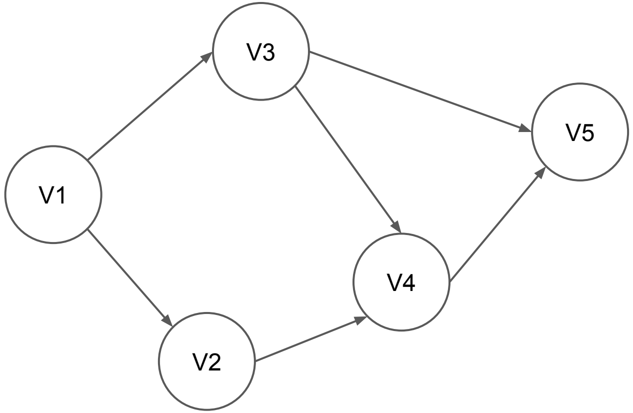
Undirected Edges



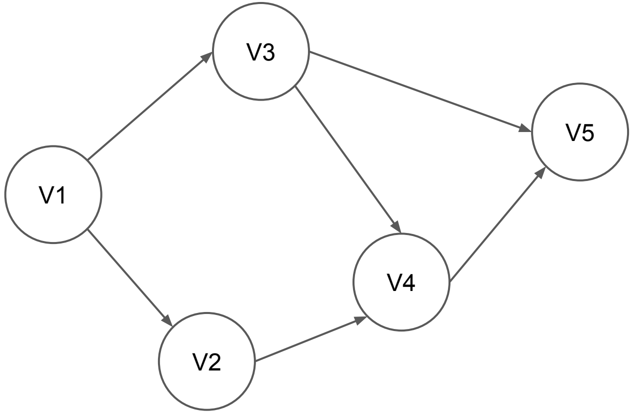
Undirected Graph



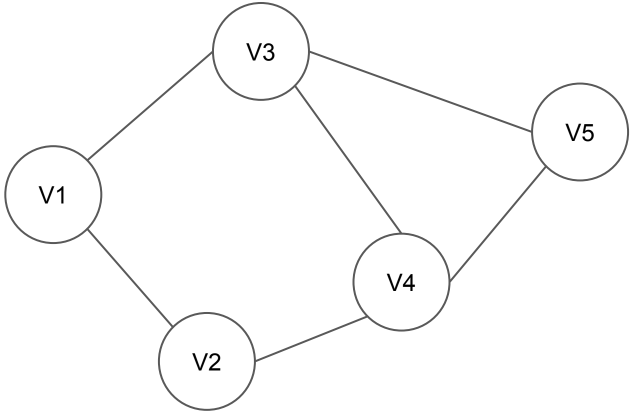
Directed Edges



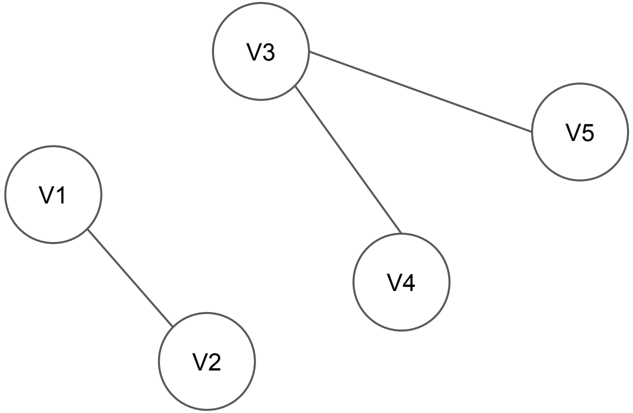
Directed Graph (Digraph)



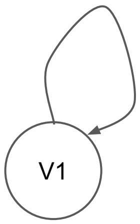
Connected Graph



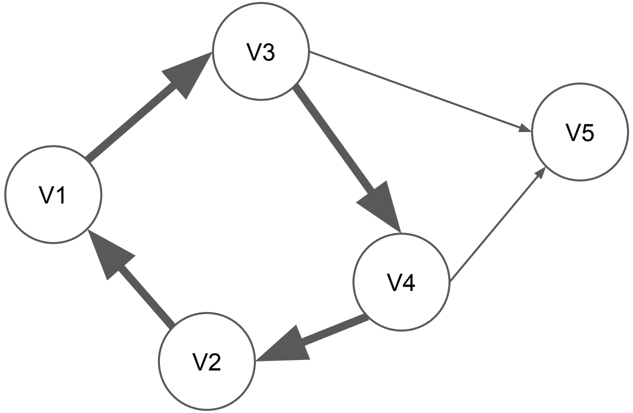
Disconnected Graph



Self loop edge

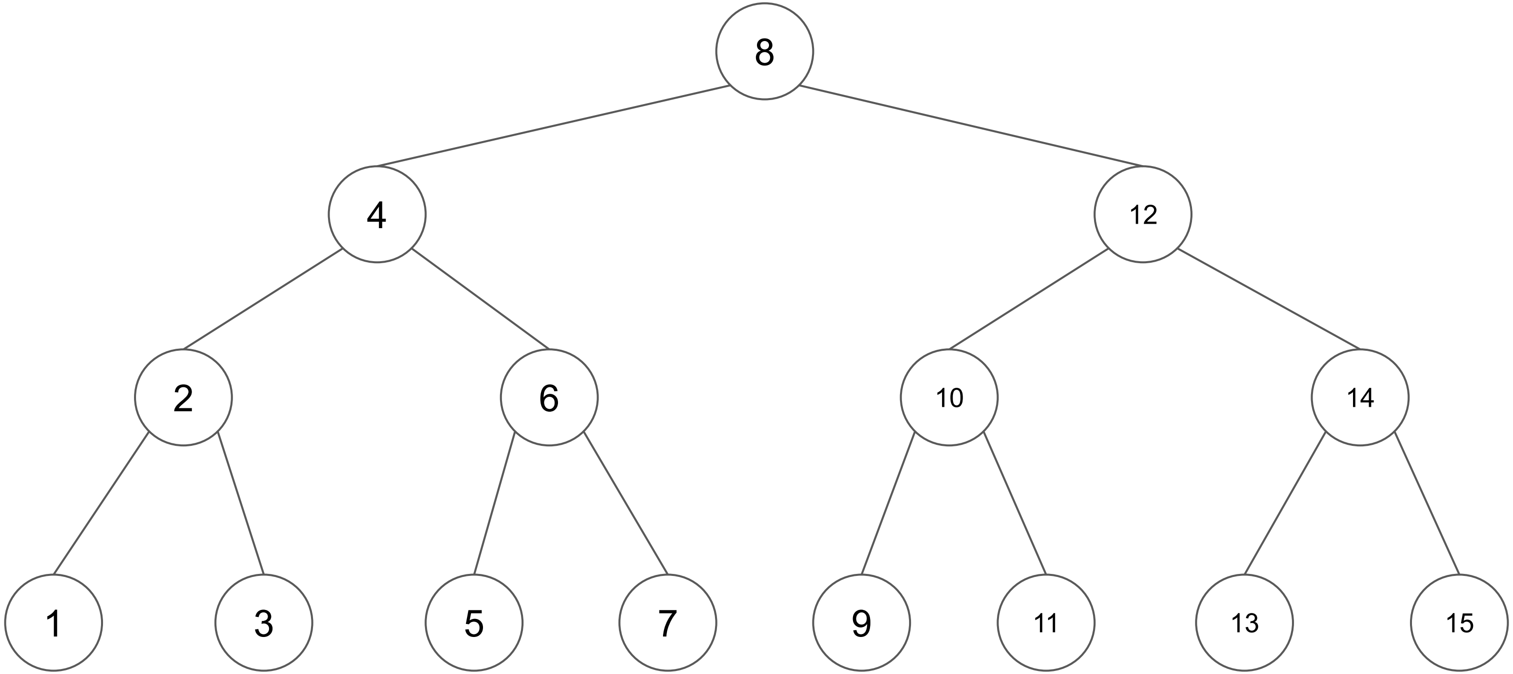


Cycle



- Parallel edges
- Strongly connected directed graph
- Weakly connected directed graph

Tree



Some notations

$G = (V, E)$ (a graph, a set of vertices and a set of edges)

$|V|$ (the number of vertices)

$|E|$ (the number of edges)

(v_1, v_2) (an edge from v1 to v2; (or between v1 and v2))

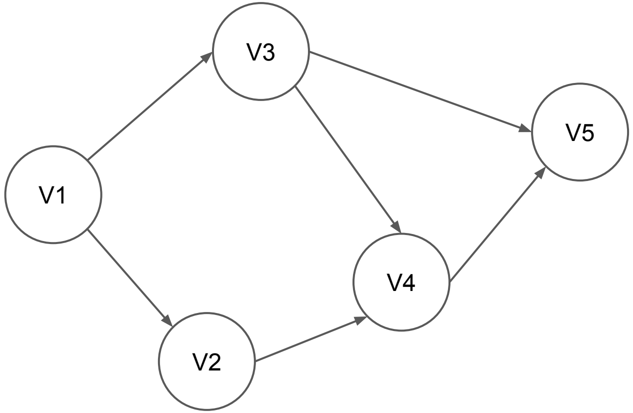
$w(v_1, v_2)$ (the weight of an edge from v1 to v2; (or between v1 and v2))

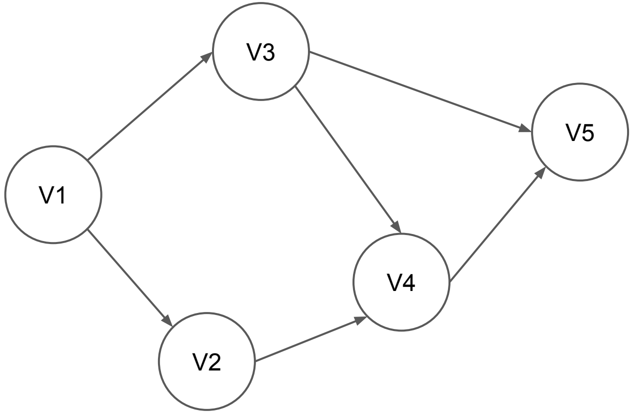
Graph

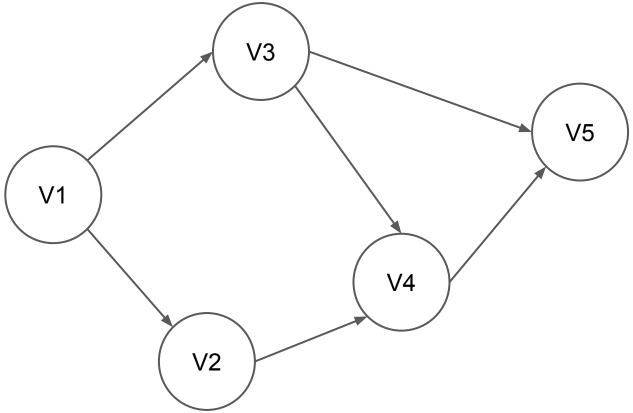
- add/remove vertex
- add/remove edge
- has edge between two vertices
- etc.

Representing a graph

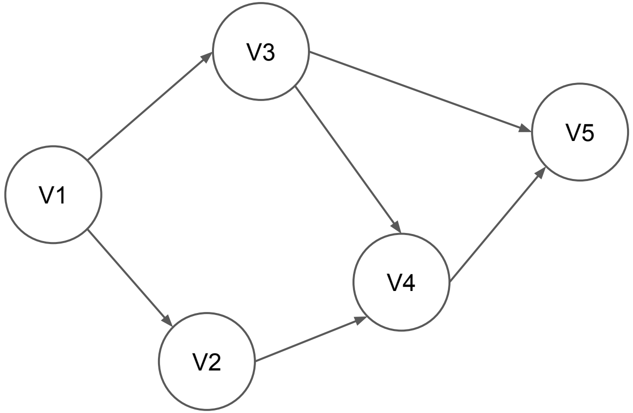
Adjacency Matrix


V1
V2
V3
V4
V5

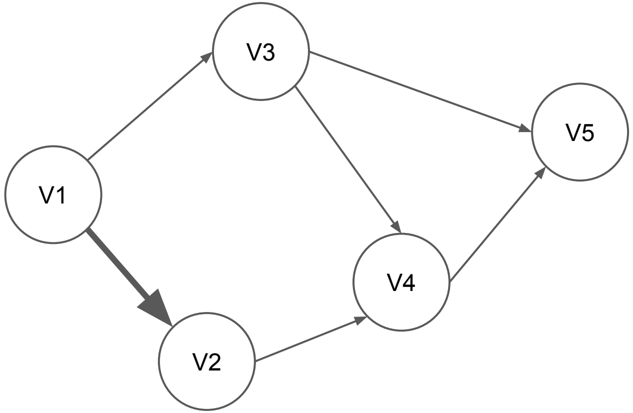

V3
V1
V5
V4
V2



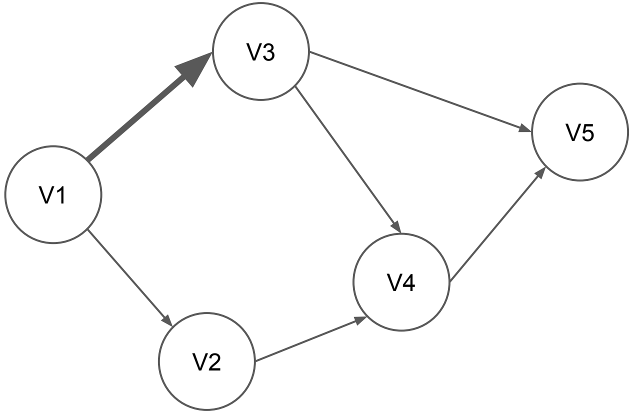
	V1	V2	V3	V4	V5
V1					
V2					
V3					
V4					
V5					



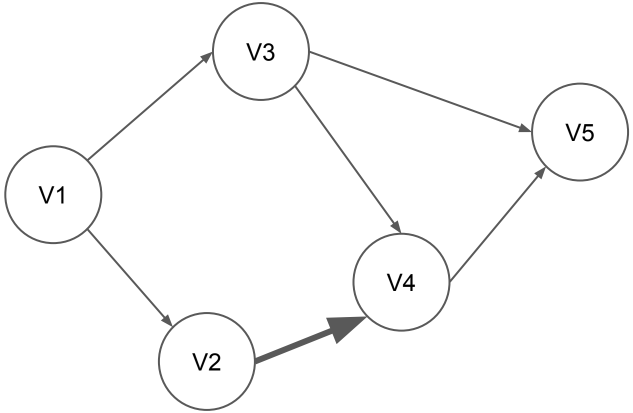
	V1	V2	V3	V4	V5
V1	0	0	0	0	0
V2	0	0	0	0	0
V3	0	0	0	0	0
V4	0	0	0	0	0
V5	0	0	0	0	0



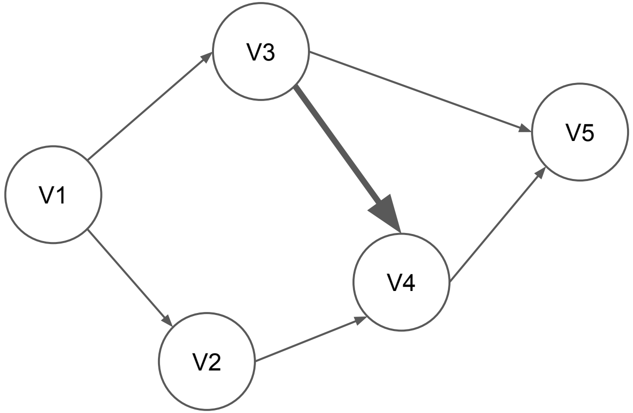
	V1	V2	V3	V4	V5
V1	0	**1**	0	0	0
V2	0	0	0	0	0
V3	0	0	0	0	0
V4	0	0	0	0	0
V5	0	0	0	0	0



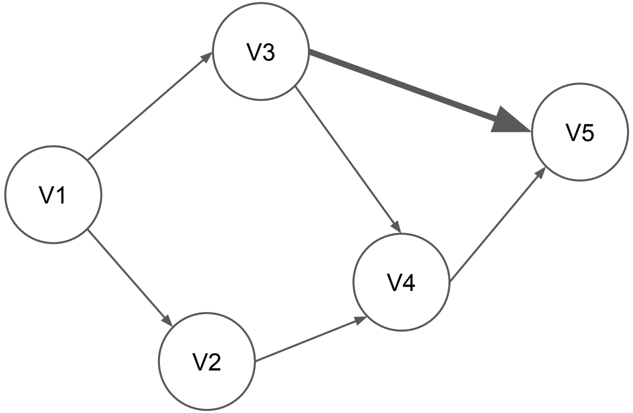
	V1	V2	V3	V4	V5
V1	0	1	**1**	0	0
V2	0	0	0	0	0
V3	0	0	0	0	0
V4	0	0	0	0	0
V5	0	0	0	0	0



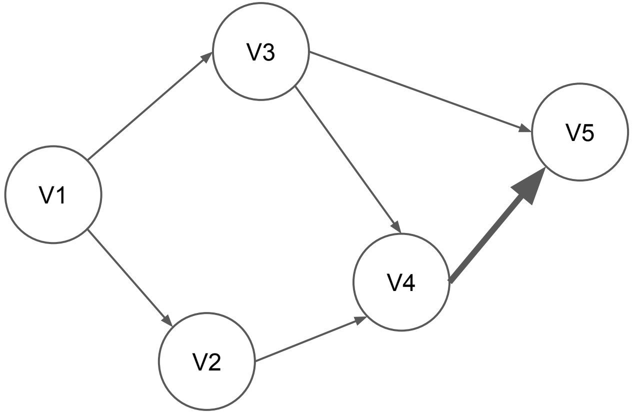
	V1	V2	V3	V4	V5
V1	0	1	1	0	0
V2	0	0	0	**1**	0
V3	0	0	0	0	0
V4	0	0	0	0	0
V5	0	0	0	0	0



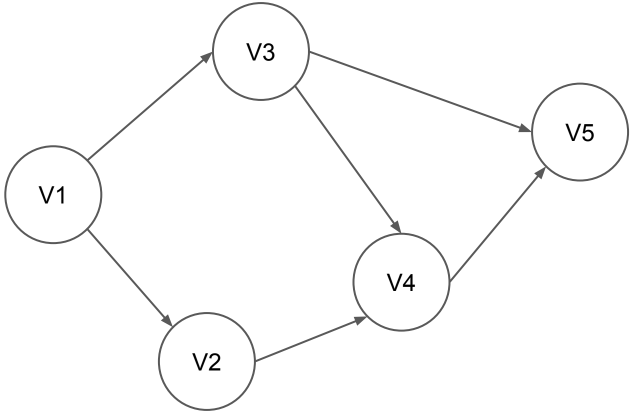
	V1	V2	V3	V4	V5
V1	0	1	1	0	0
V2	0	0	0	1	0
V3	0	0	0	**1**	0
V4	0	0	0	0	0
V5	0	0	0	0	0



	V1	V2	V3	V4	V5
V1	0	1	1	0	0
V2	0	0	0	1	0
V3	0	0	0	1	**1**
V4	0	0	0	0	0
V5	0	0	0	0	0

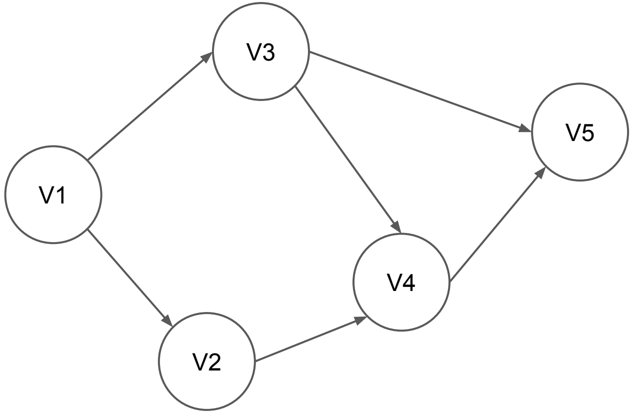


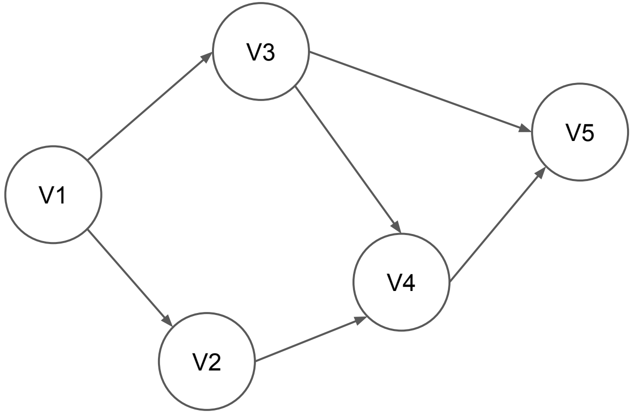
	V1	V2	V3	V4	V5
V1	0	1	1	0	0
V2	0	0	0	1	0
V3	0	0	0	1	1
V4	0	0	0	0	**1**
V5	0	0	0	0	0



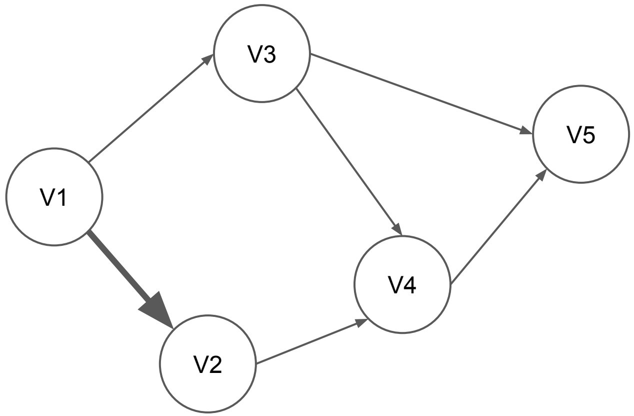
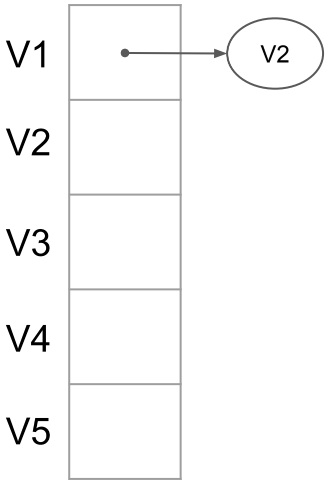
	V1	V2	V3	V4	V5
V1	0	1	1	0	0
V2	0	0	0	1	0
V3	0	0	0	1	1
V4	0	0	0	0	1
V5	0	0	0	0	0

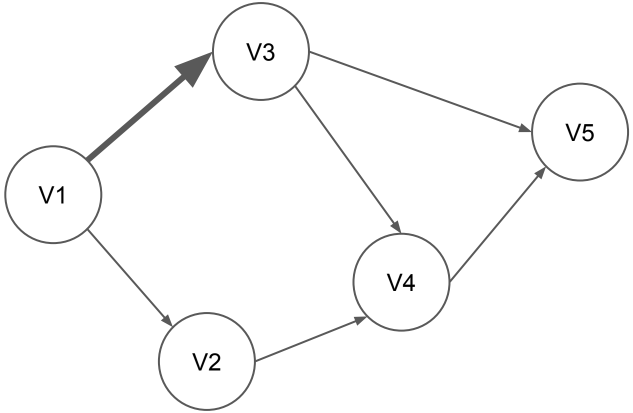
Adjacency List

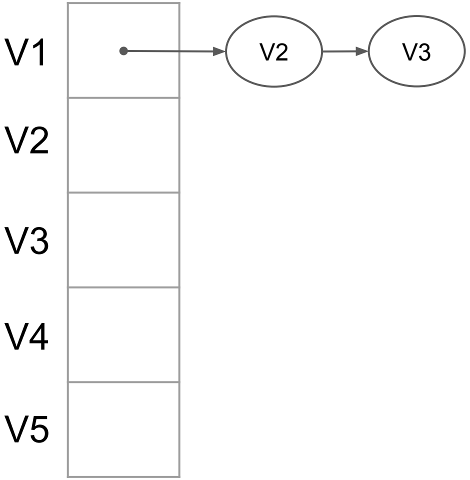

V3
V5
V1
V4
V2

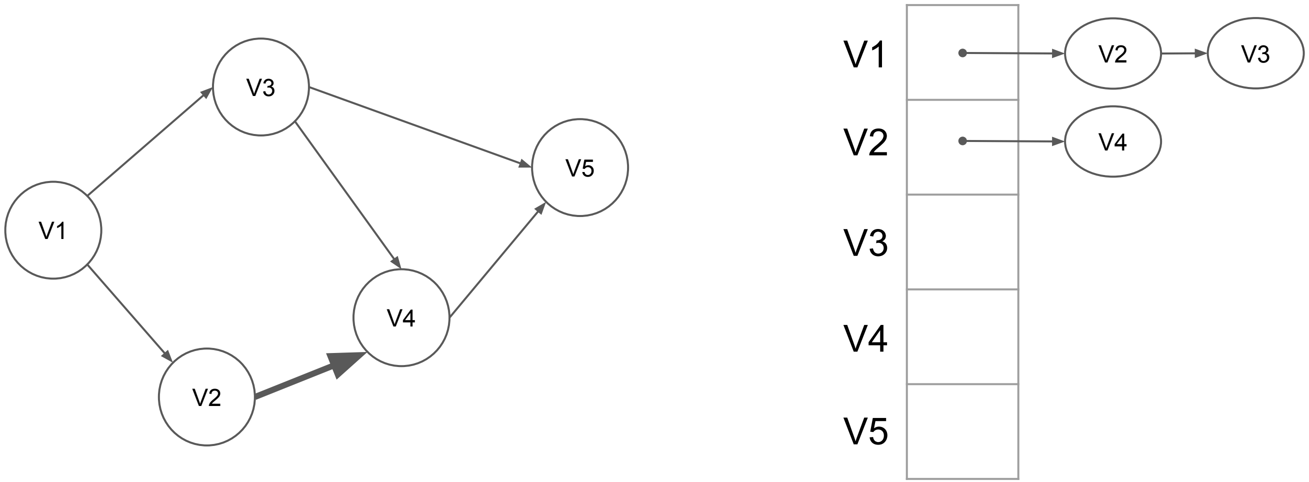


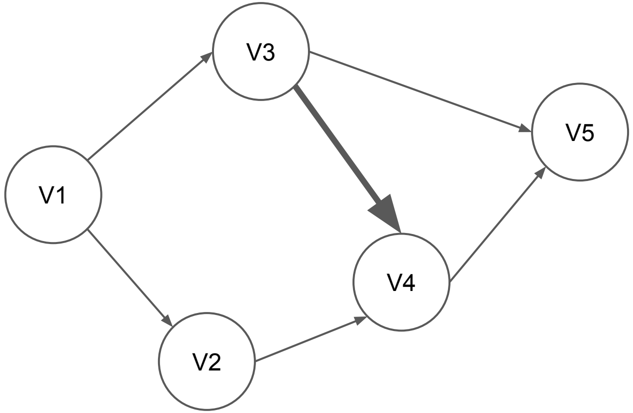
V1	
V2	
V3	
V4	
V5	

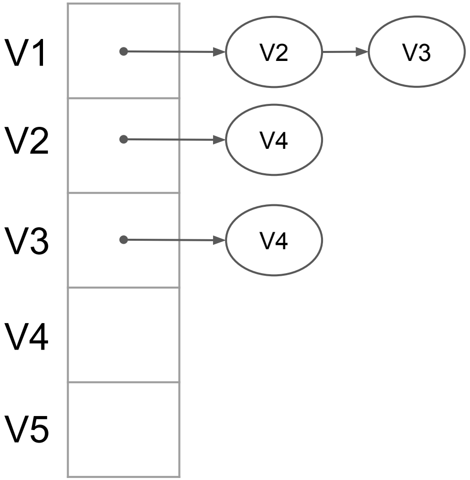

V1
V2
V3
V4
V5

V1
V2
V3
V4
V5
V2

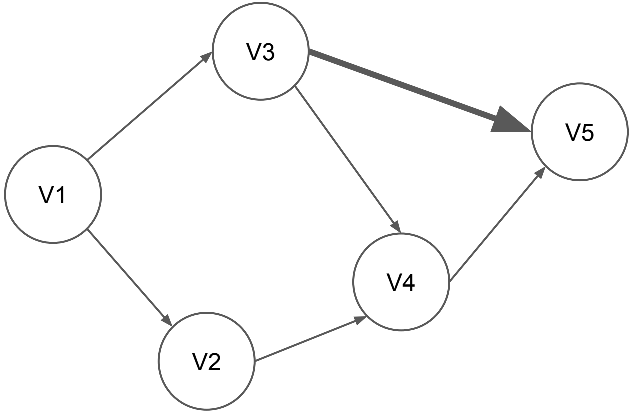

V1
V2
V3
V4
V5

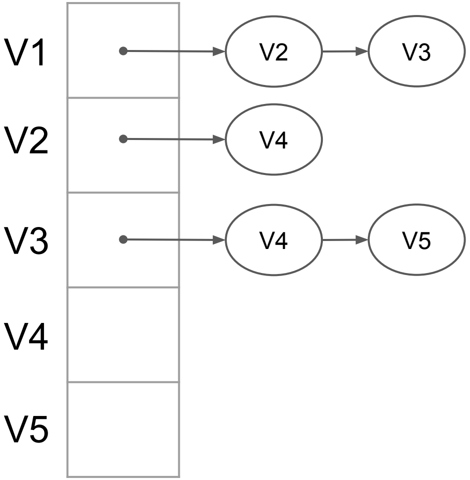

V1
V2
V3
V4
V5
V2
V3

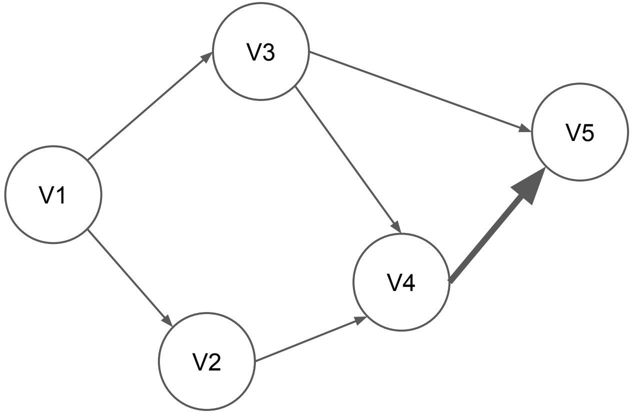

V1
V2
V3
V4
V5
V1
V2
V3
V2
V4
V3
V4
V5

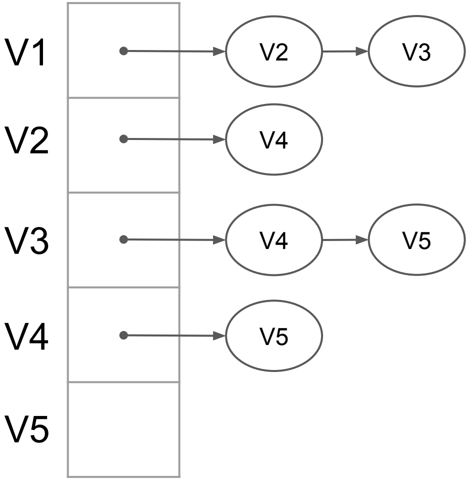

V1
V2
V3
V4
V5

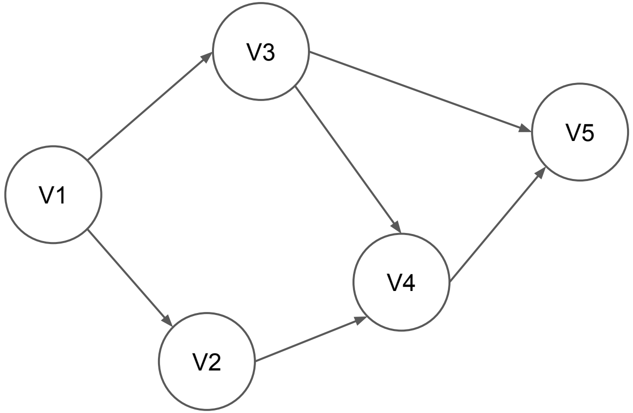

V1
V2
V3
V2
V4
V3
V4
V4
V5

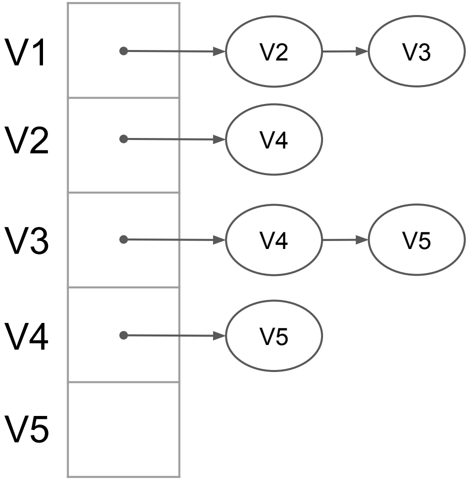

V1
V2
V3
V4
V5


V1
V2
V3
V2
V4
V3
V4
V5
V4
V5


V1
V2
V3
V4
V5


V1
V2
V3
V2
V4
V3
V4
V5
V4
V5
V5


V1
V2
V3
V4
V5

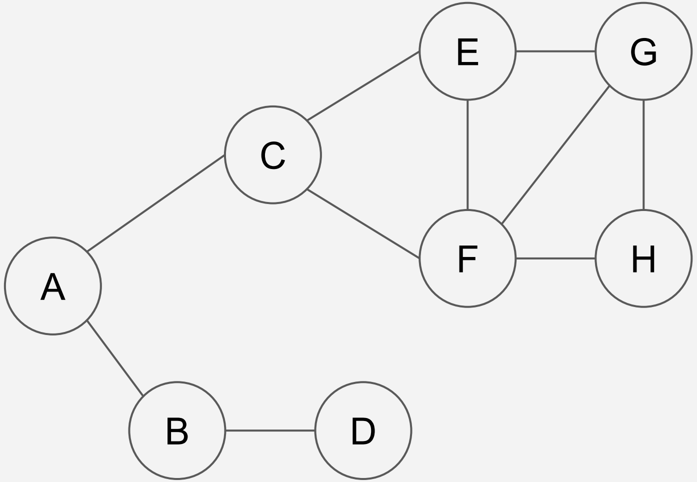

V1
V2
V3
V2
V4
V3
V4
V5
V4
V5
V5

Discussion:
What is the worst-case running time of add/remove/has edge operations performed on each of the adjacency matrix and the adjacency list?

Discussion:
How much (memory) space does each of the adjacency matrix and the adjacency list need?

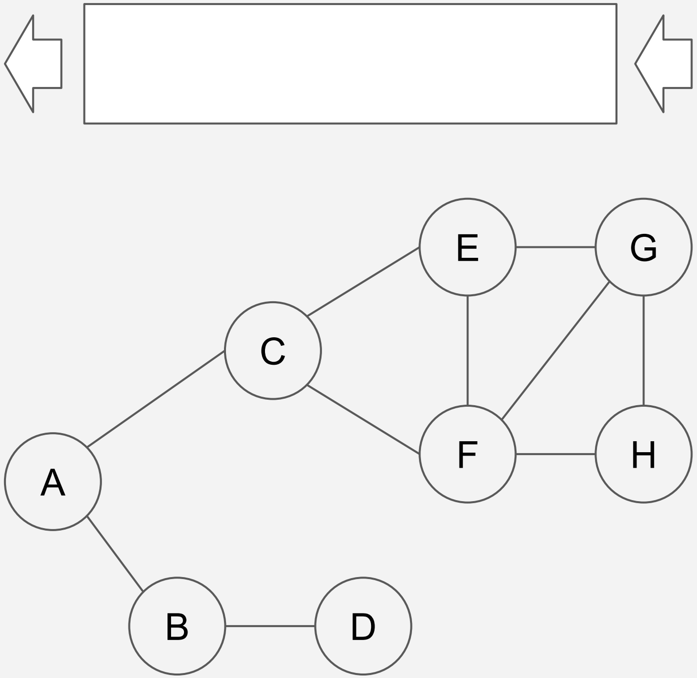
Breadth-first search (BFS)

Breadth-first search (BFS)



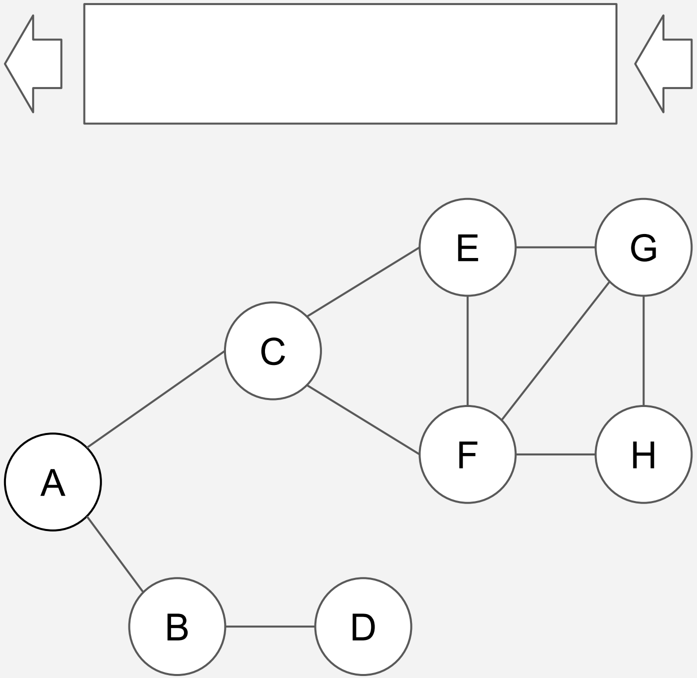
Breadth-first search (BFS)

(using a queue)



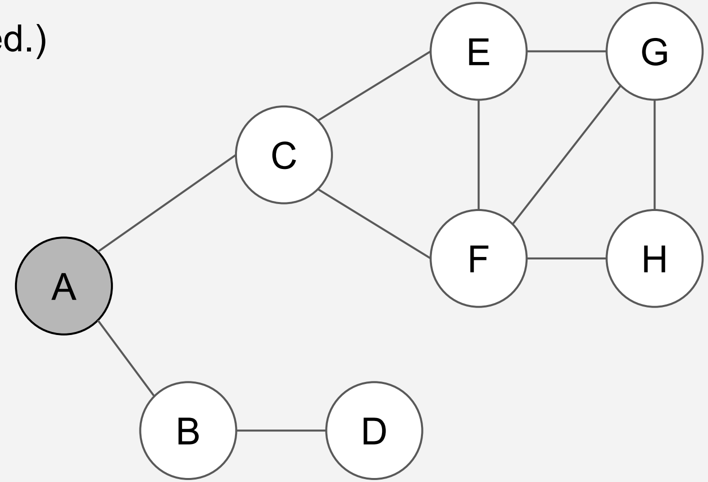
Breadth-first search (BFS)

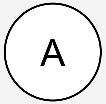
(color all vertices White)



Breadth-first search (BFS)

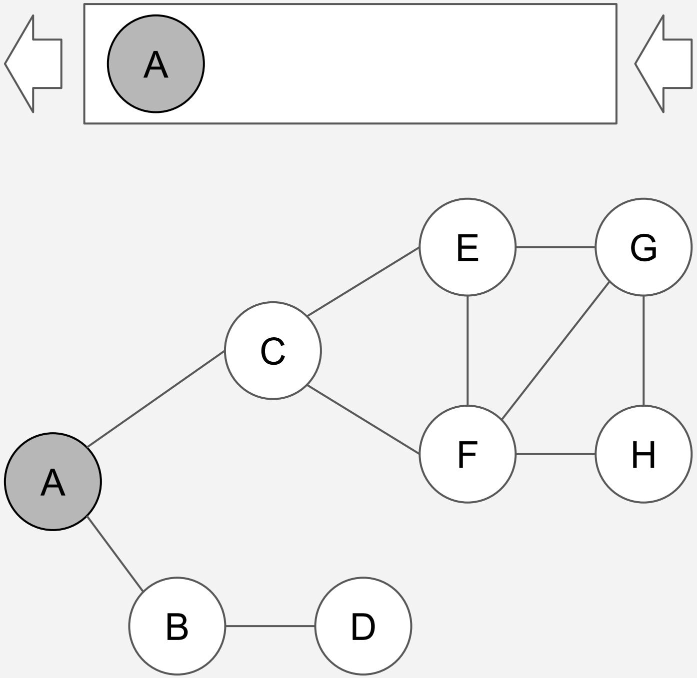
(start from A;
color A Gray; A is discovered.)

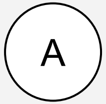




Breadth-first search (BFS)

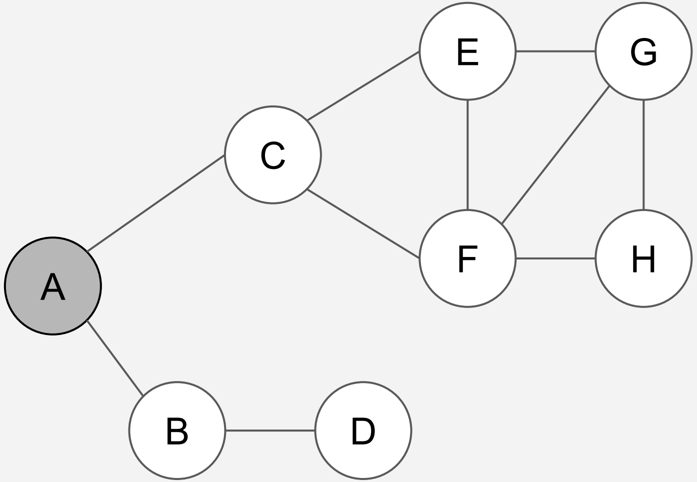
(enqueue A)

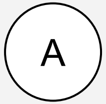




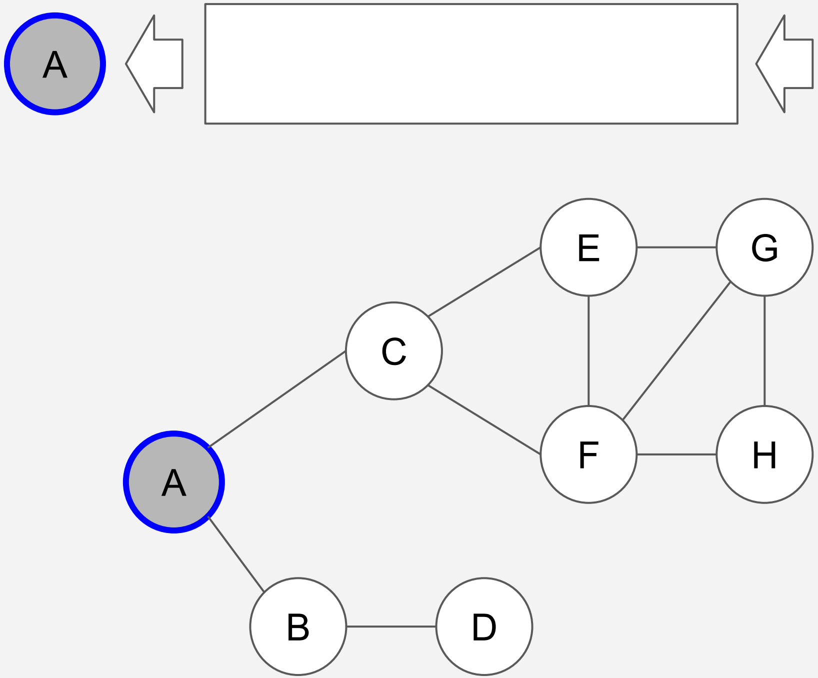
Breadth-first search (BFS)

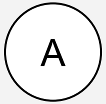
(front and dequeue A)





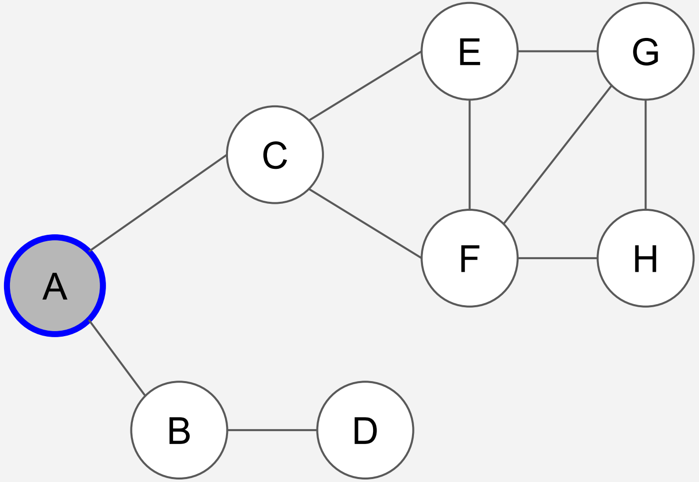
Breadth-first search (BFS)

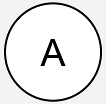




Breadth-first search (BFS)

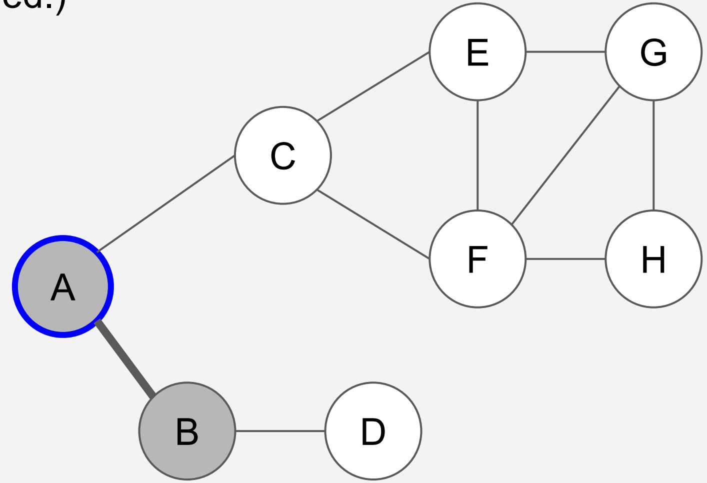
(explore edges from A)

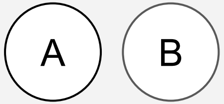




Breadth-first search (BFS)

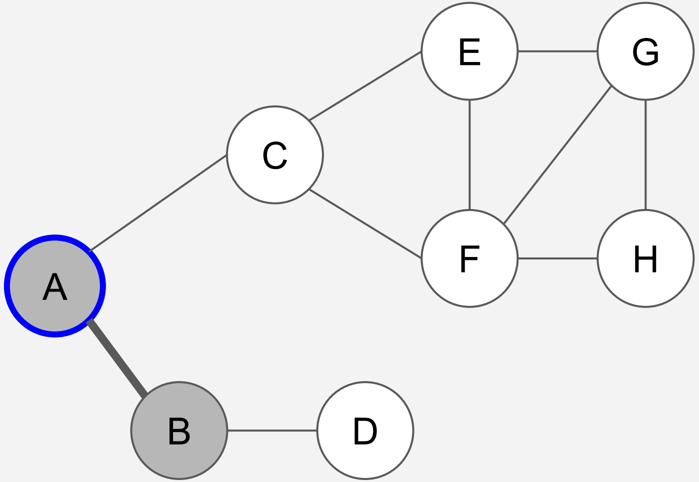
(color B Gray; B is discovered.)

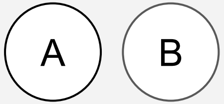




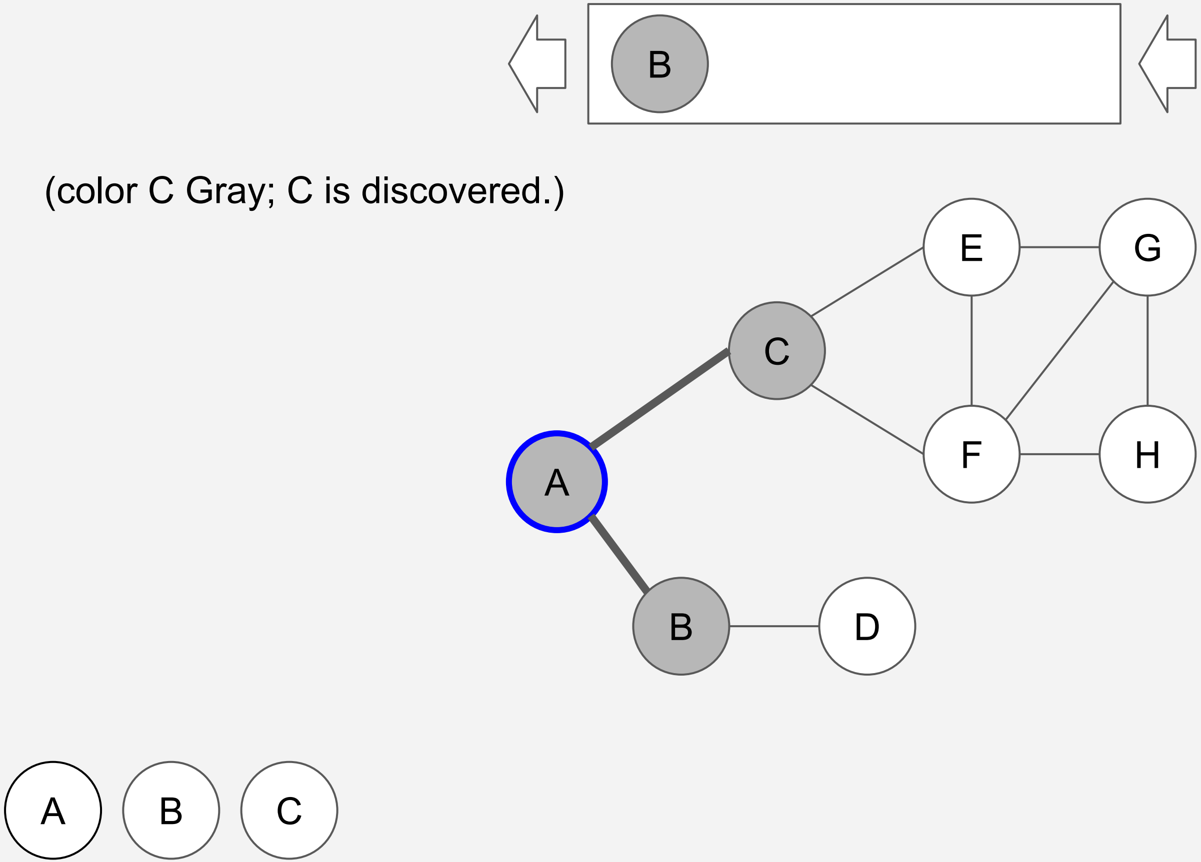
Breadth-first search (BFS)

(enqueue B)



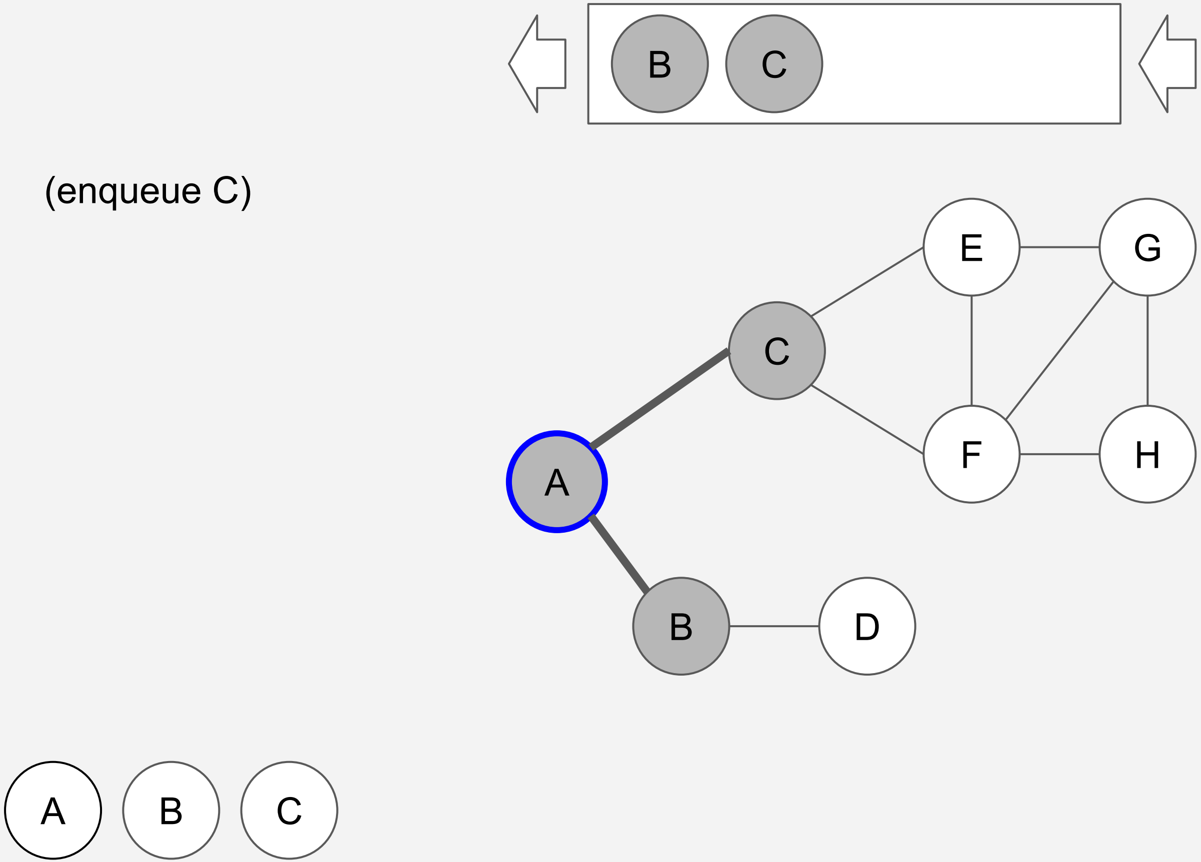


Breadth-first search (BFS)

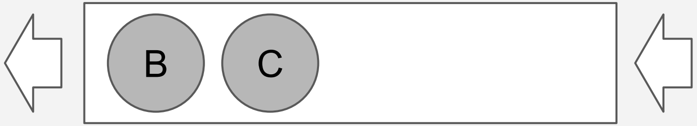


Breadth-first search (BFS)

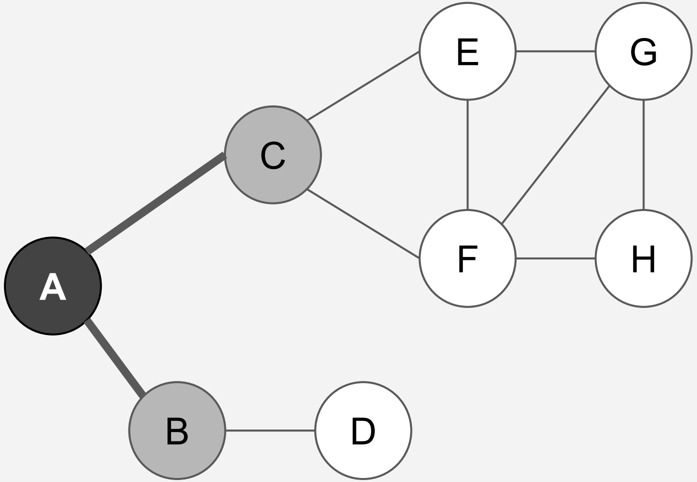
(enqueue C)

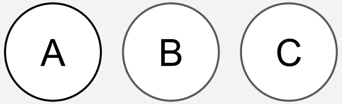


Breadth-first search (BFS)



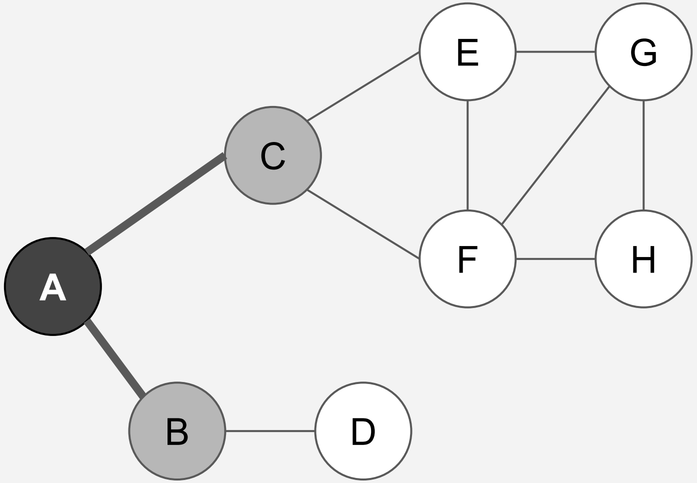
(color A Black; done with A)

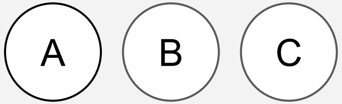




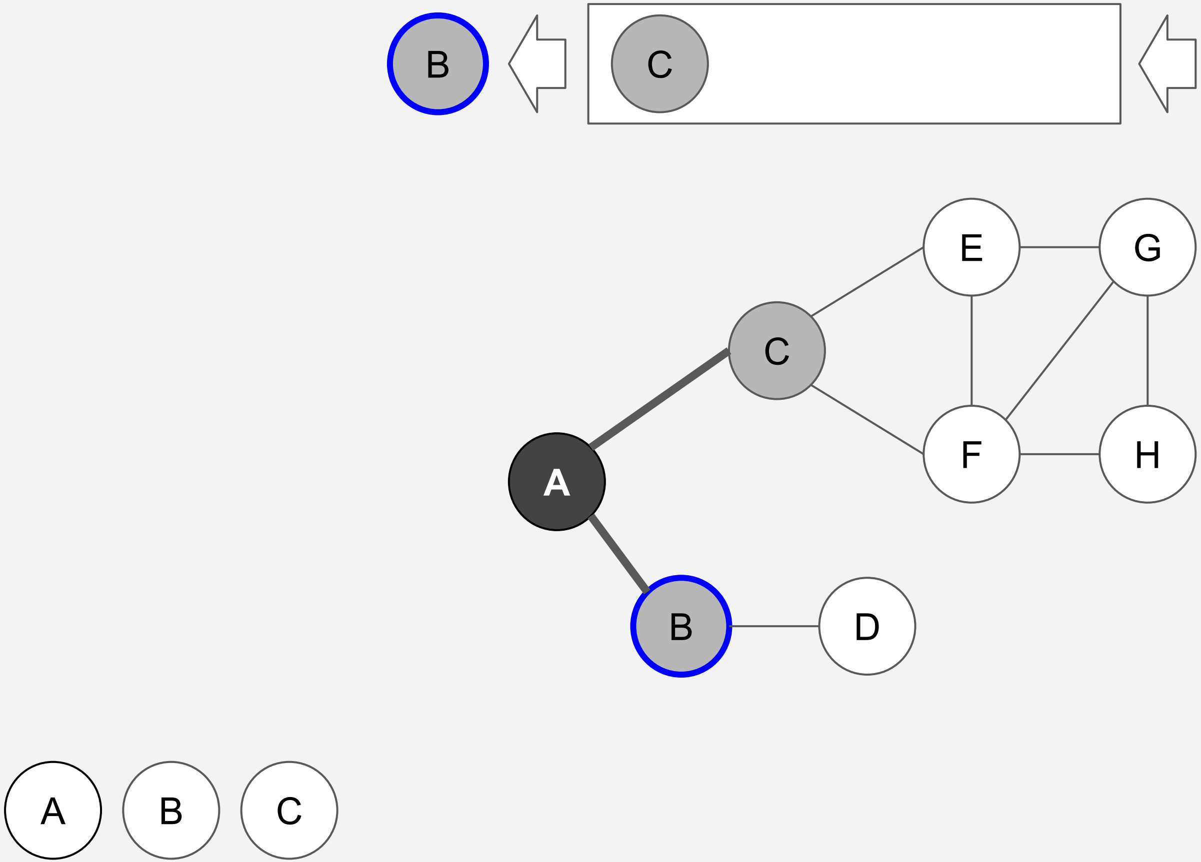
Breadth-first search (BFS)

(front and dequeue B)



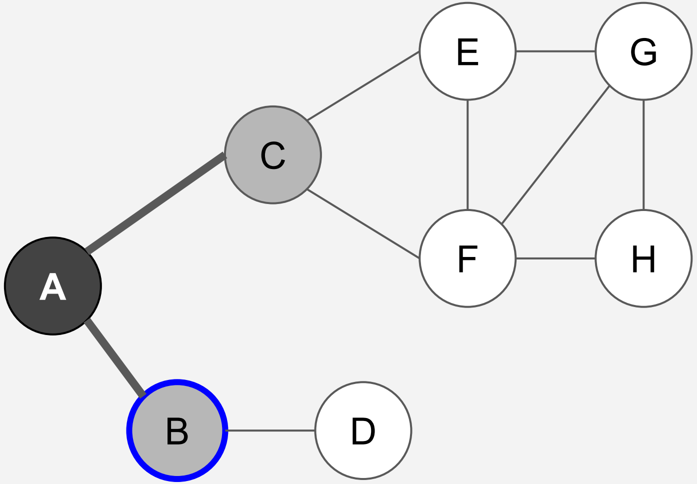


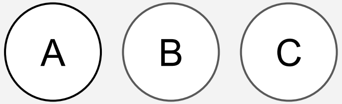
Breadth-first search (BFS)



Breadth-first search (BFS)

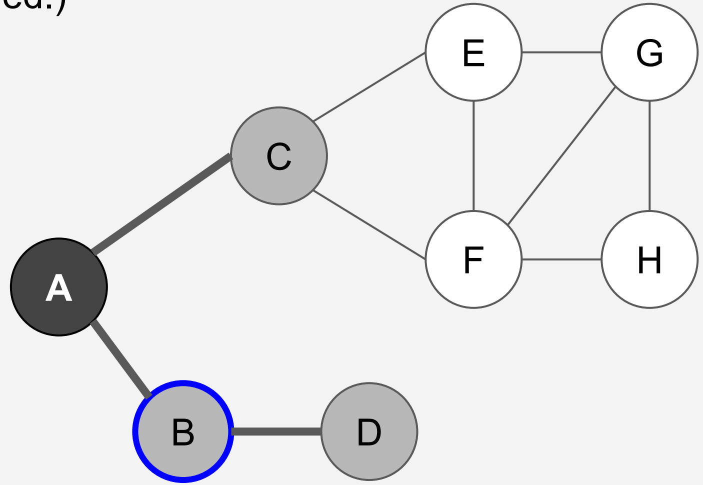
(explore edges from B)

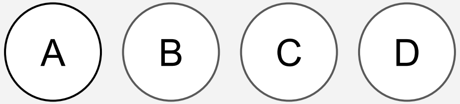




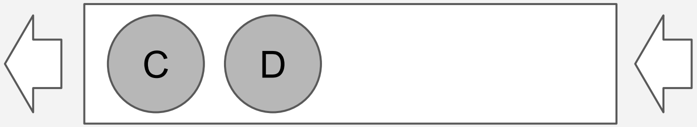
Breadth-first search (BFS)

(color D Gray; D is discovered.)

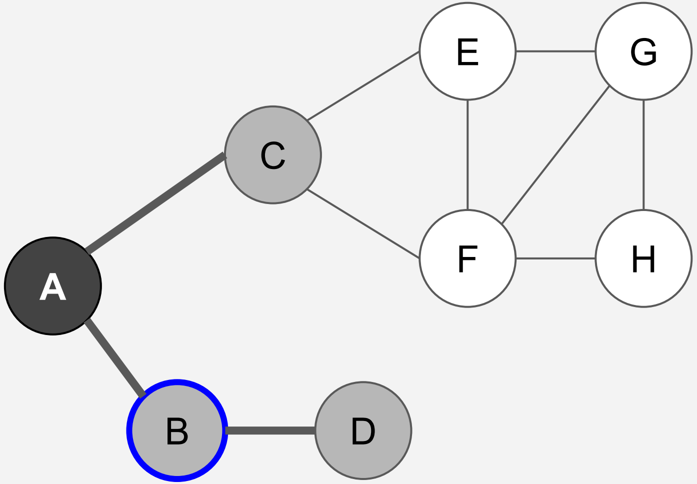


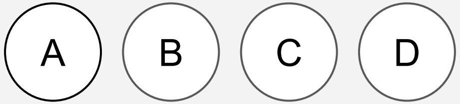


Breadth-first search (BFS)

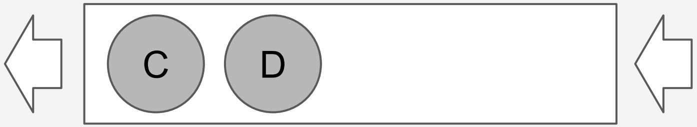


(enqueue D)

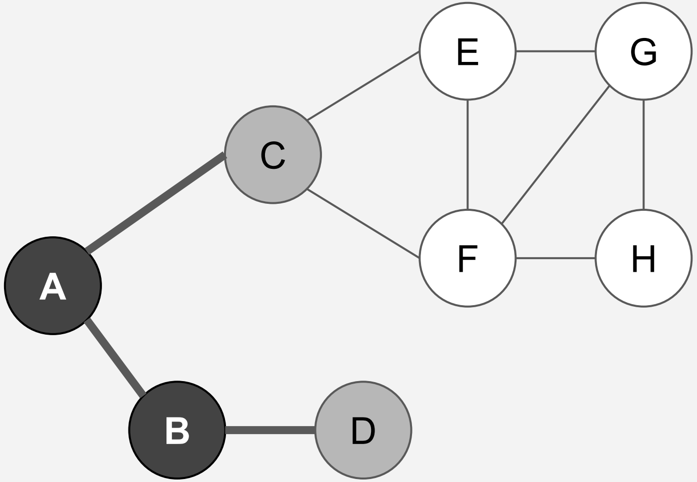


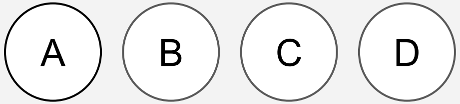


Breadth-first search (BFS)



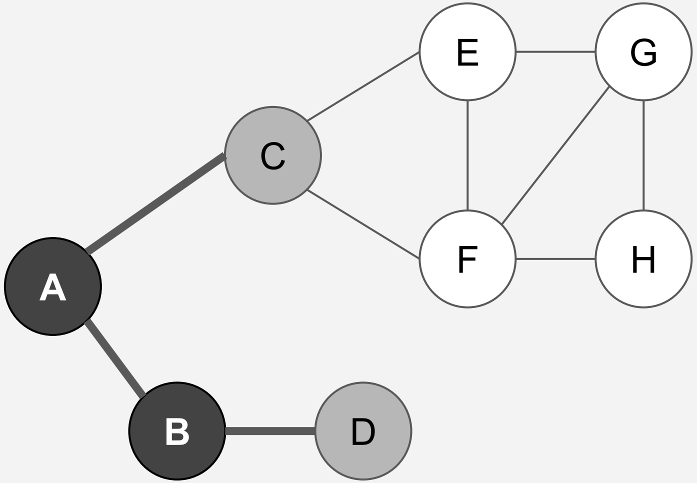
(color B Black; done with B)

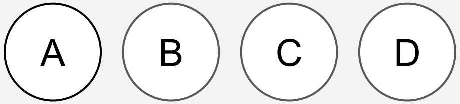




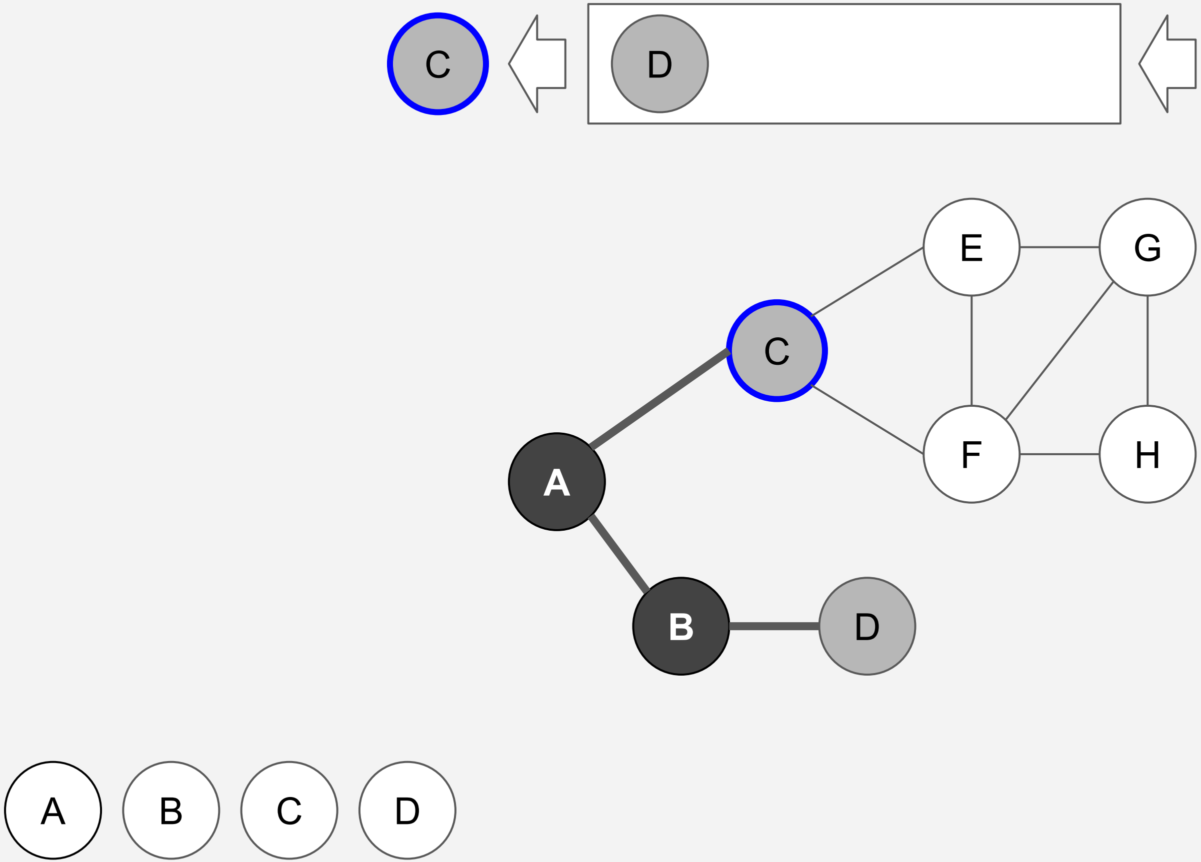
Breadth-first search (BFS)

(front and dequeue C)



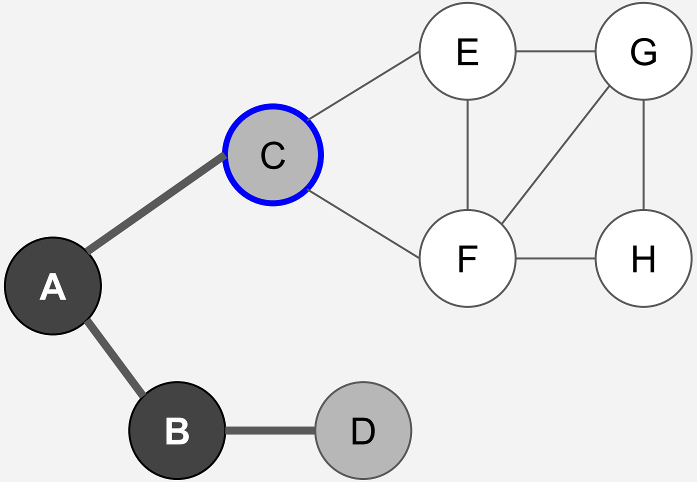


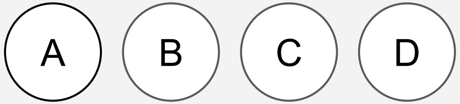
Breadth-first search (BFS)



Breadth-first search (BFS)

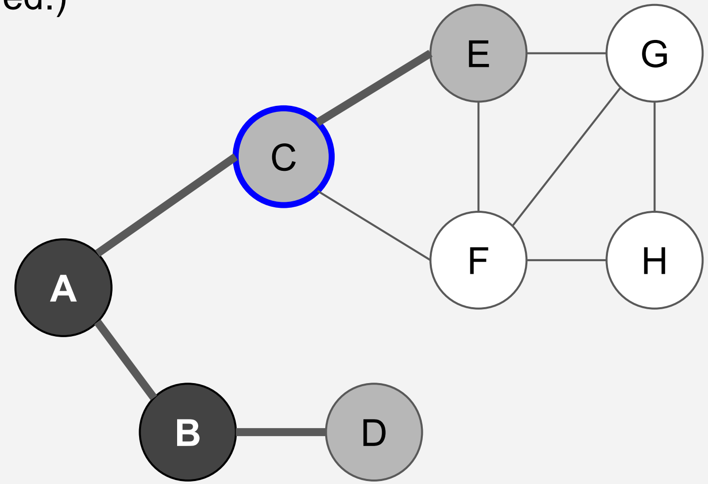
(explore edges from C)

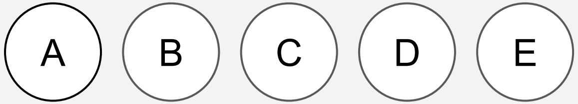




Breadth-first search (BFS)

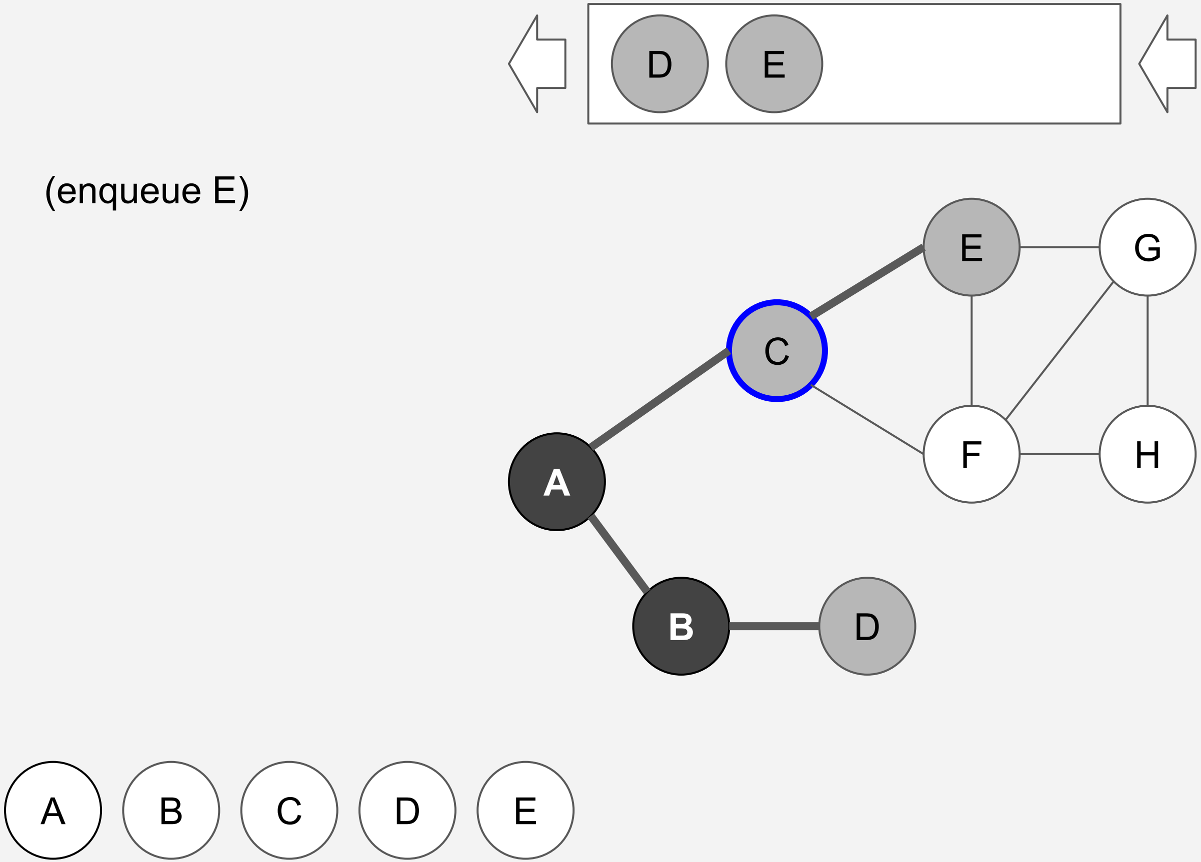
(color E Gray; E is discovered.)



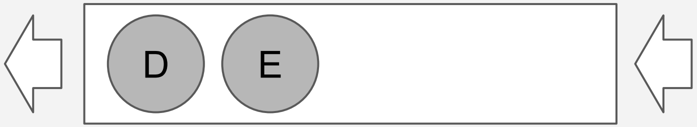


Breadth-first search (BFS)

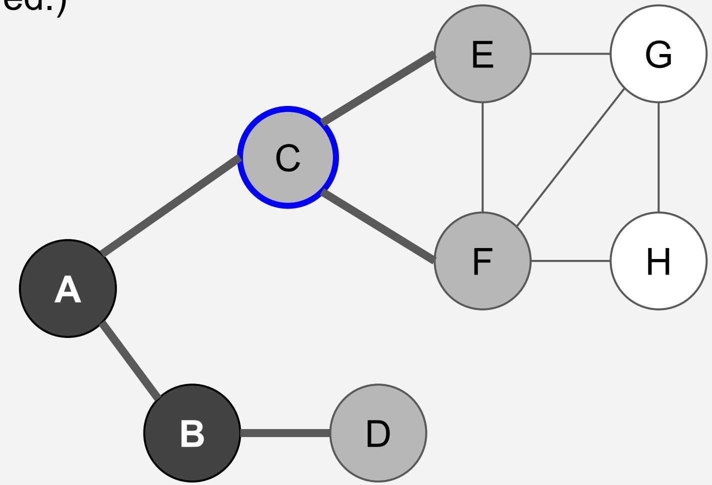
(enqueue E)

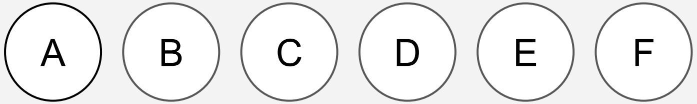


Breadth-first search (BFS)

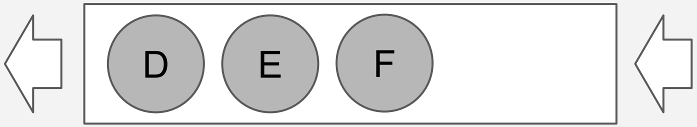


(color F Gray; F is discovered.)

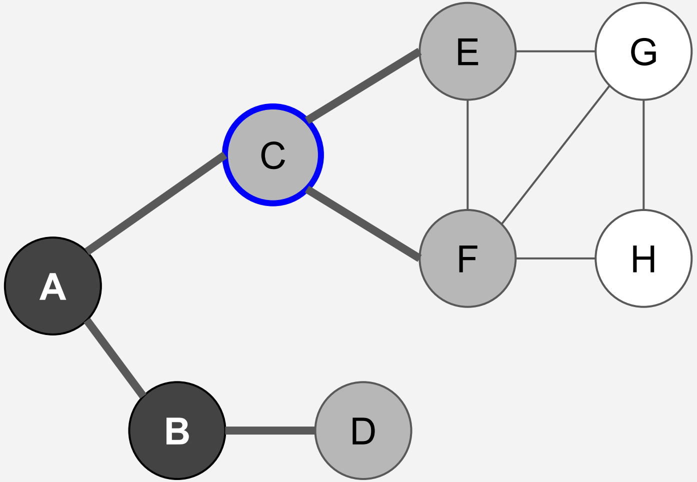


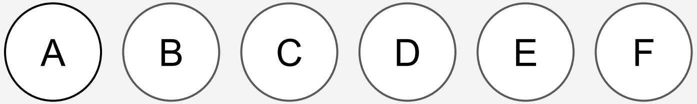


Breadth-first search (BFS)

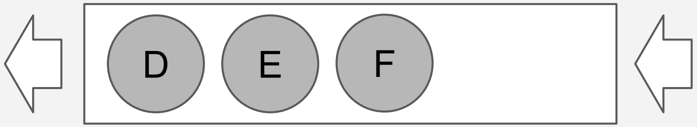


(enqueue F)

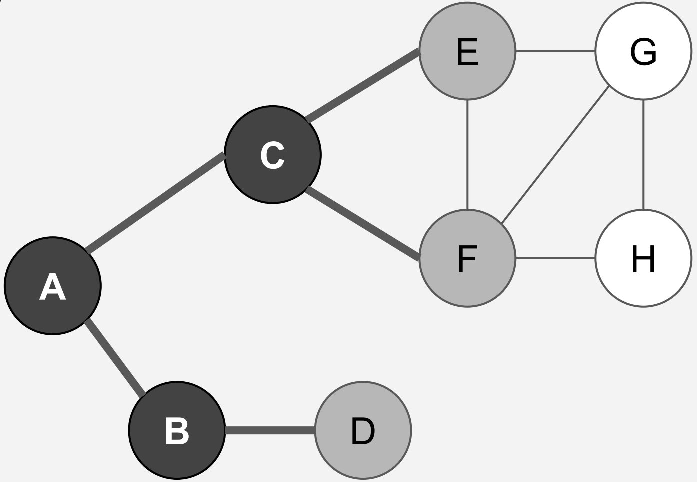


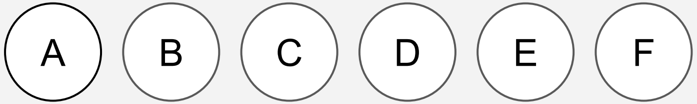


Breadth-first search (BFS)

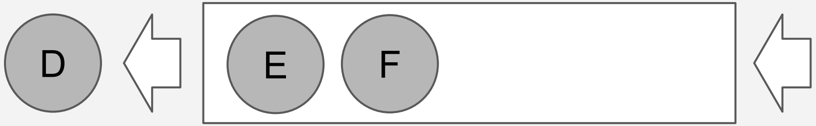


(color C Black; done with C)

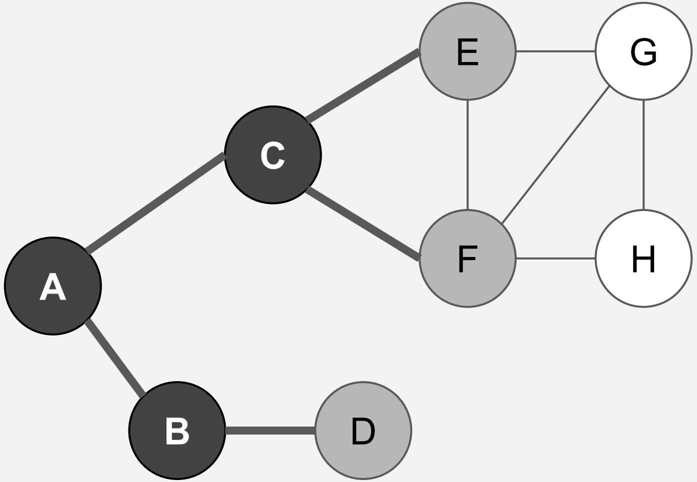


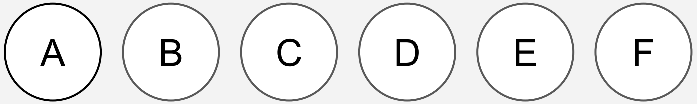


Breadth-first search (BFS)

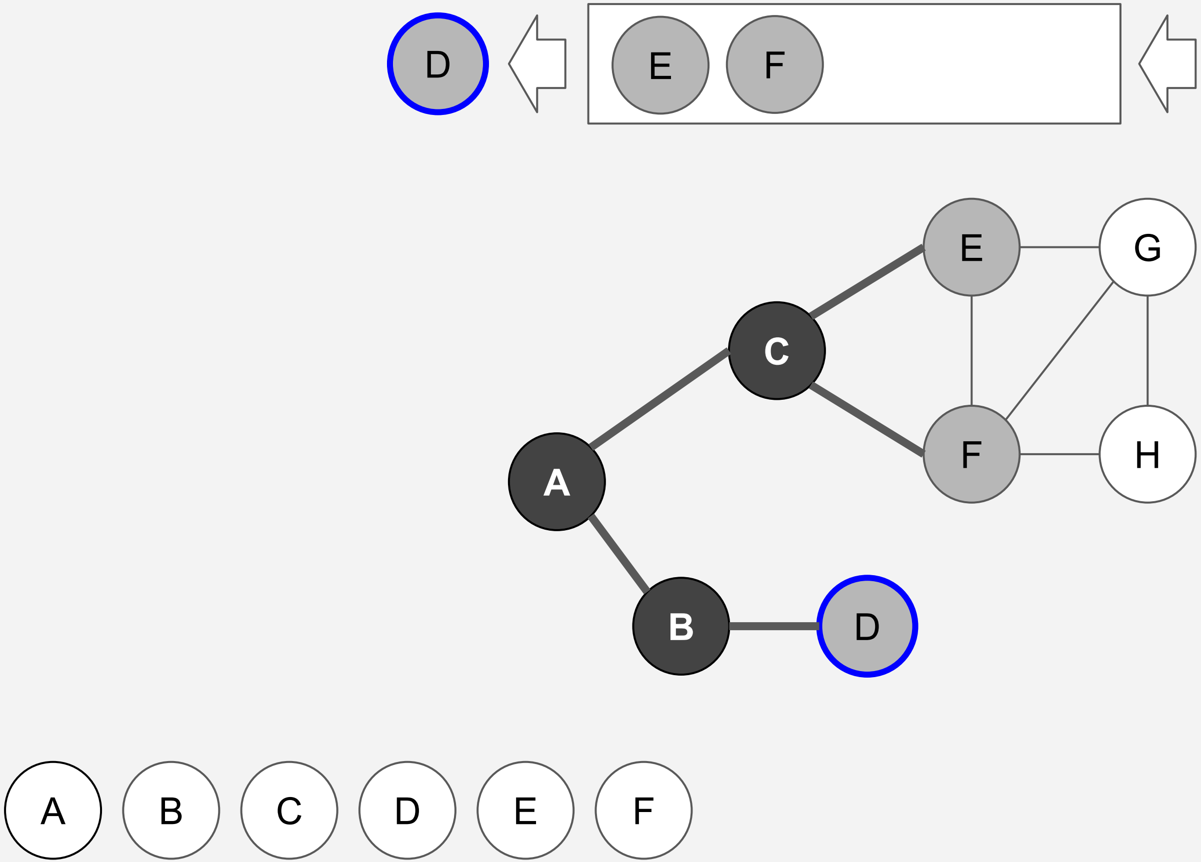


(front and dequeue D)

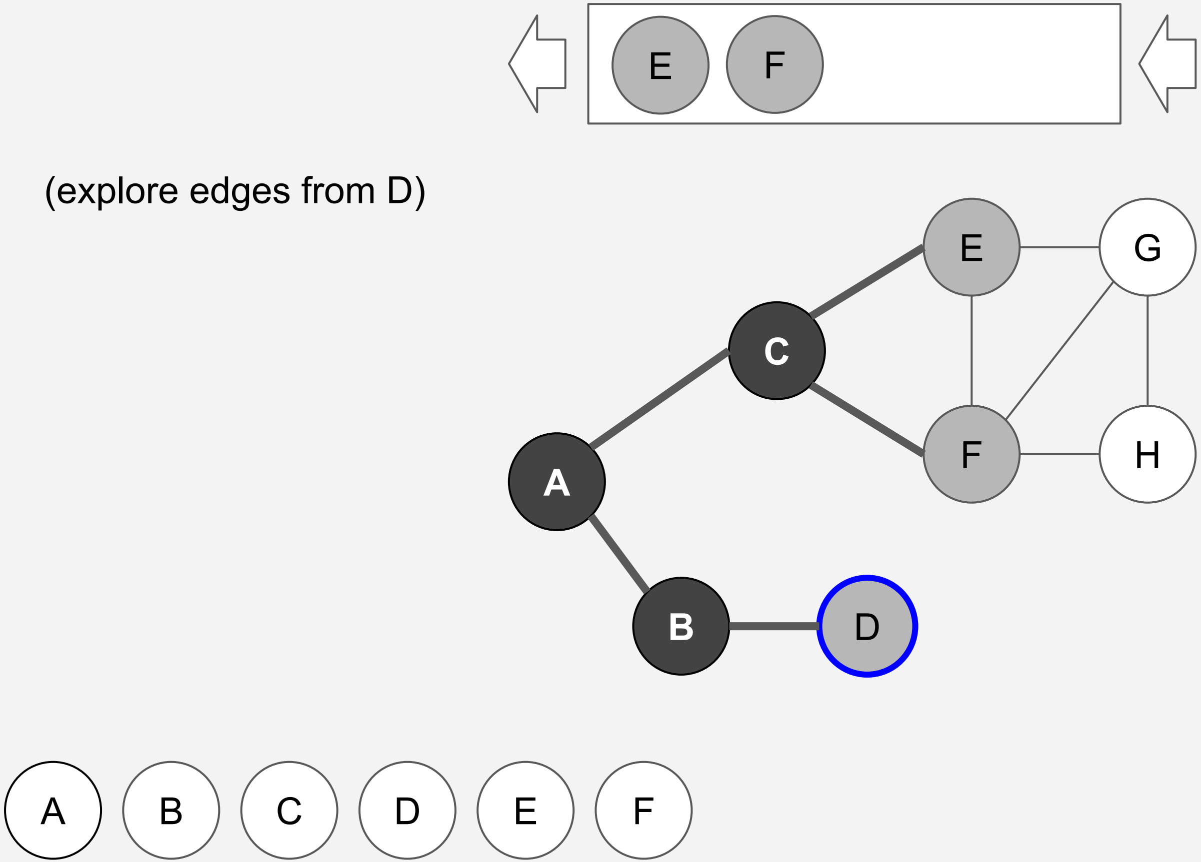




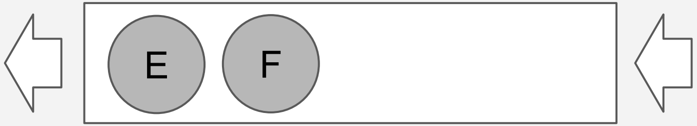
Breadth-first search (BFS)



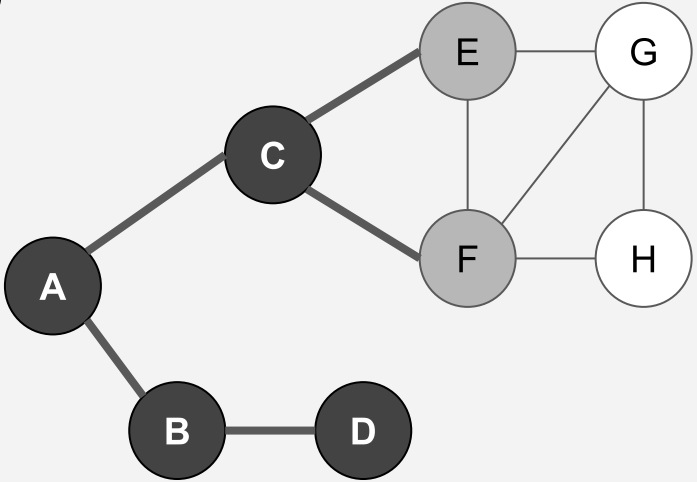
Breadth-first search (BFS)

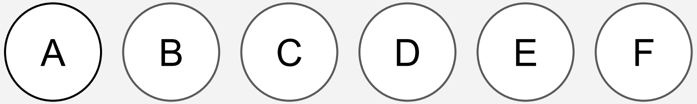


Breadth-first search (BFS)



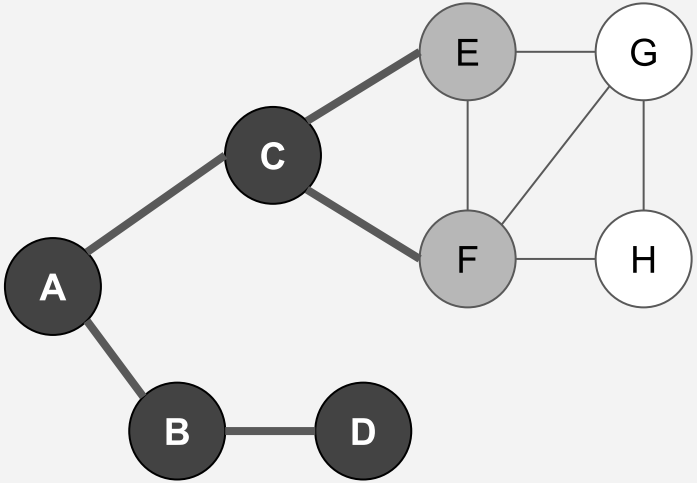
(color D Black; done with D)

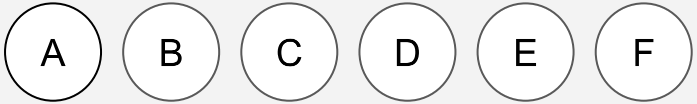




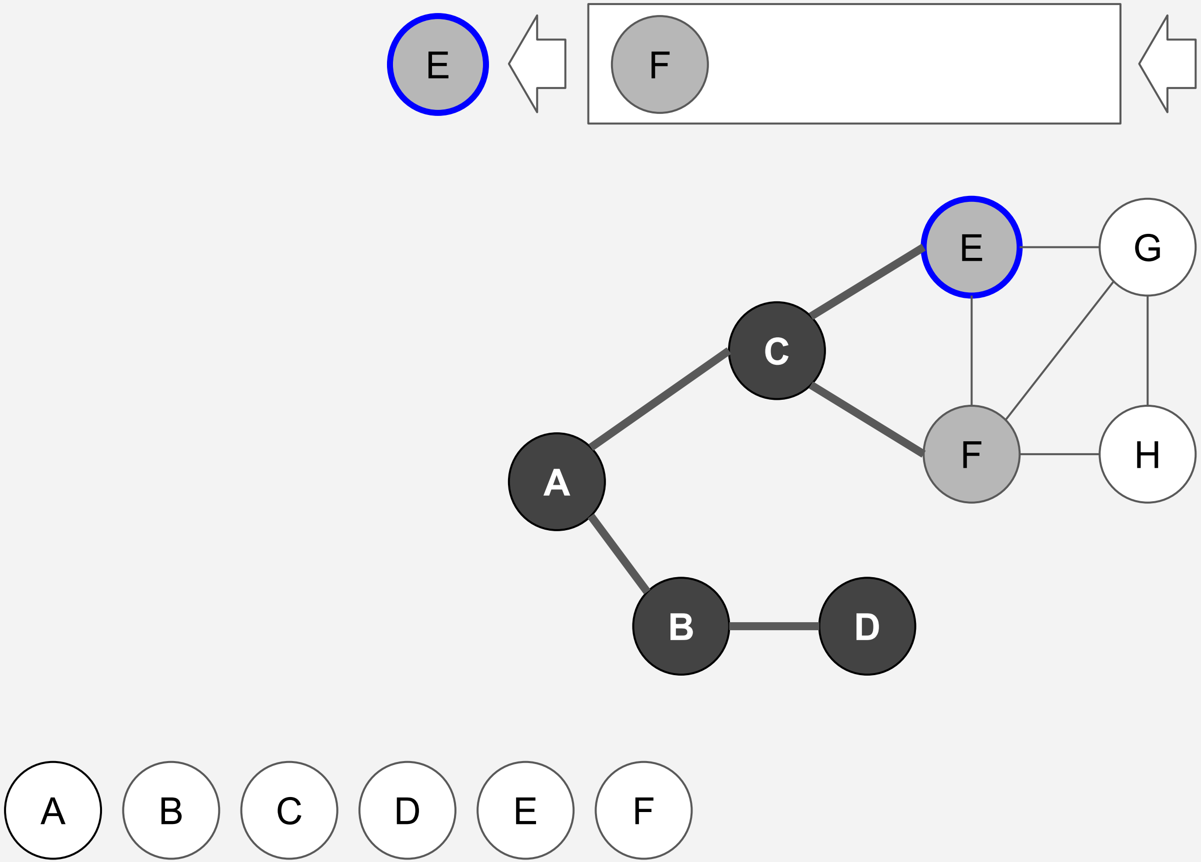
Breadth-first search (BFS)

(front and deque E)



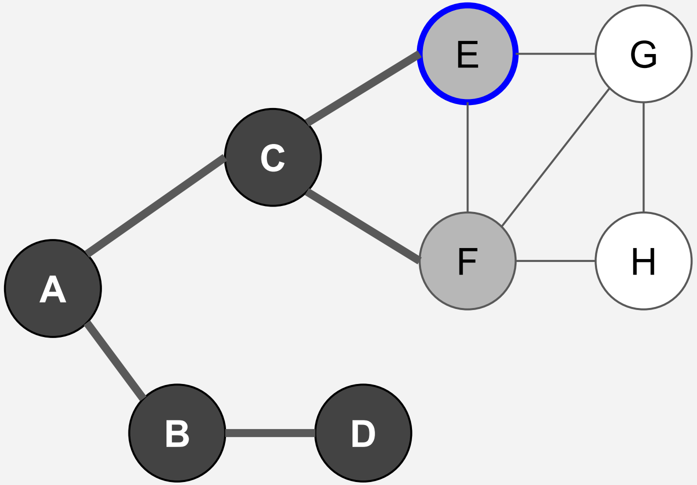


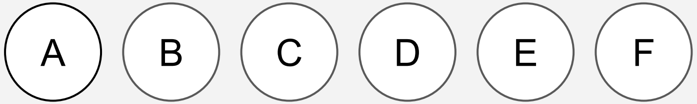
Breadth-first search (BFS)



Breadth-first search (BFS)

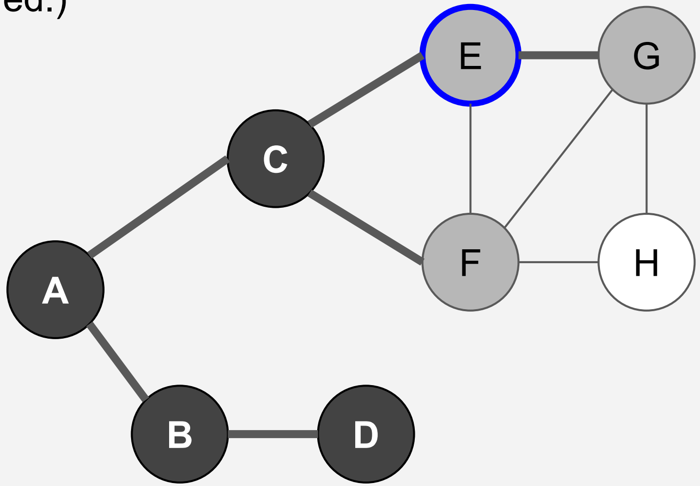
(explore edges from E)

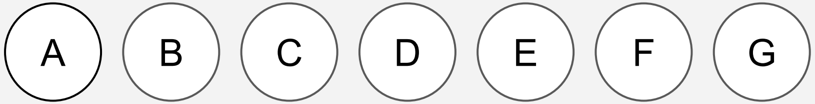




Breadth-first search (BFS)

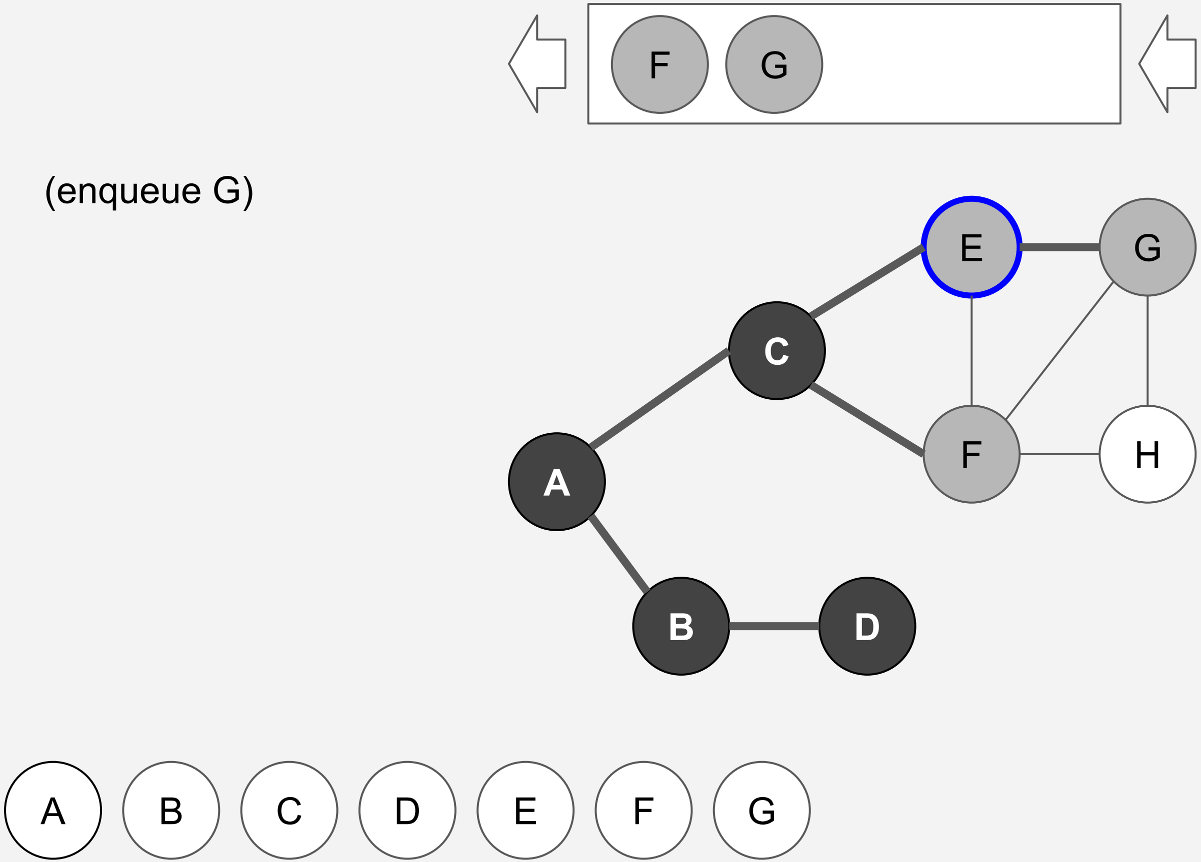
(color G Gray; G is discovered.)



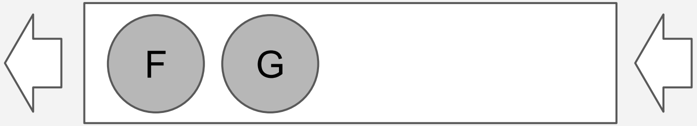


Breadth-first search (BFS)

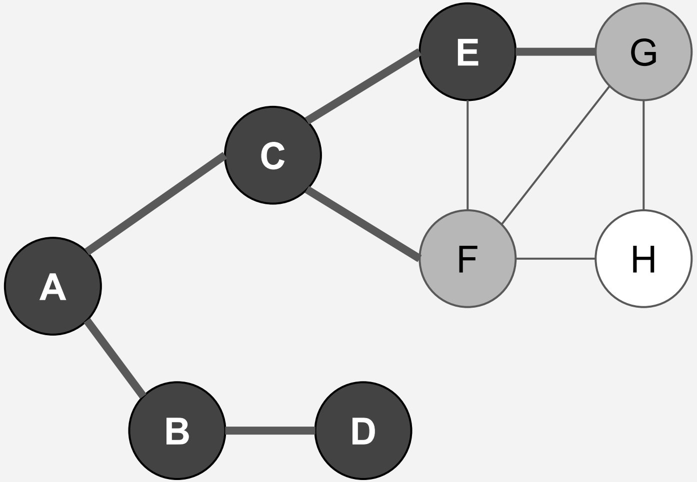
(enqueue G)

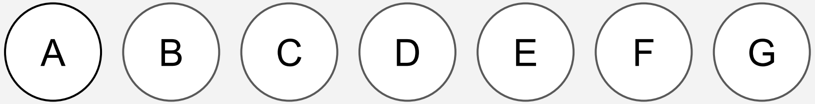


Breadth-first search (BFS)



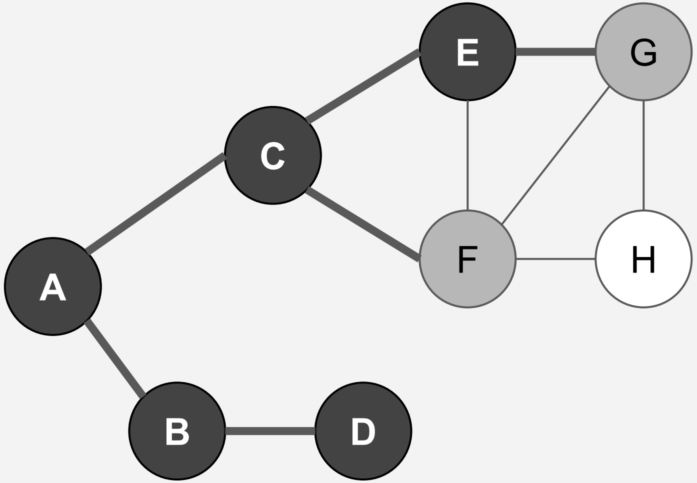
(color E Black; done with E)

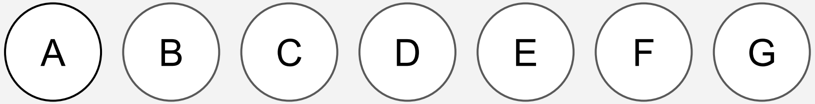




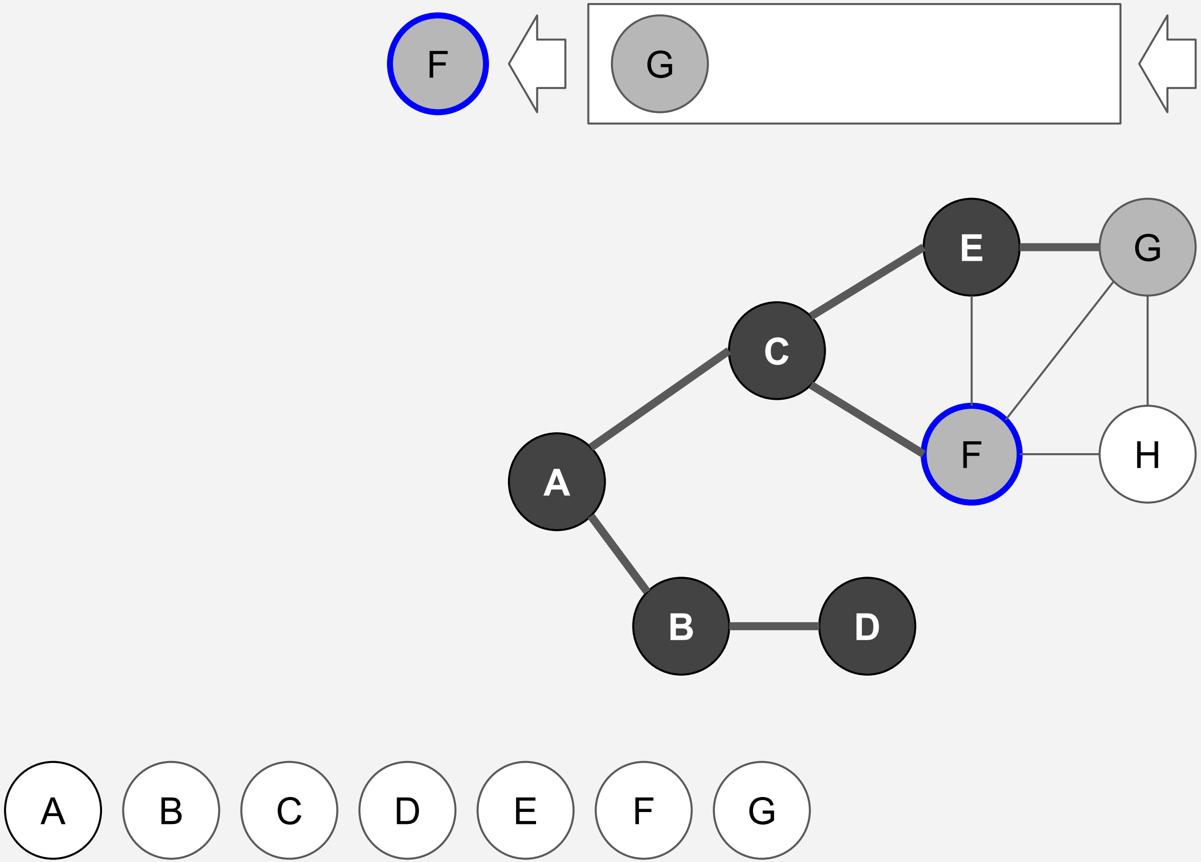
Breadth-first search (BFS)

(front and dequeue F)



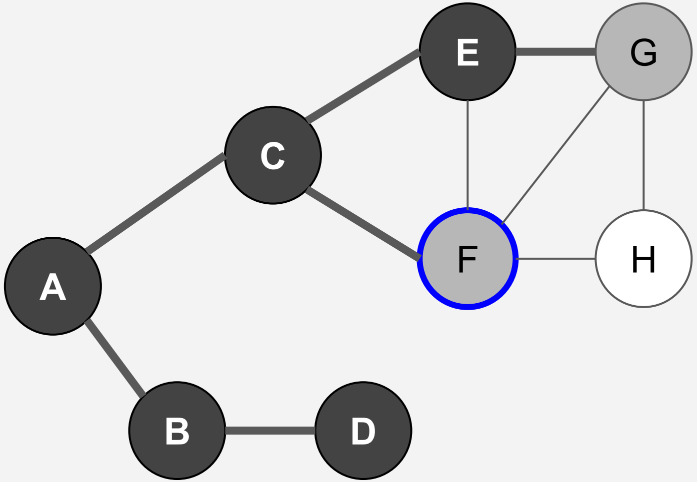


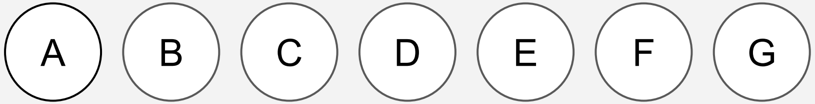
Breadth-first search (BFS)



Breadth-first search (BFS)

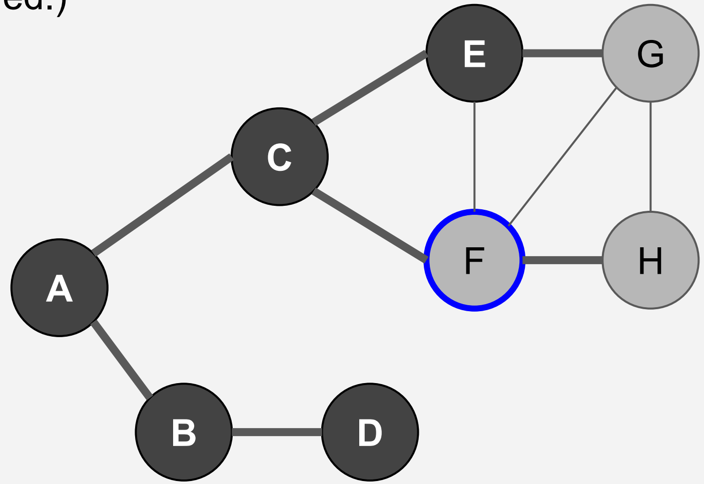
(explore edges from F)

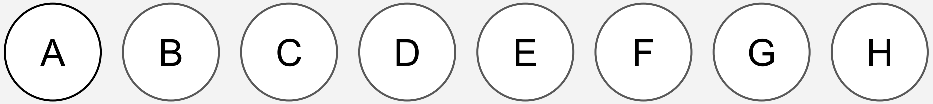




Breadth-first search (BFS)

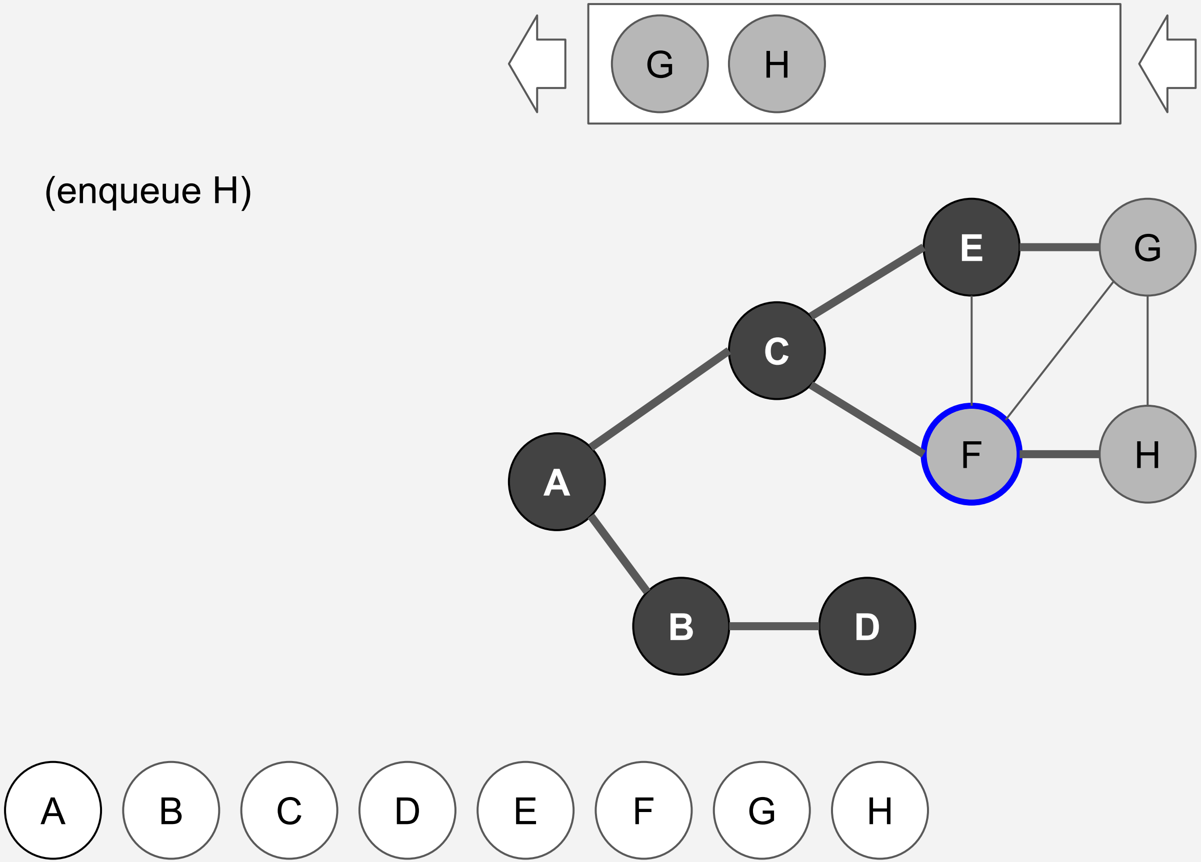
(color H Gray; H is discovered.)



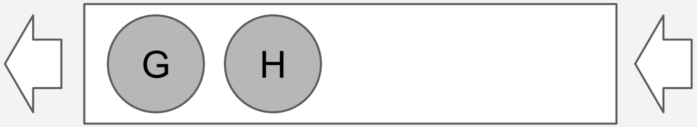


Breadth-first search (BFS)

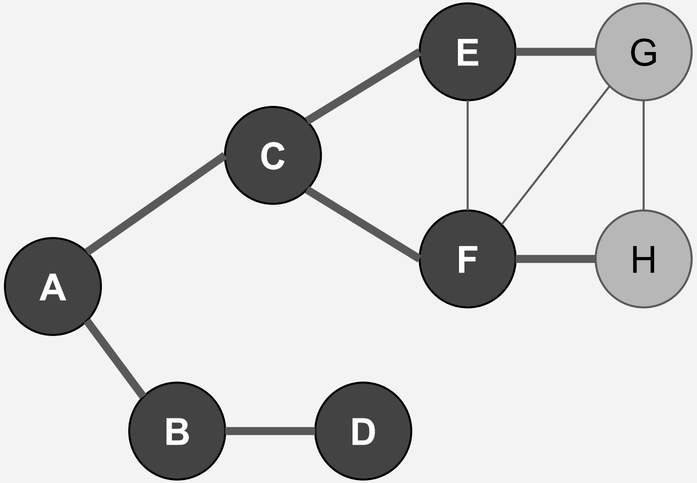
(enqueue H)

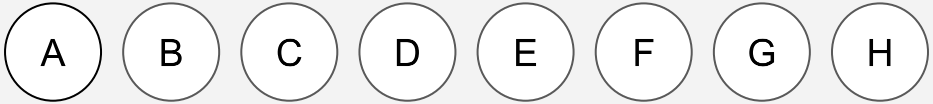


Breadth-first search (BFS)



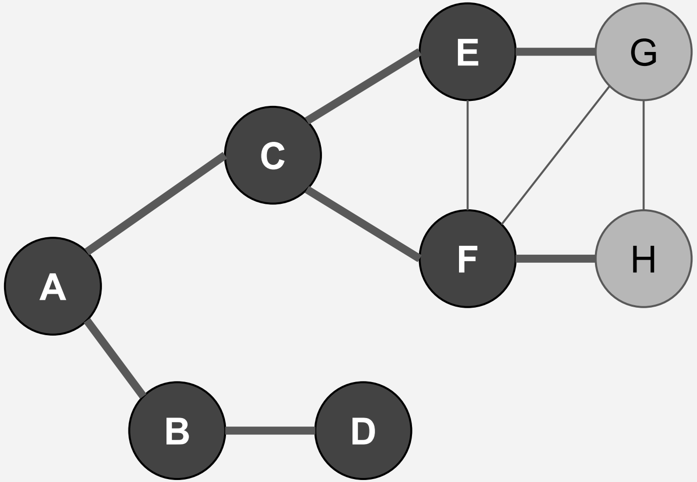
(color F Black; done with F)

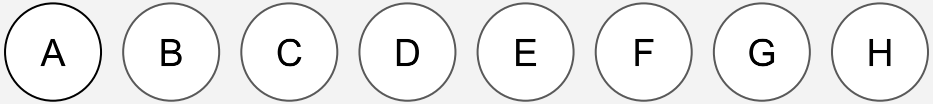




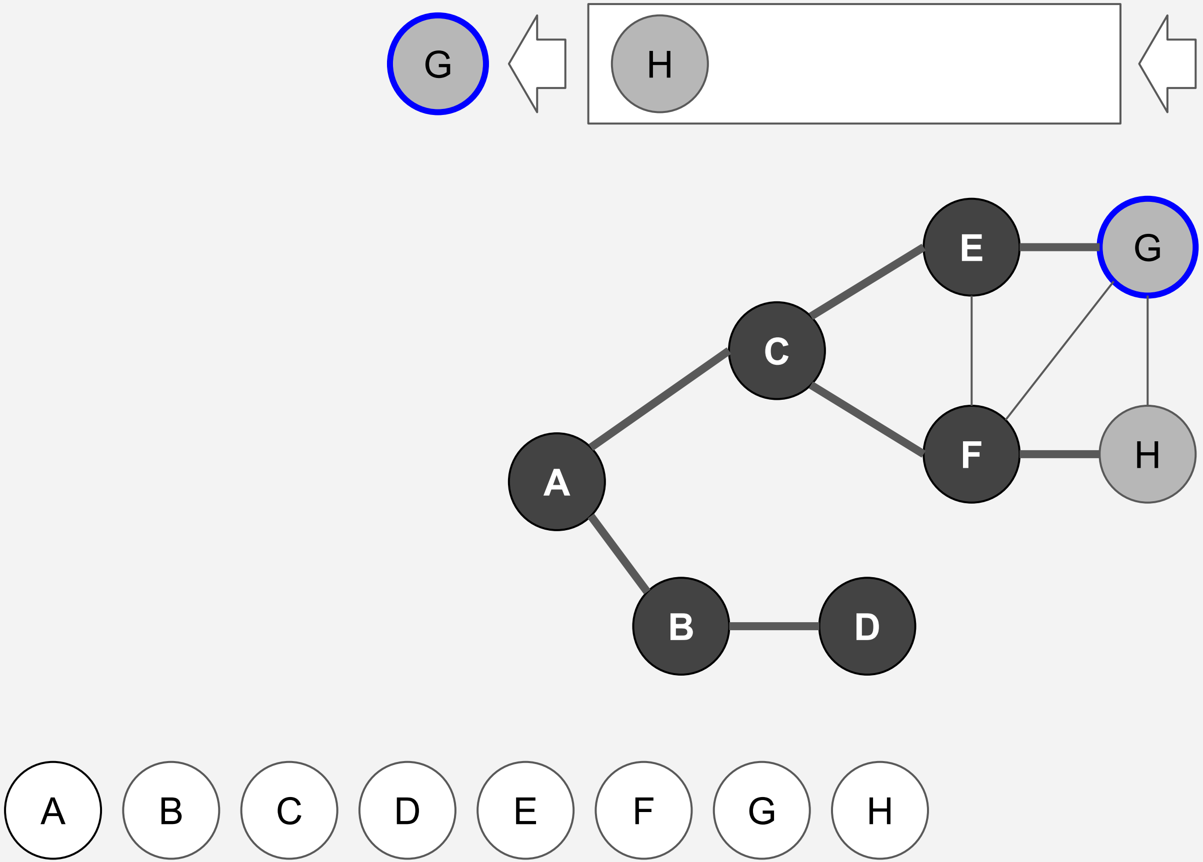
Breadth-first search (BFS)

(front and dequeue G)



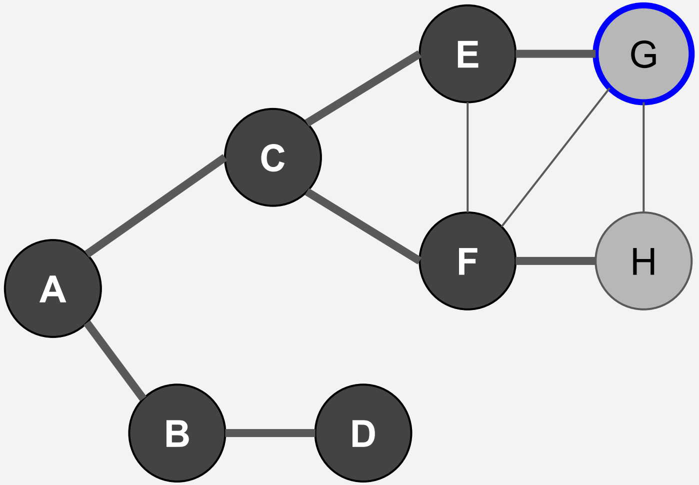


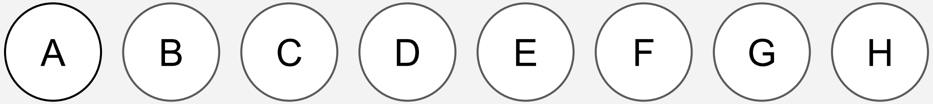
Breadth-first search (BFS)



Breadth-first search (BFS)

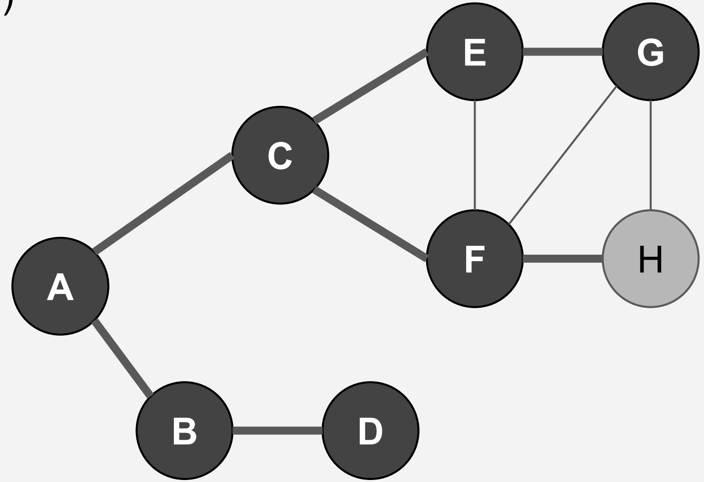
(explore edges from G)

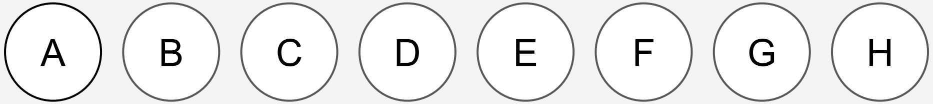




Breadth-first search (BFS)

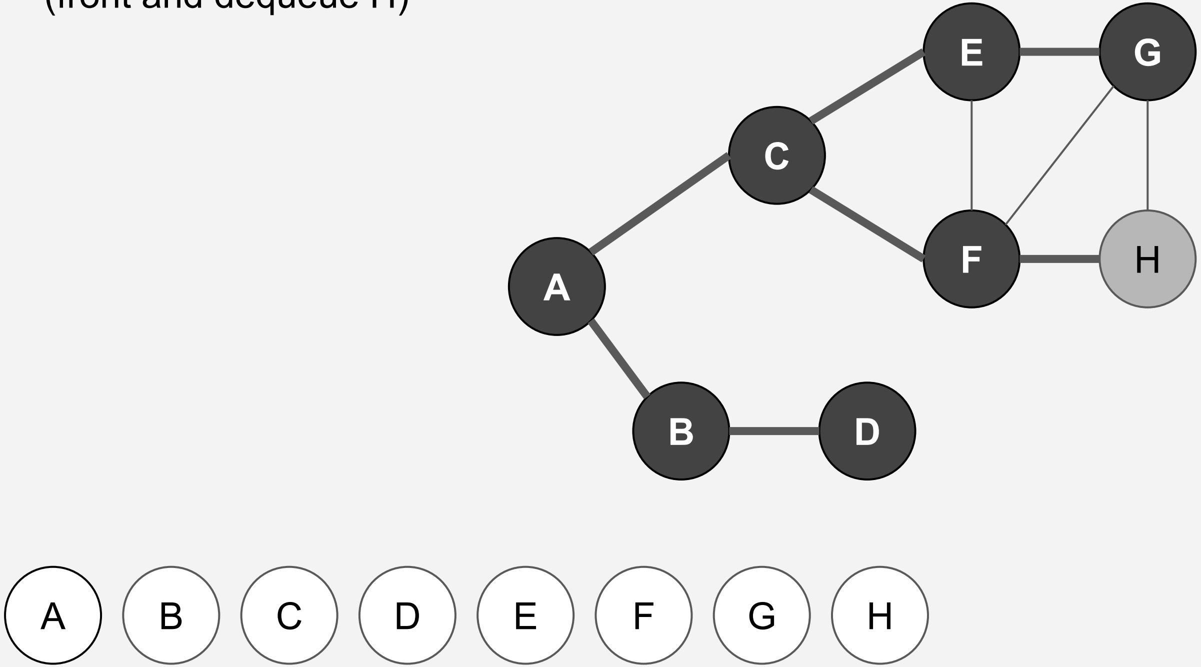
(color G Black; done with G)



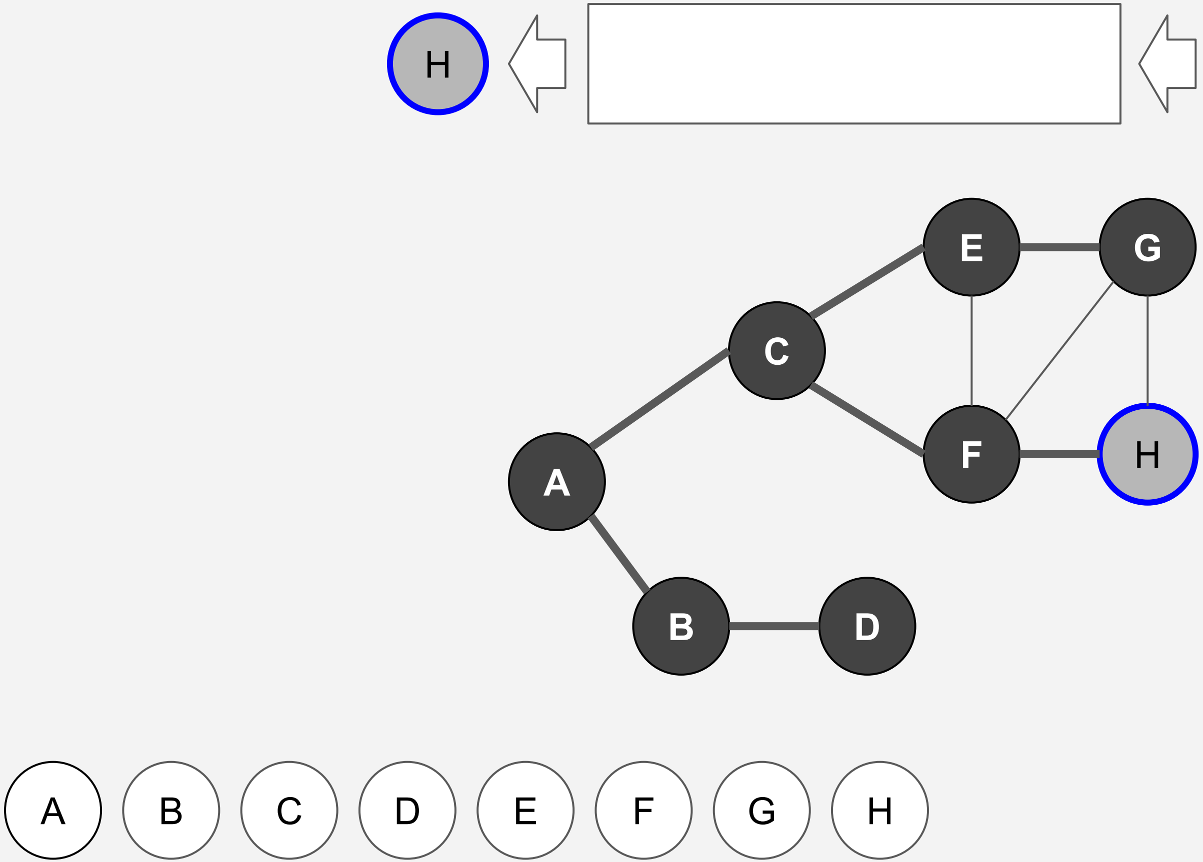


Breadth-first search (BFS)

(front and dequeue H)

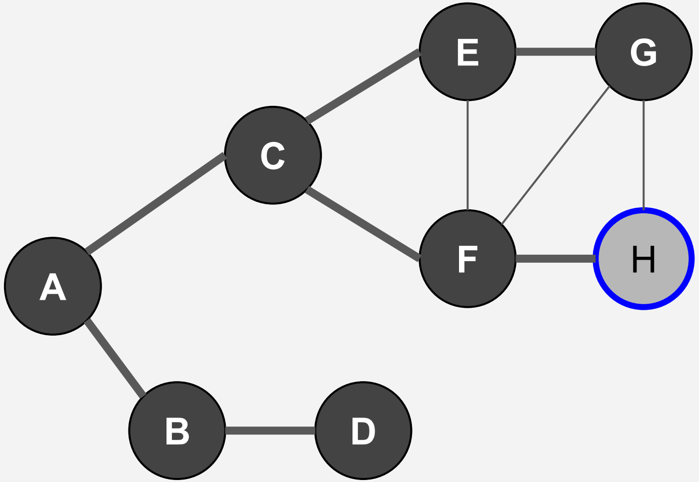


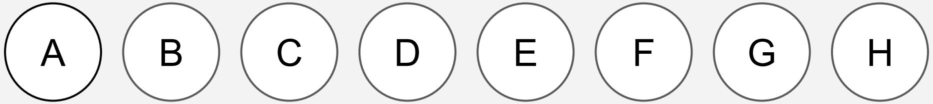
Breadth-first search (BFS)



Breadth-first search (BFS)

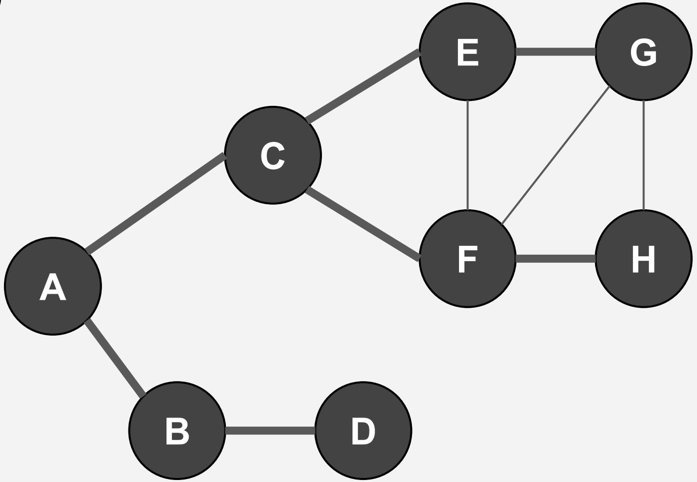
(explore edges from H)

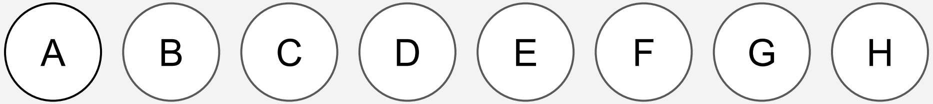




Breadth-first search (BFS)

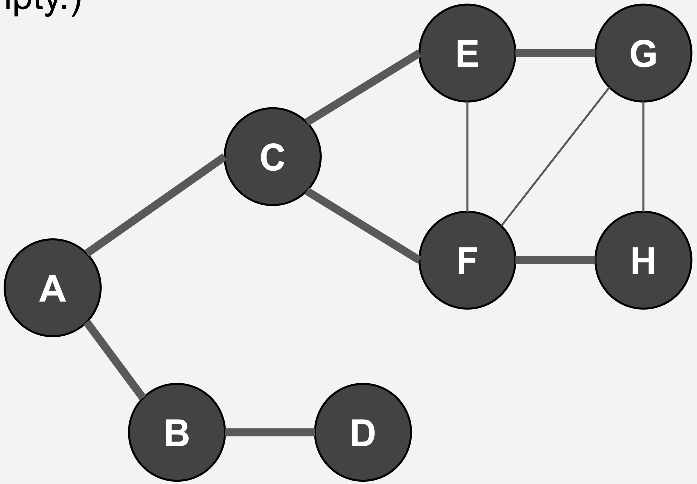
(color H Black; done with H)

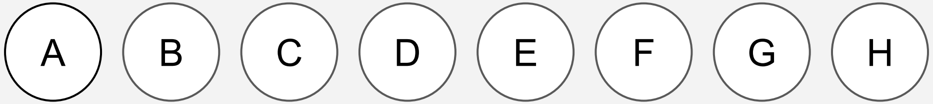




Breadth-first search (BFS)

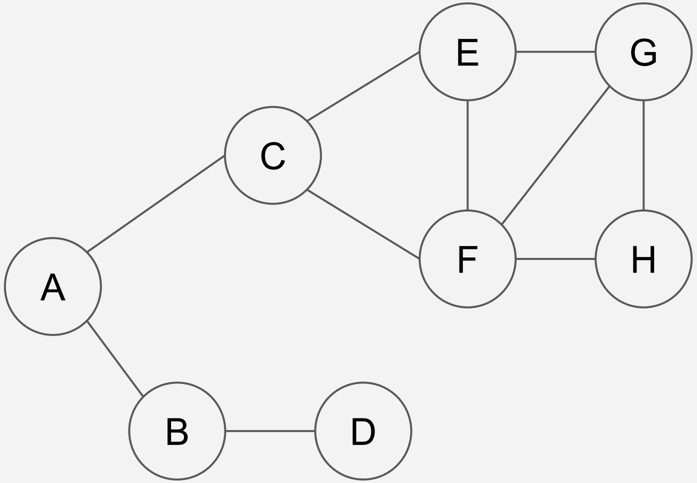
(done; The queue is now empty.)





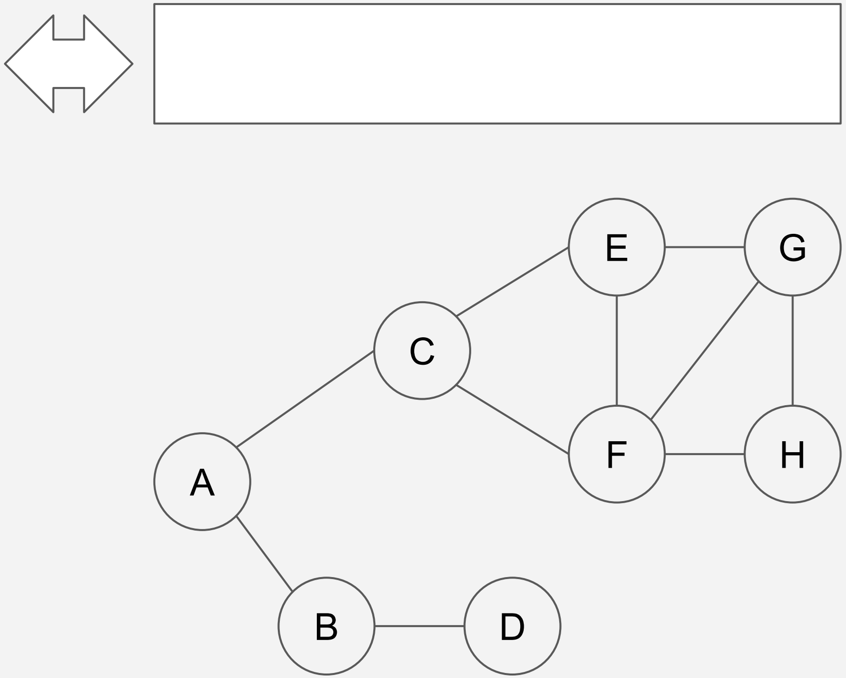
Depth-first search (DFS)

Depth-first search (DFS)



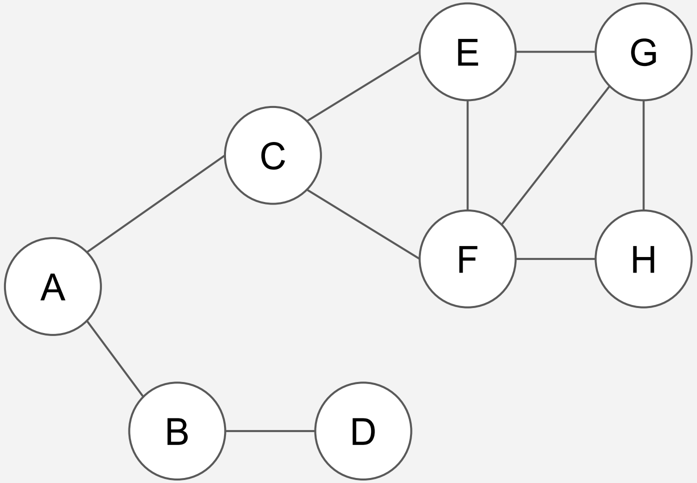
Depth-first search (DFS)

(using a stack)



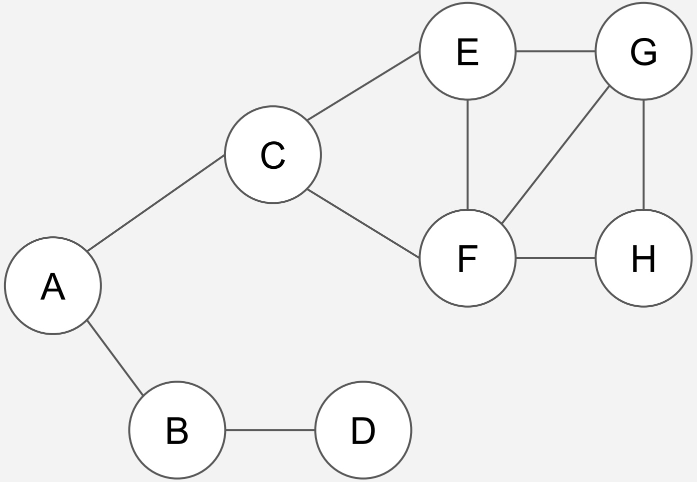
Depth-first search (DFS)

(color all vertices White)



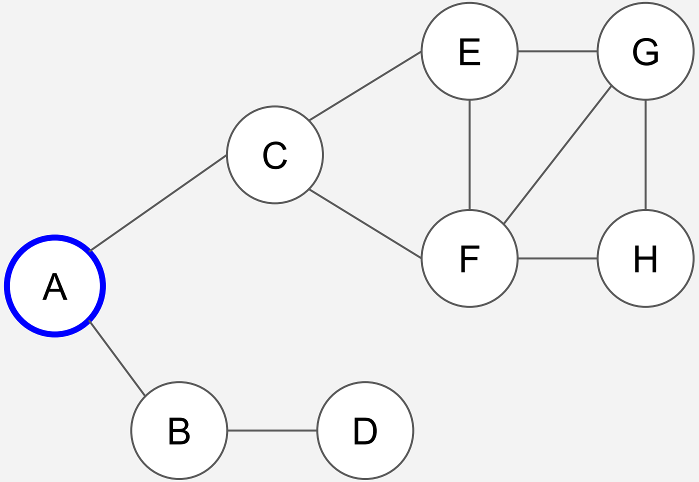
Depth-first search (DFS)

(start from A; push A)



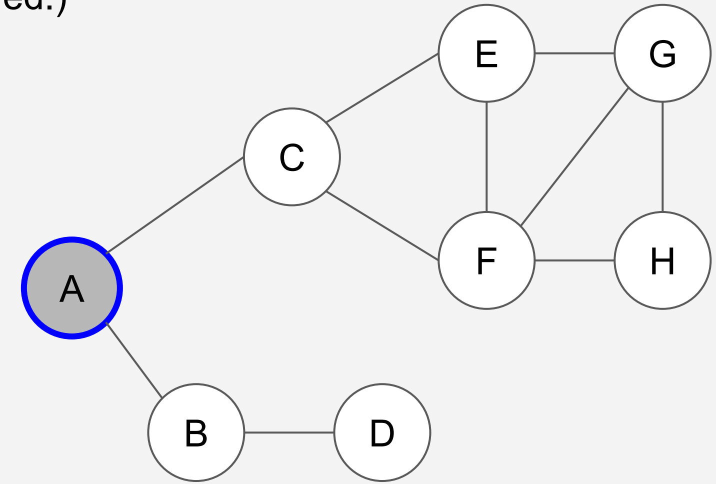
Depth-first search (DFS)

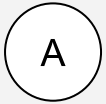
(top A)



Depth-first search (DFS)

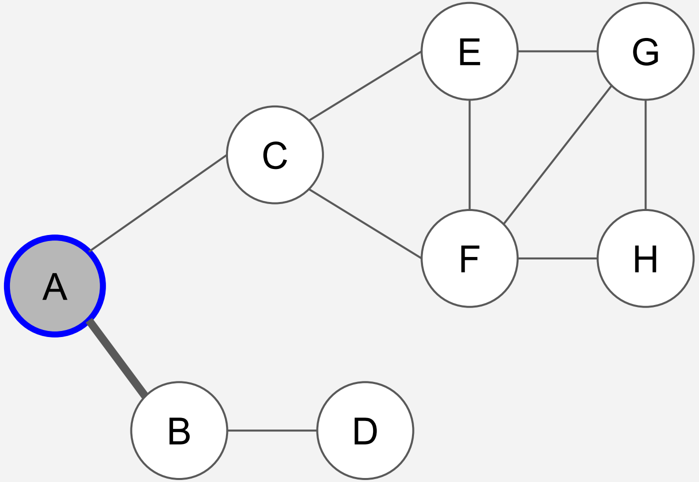
(color A Gray; A is discovered.)

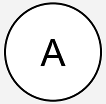




Depth-first search (DFS)

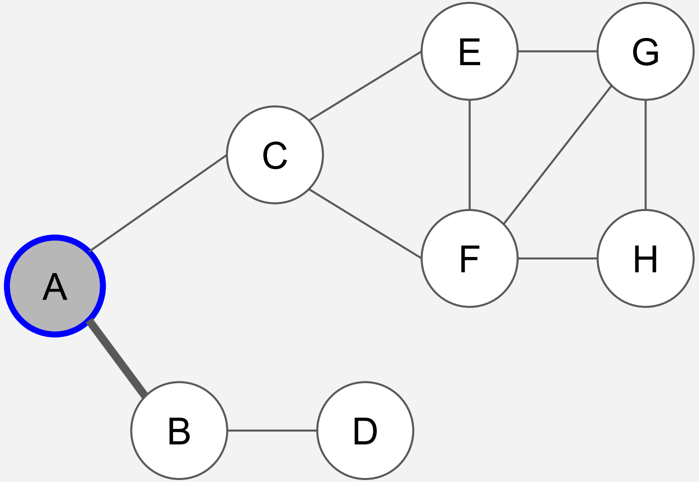
(explore edges from A)

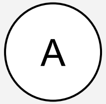




Depth-first search (DFS)

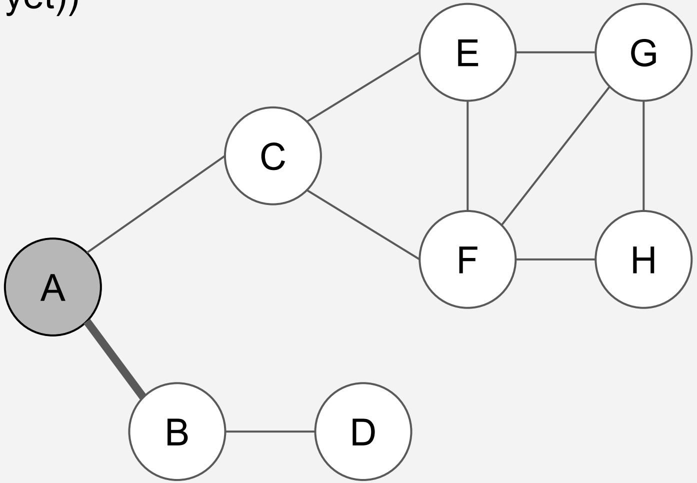
(push B)

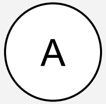




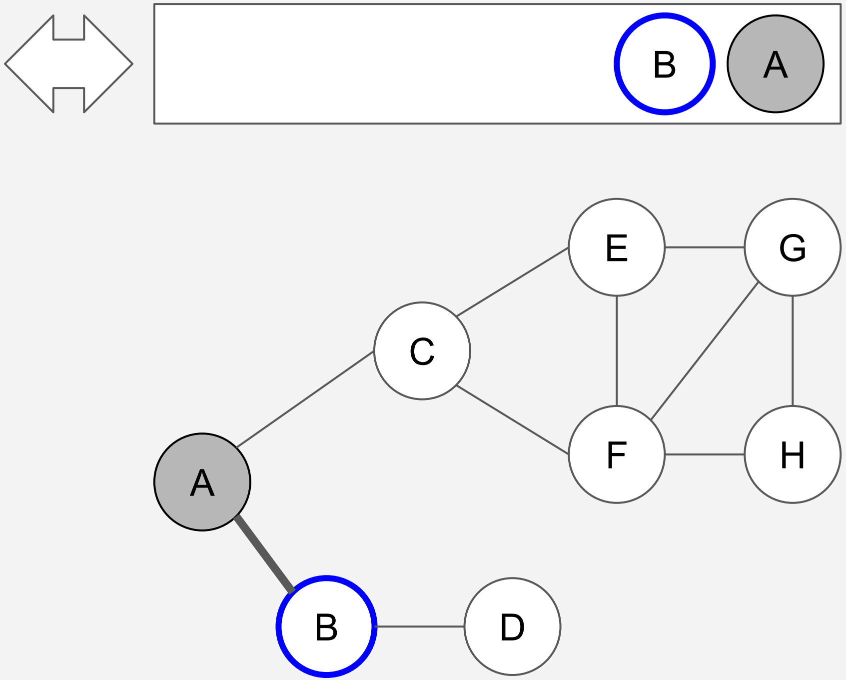
Depth-first search (DFS)

(start over; (not done with A yet))

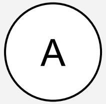




Depth-first search (DFS)

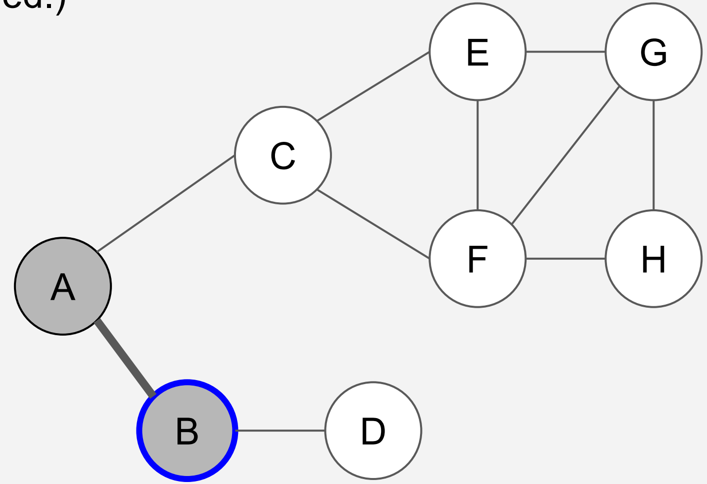


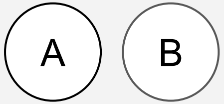
(top B)



Depth-first search (DFS)

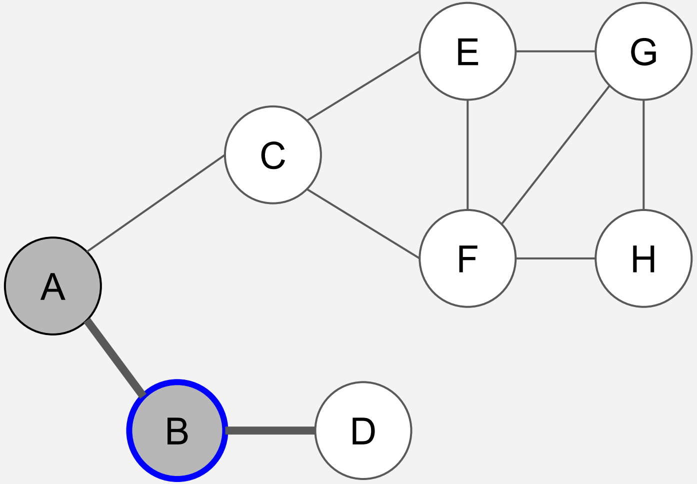
(color B Gray; B is discovered.)

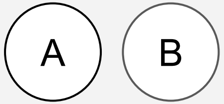




Depth-first search (DFS)

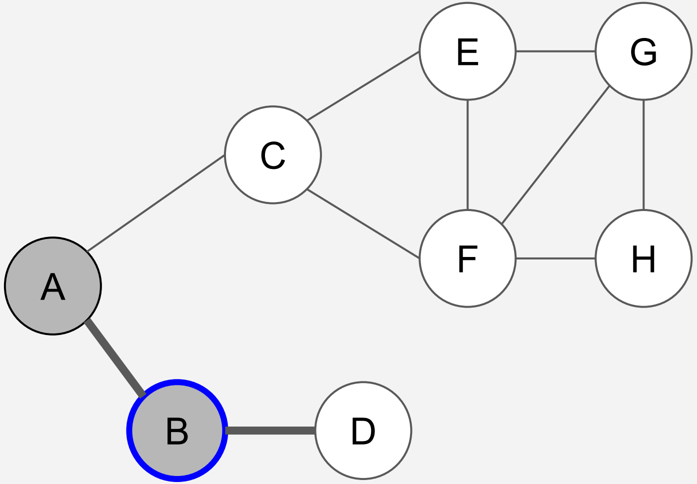
(explore edges from B)

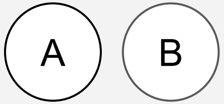




Depth-first search (DFS)

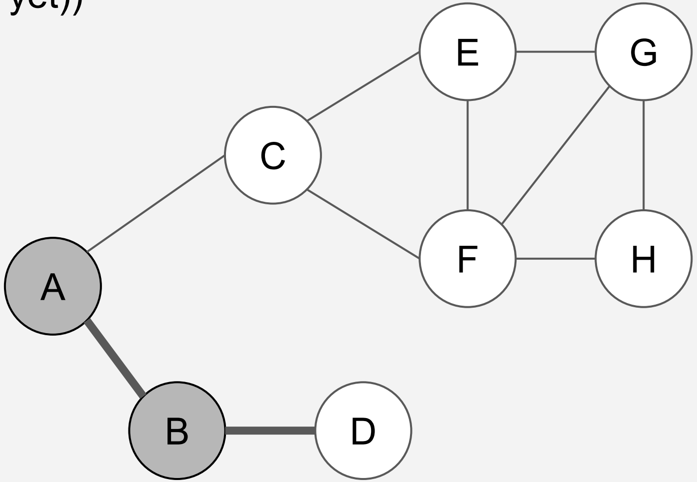
(push D)

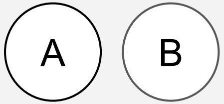




Depth-first search (DFS)

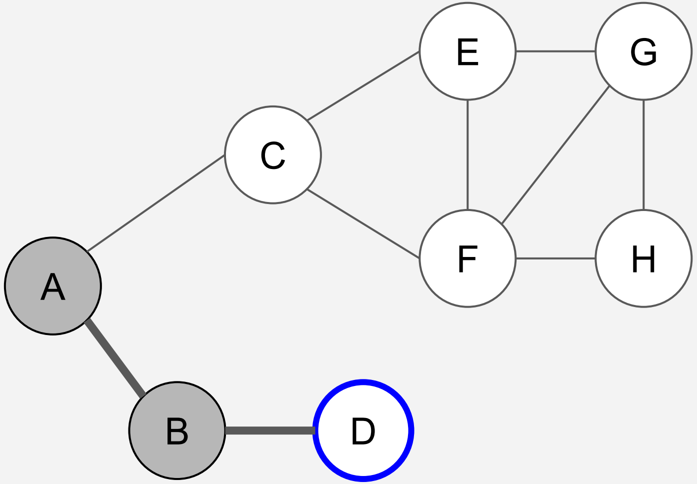
(start over; (not done with B yet))

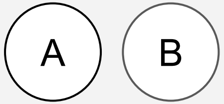




Depth-first search (DFS)

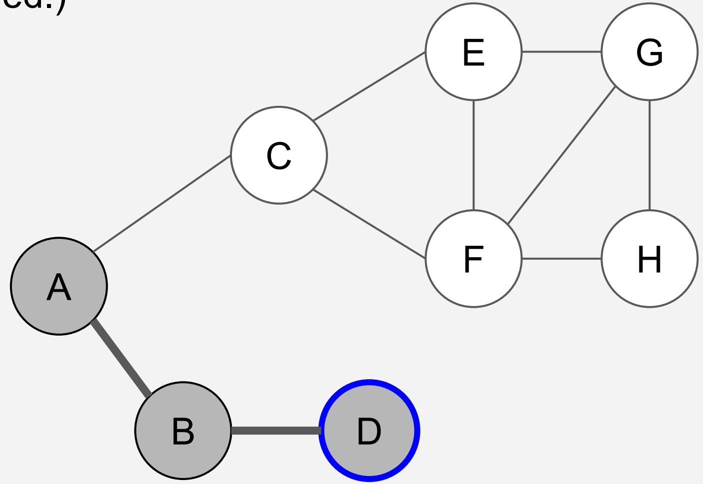
(top D)

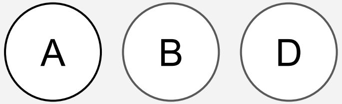




Depth-first search (DFS)

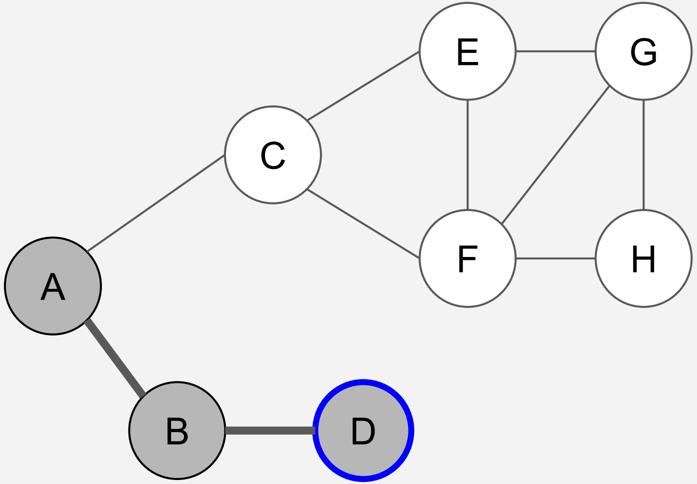
(color D Gray; D is discovered.)

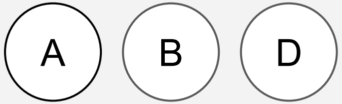




Depth-first search (DFS)

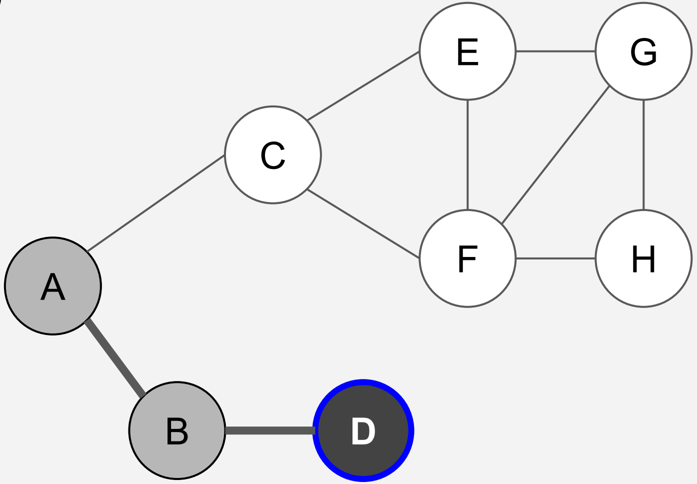
(explore edges from D)

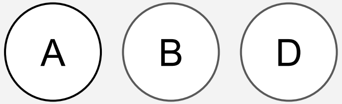




Depth-first search (DFS)

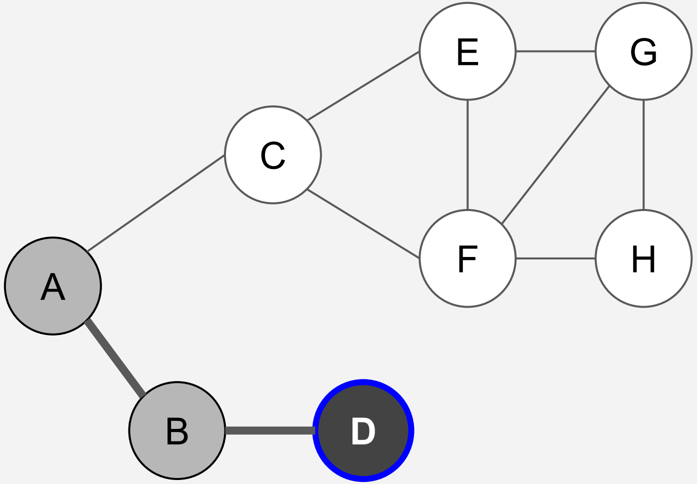
(color D Black; done with D)

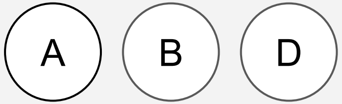




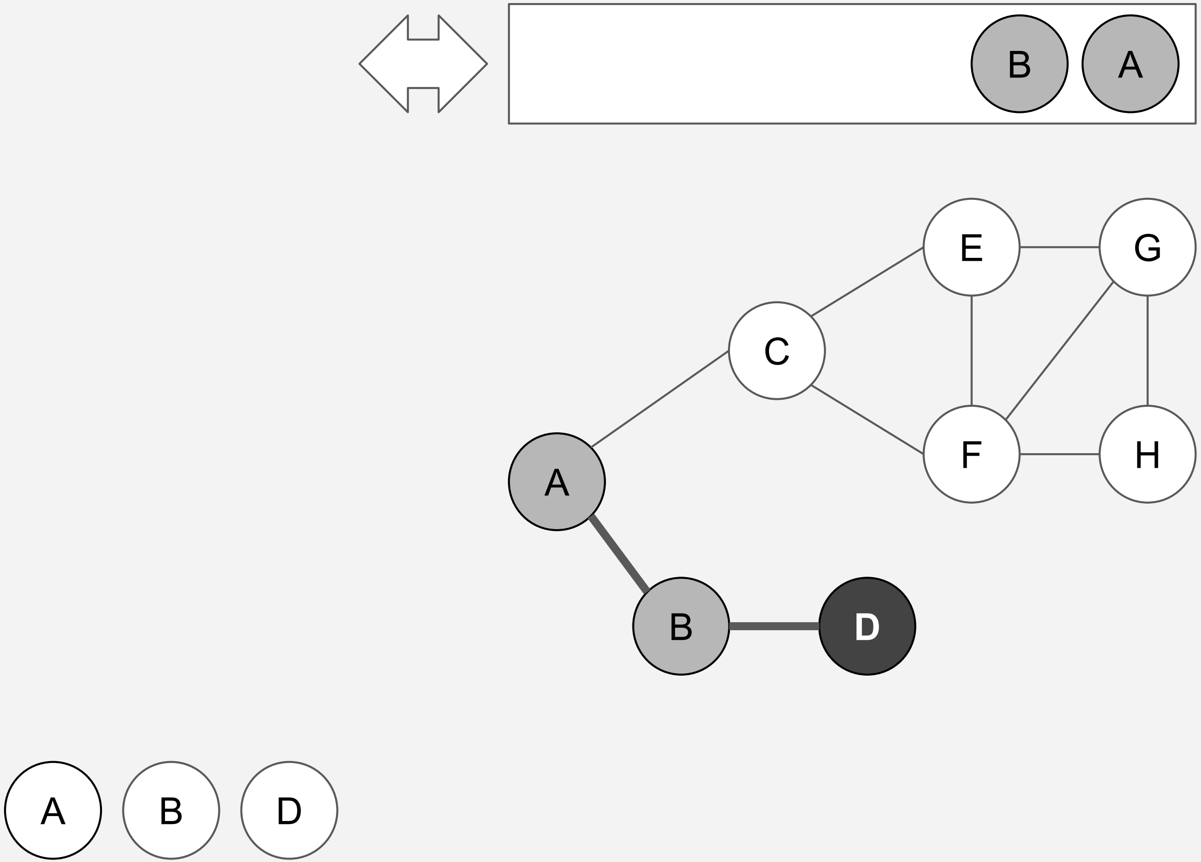
Depth-first search (DFS)

(pop D)



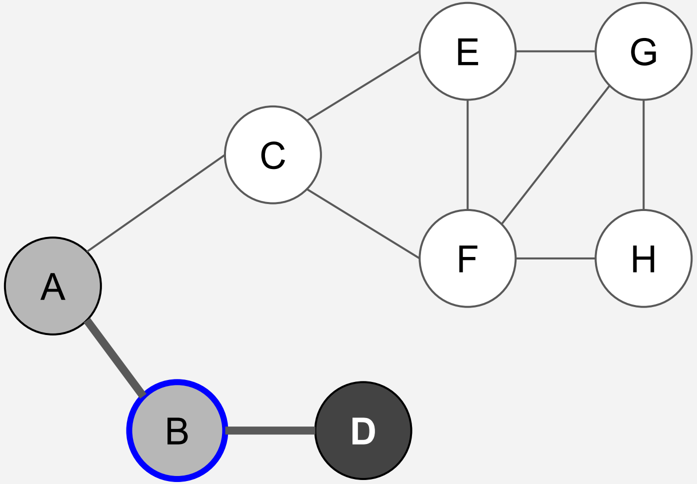


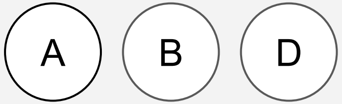
Depth-first search (DFS)



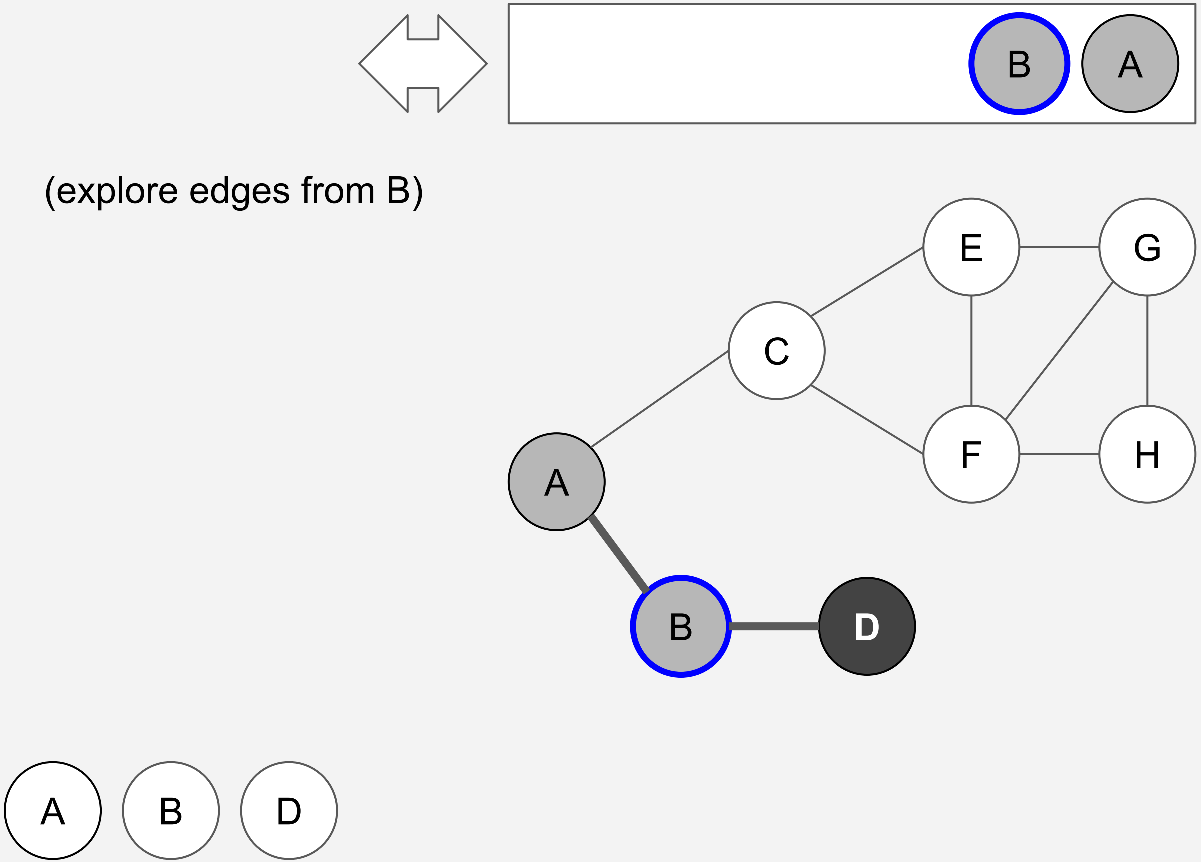
Depth-first search (DFS)

(top B)



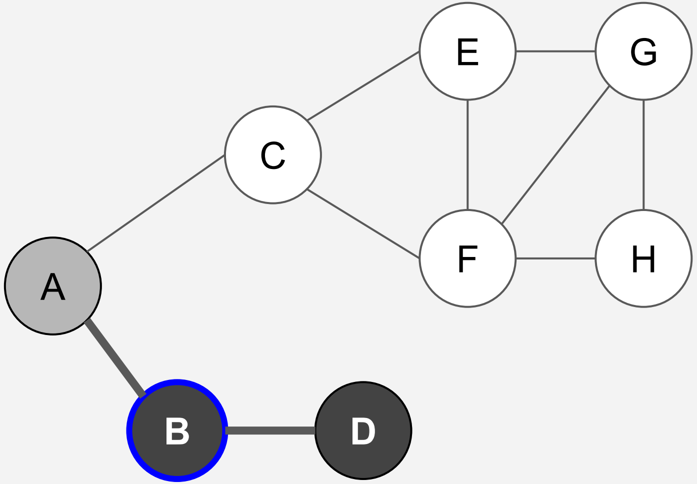


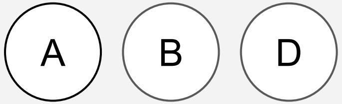
Depth-first search (DFS)



Depth-first search (DFS)

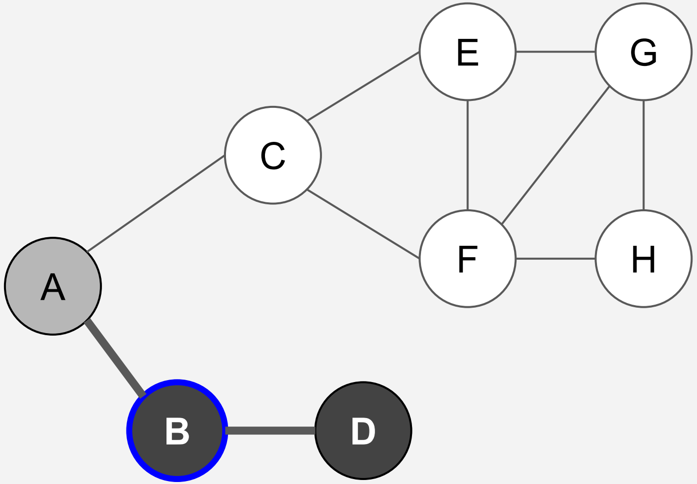
(color B Black; done with B)

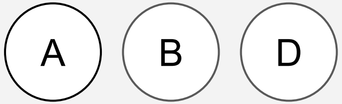




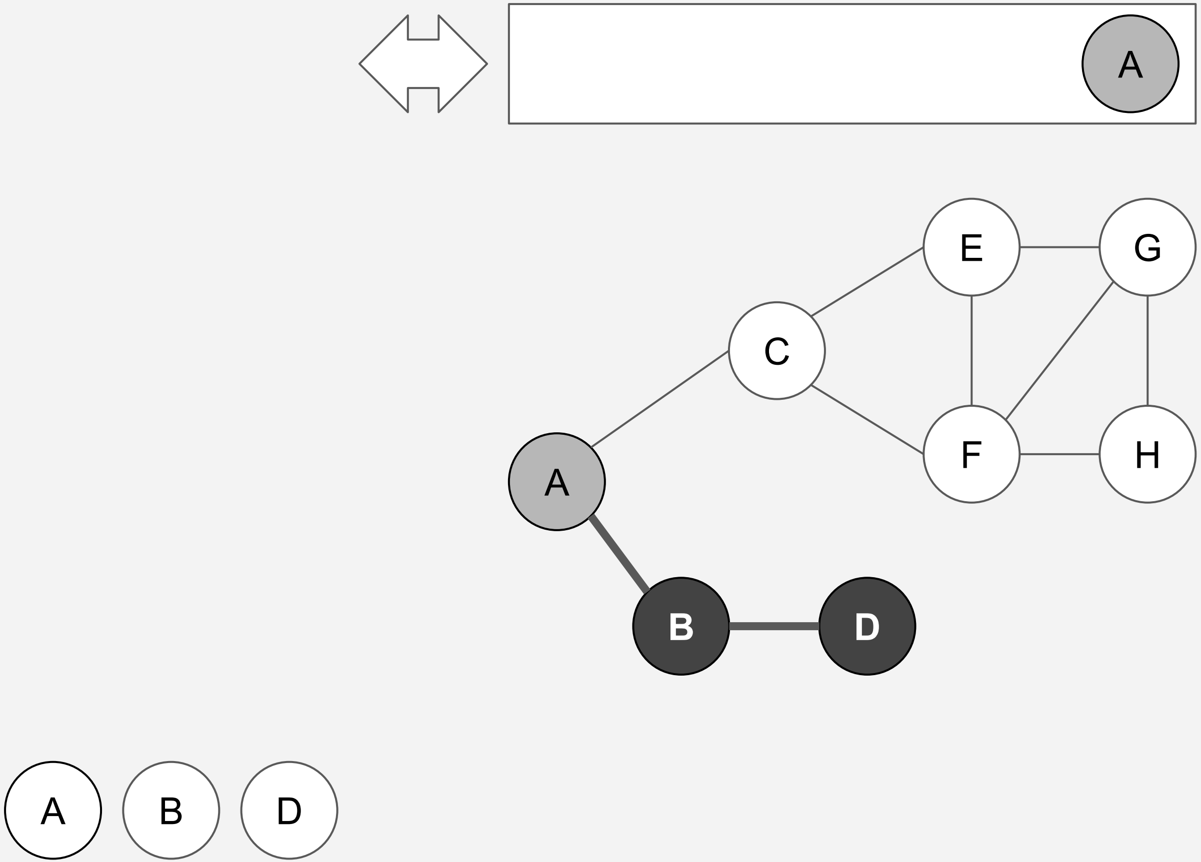
Depth-first search (DFS)

(pop B)

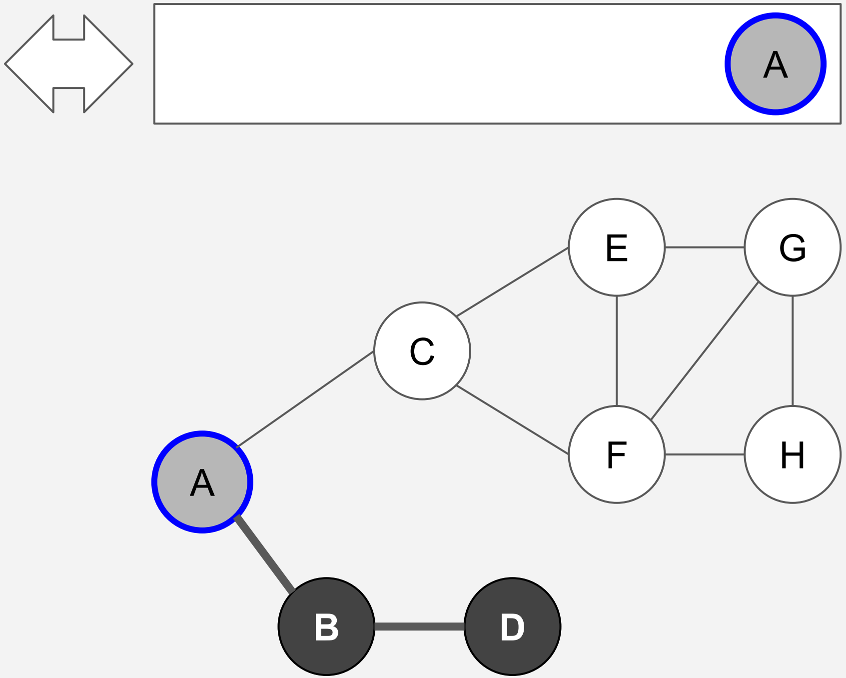




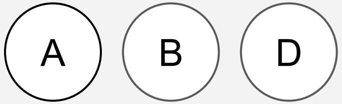
Depth-first search (DFS)



Depth-first search (DFS)

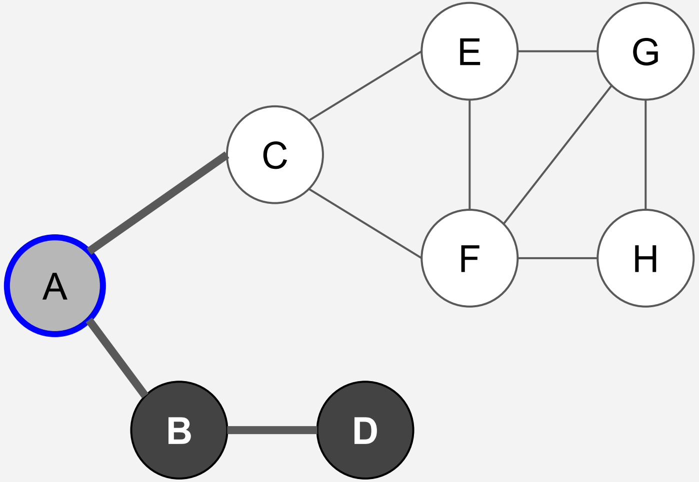


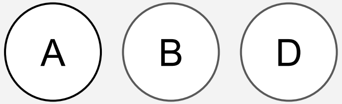
(top A)



Depth-first search (DFS)

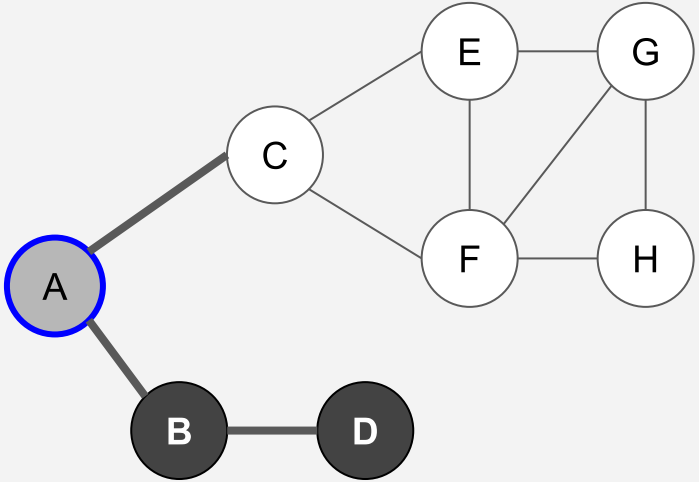
(explore edges from A)

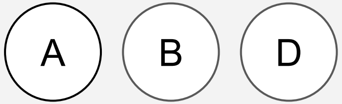




Depth-first search (DFS)

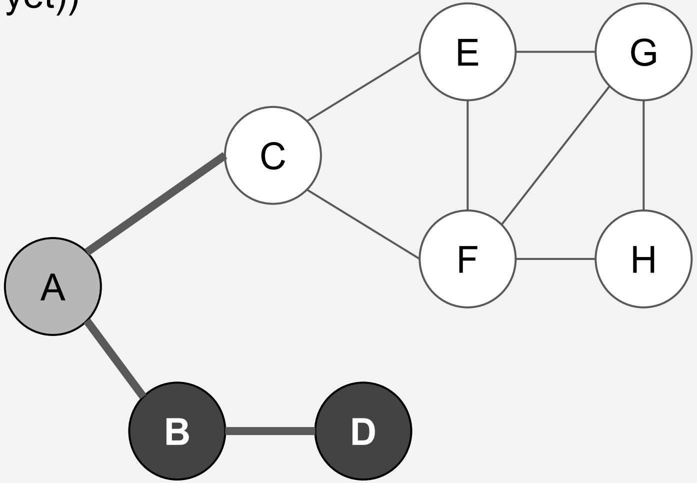
(push C)

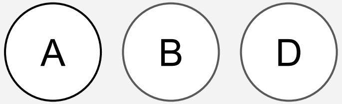




Depth-first search (DFS)

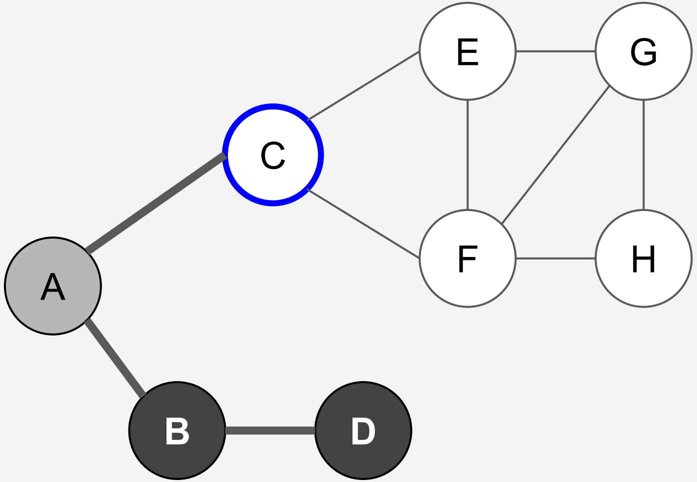
(start over; (not done with A yet))

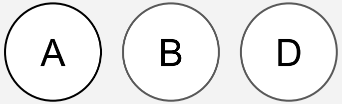




Depth-first search (DFS)

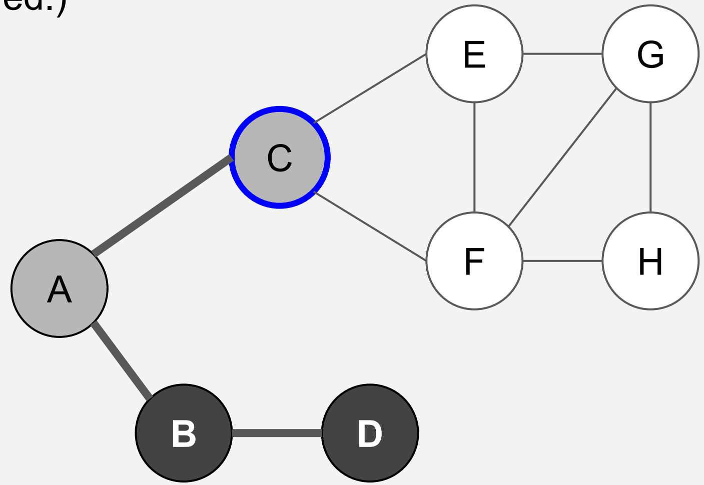
(top C)

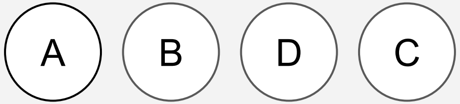




Depth-first search (DFS)

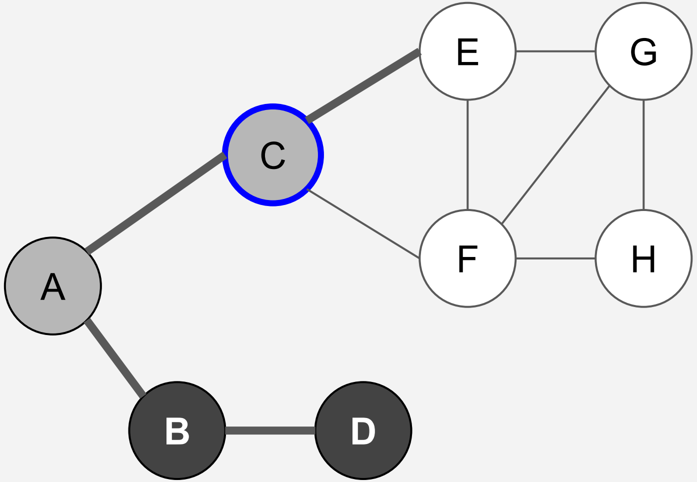
(color C Gray; C is discovered.)

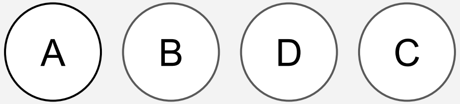




Depth-first search (DFS)

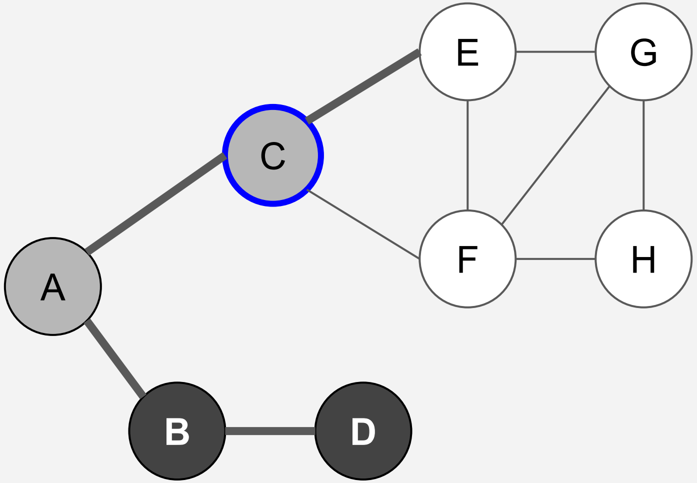
(explore edges from C)

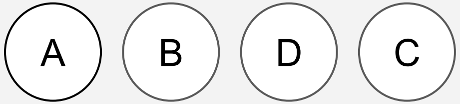




Depth-first search (DFS)

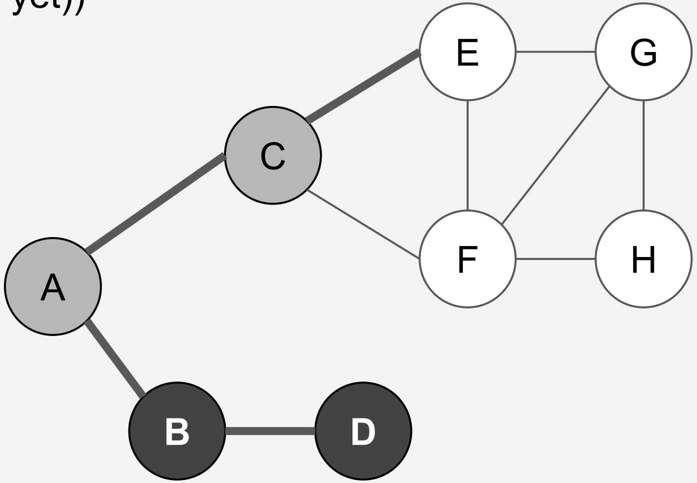
(push E)

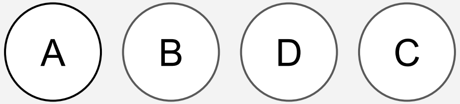




Depth-first search (DFS)

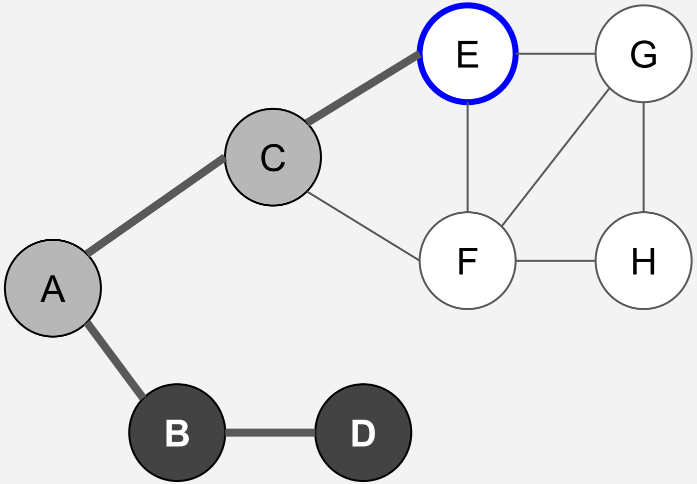
(start over; (not done with C yet))

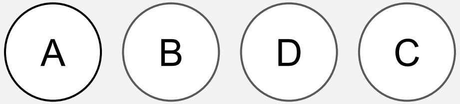




Depth-first search (DFS)

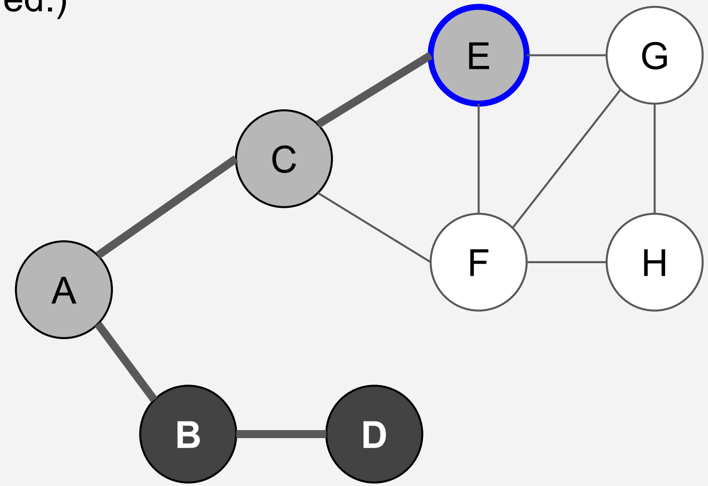
(top E)

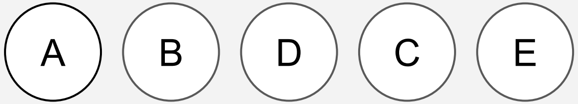




Depth-first search (DFS)

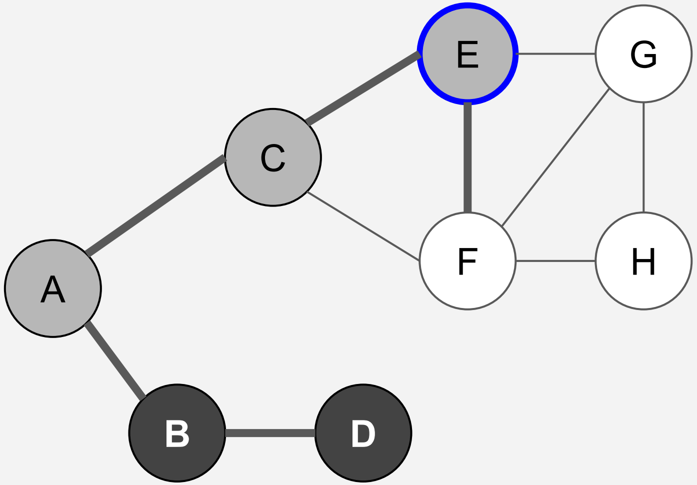
(color E Gray; E is discovered.)

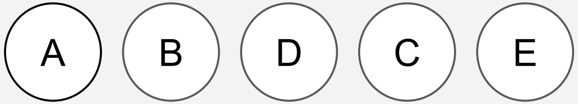




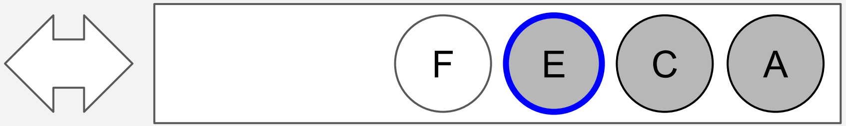
Depth-first search (DFS)

(explore edges from E)

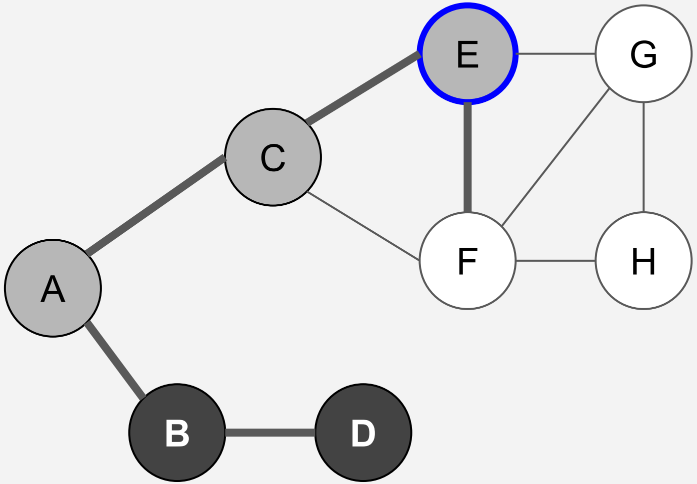


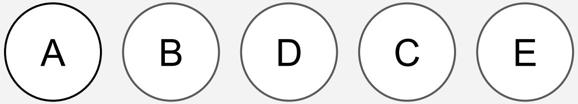


Depth-first search (DFS)

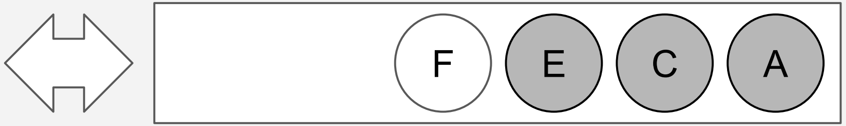


(push F)

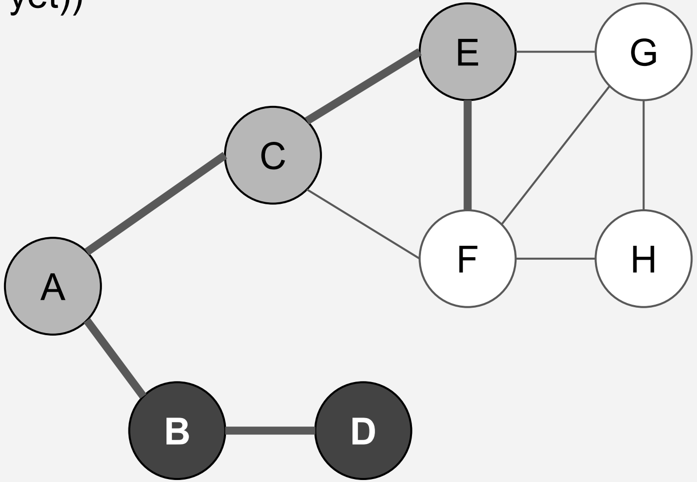


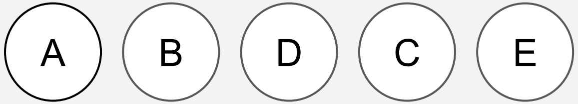


Depth-first search (DFS)

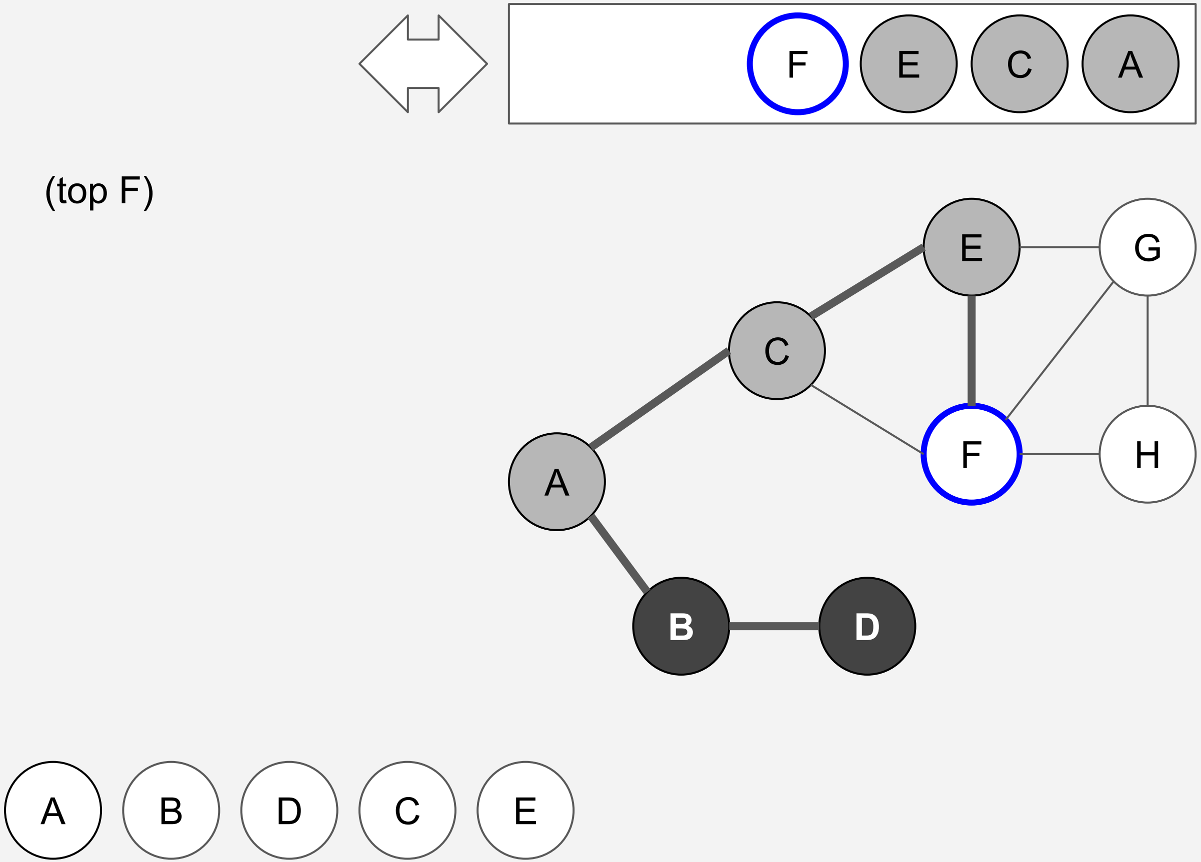


(start over; (not done with E yet))

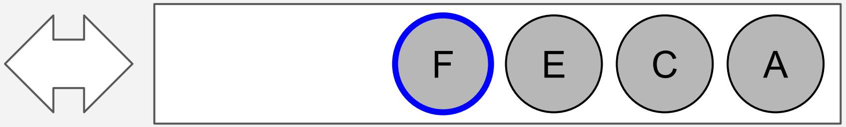




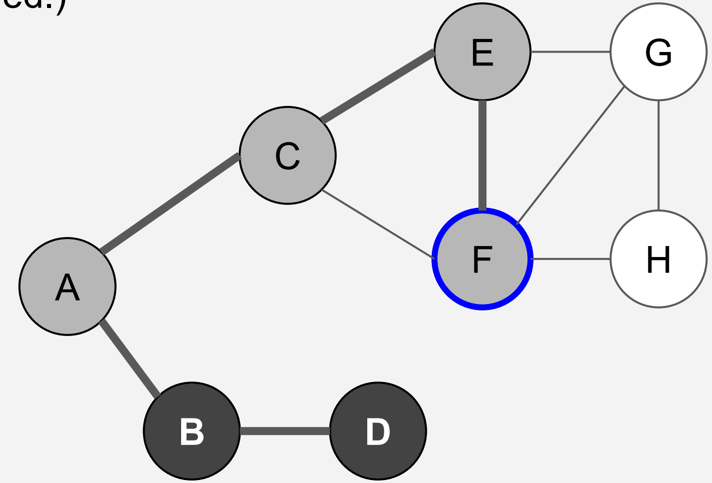
Depth-first search (DFS)

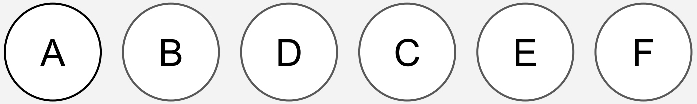


Depth-first search (DFS)

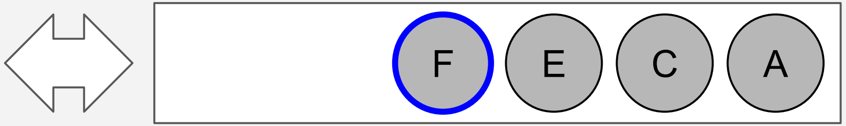


(color F Gray; F is discovered.)

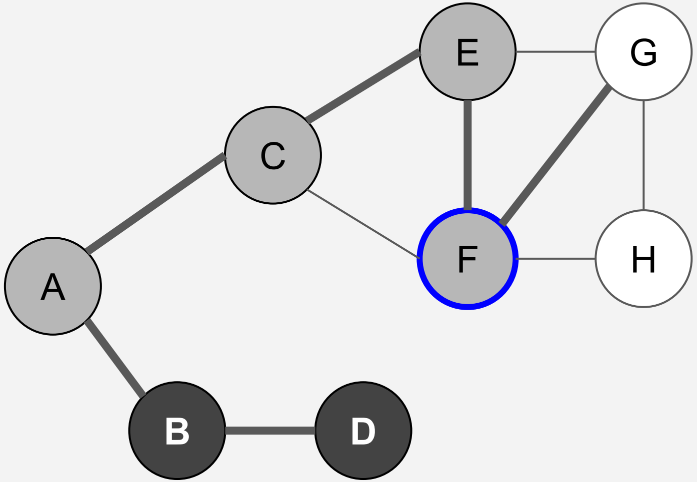


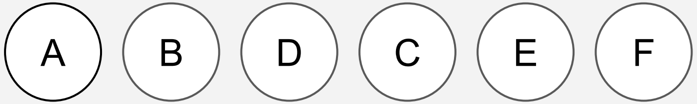


Depth-first search (DFS)

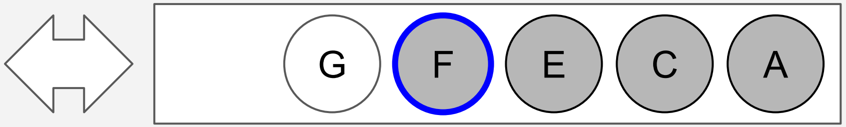


(explore edges from F)

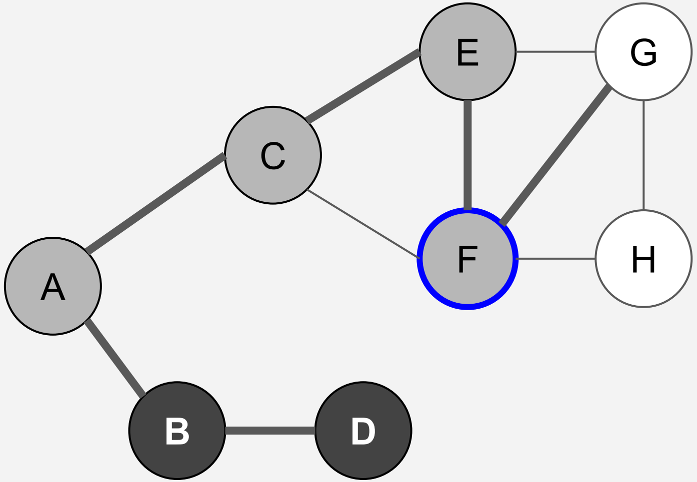


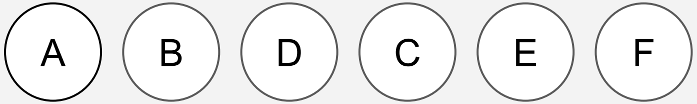


Depth-first search (DFS)

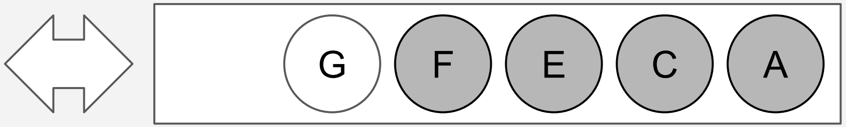


(push G)

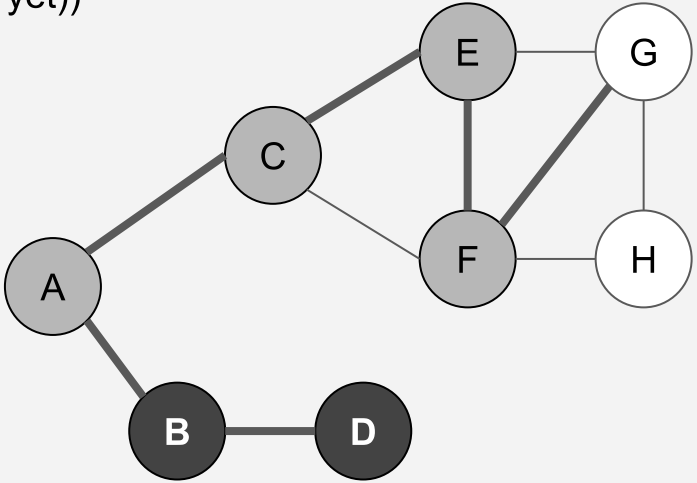


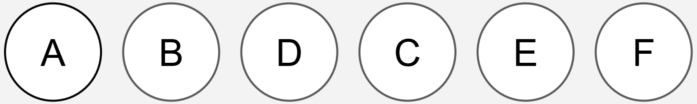


Depth-first search (DFS)

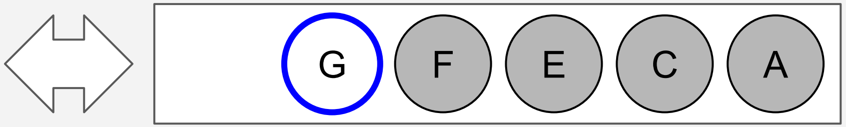


(start over; (not done with F yet))

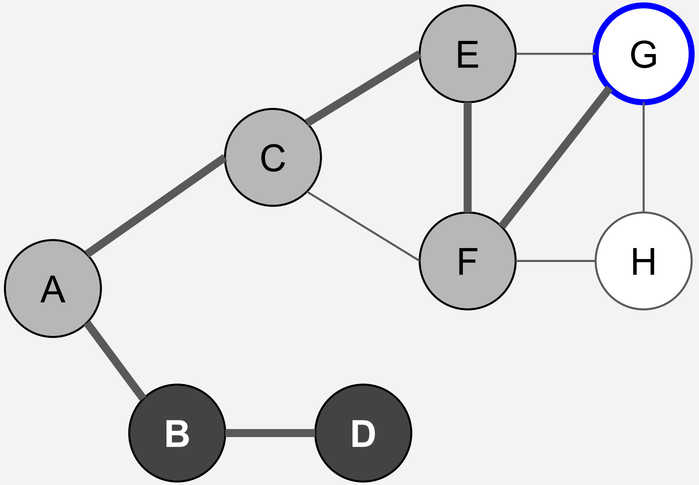


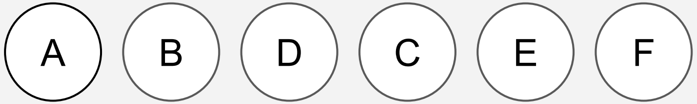


Depth-first search (DFS)

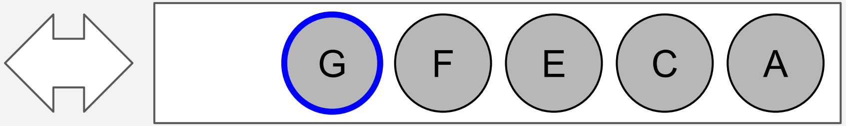


(top G)

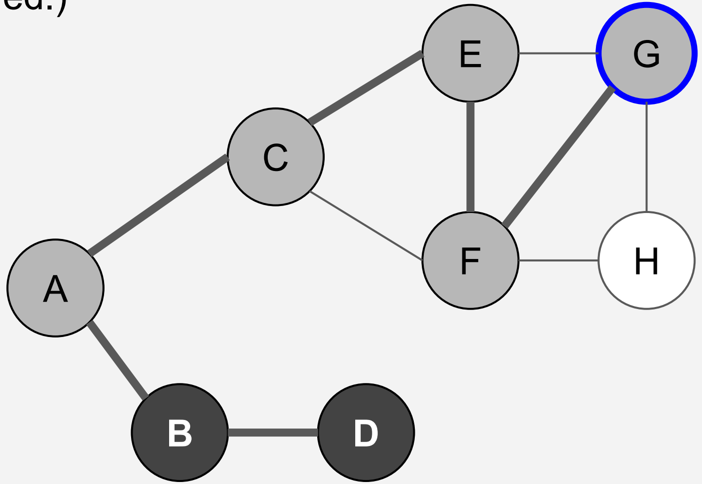


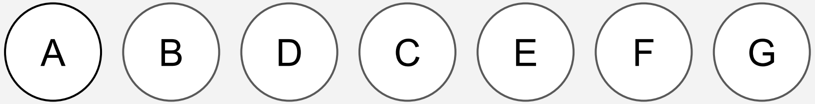


Depth-first search (DFS)

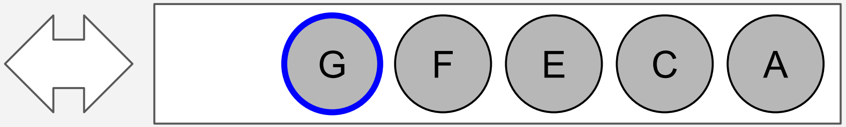


(color G Gray; G is discovered.)

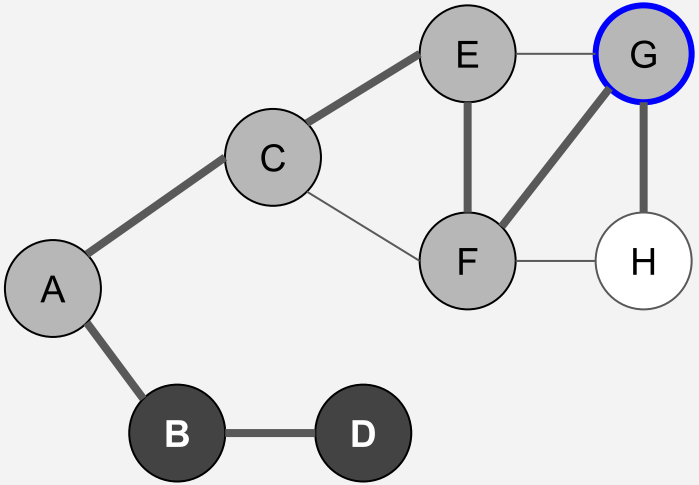


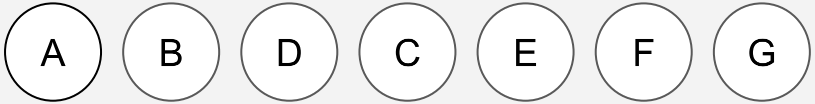


Depth-first search (DFS)

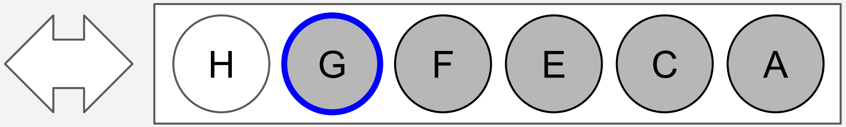


(explore edges from G)

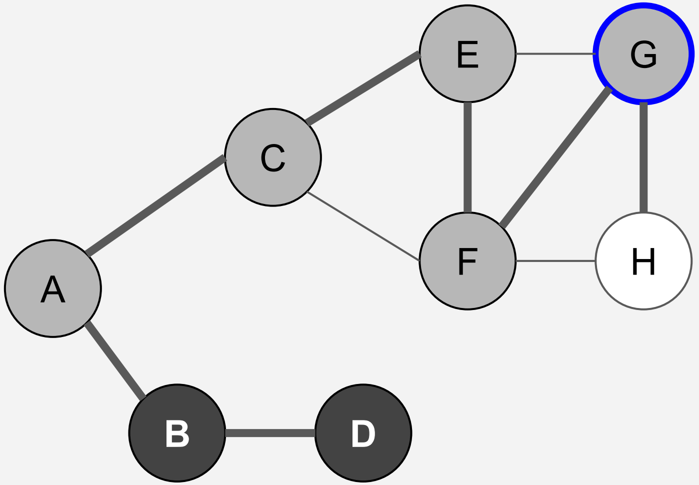


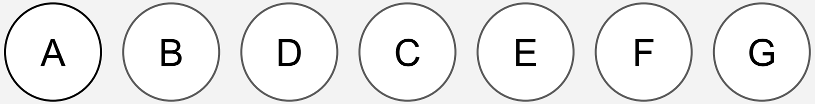


Depth-first search (DFS)

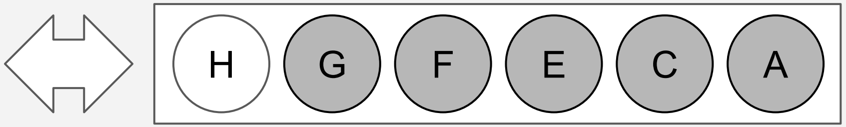


(push H)

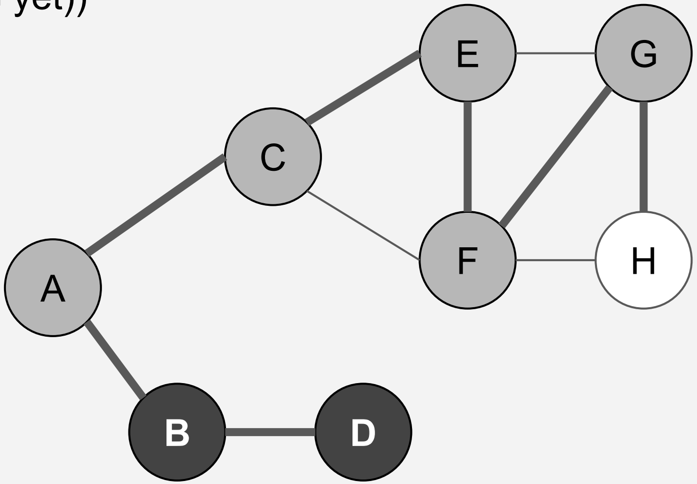


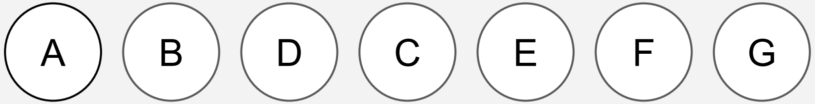


Depth-first search (DFS)

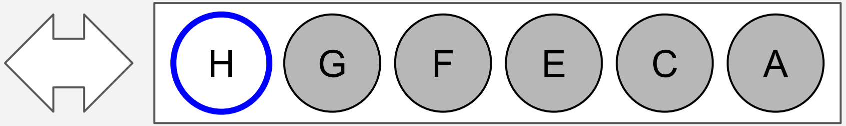


(start over; (not done with G yet))

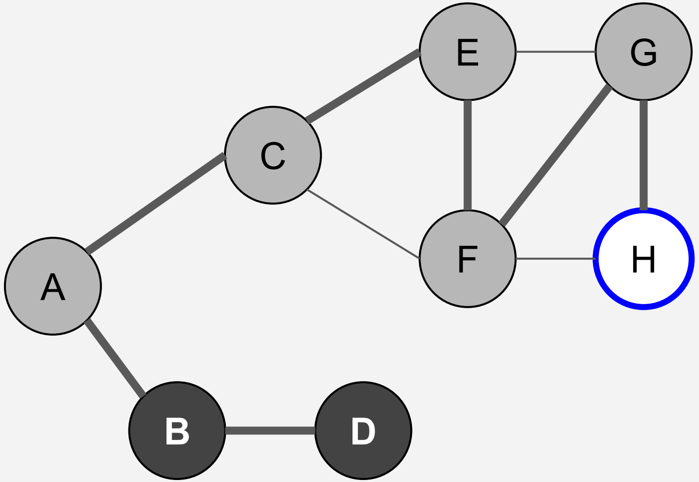


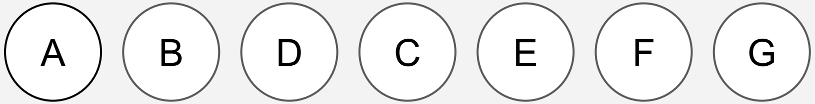


Depth-first search (DFS)

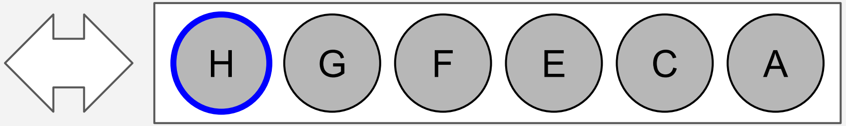


(top H)

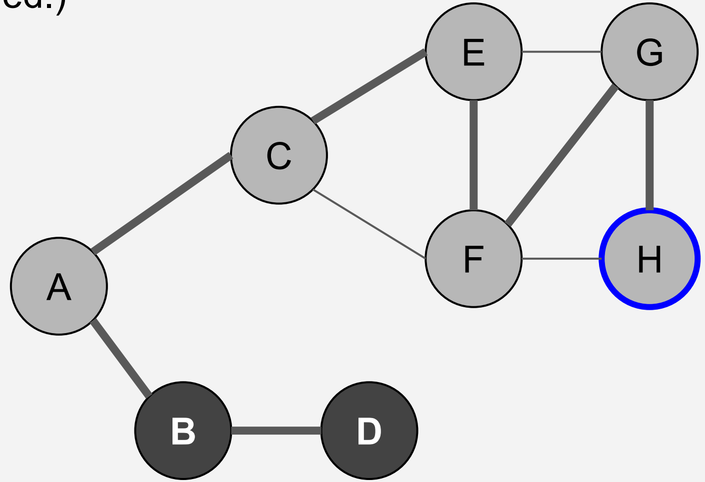


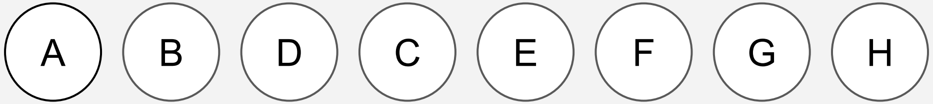


Depth-first search (DFS)

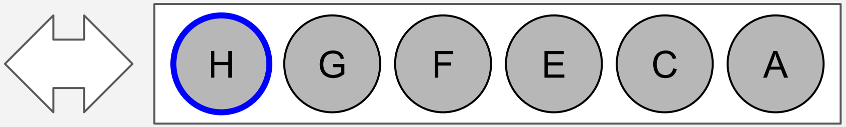


(color H Gray; H is discovered.)

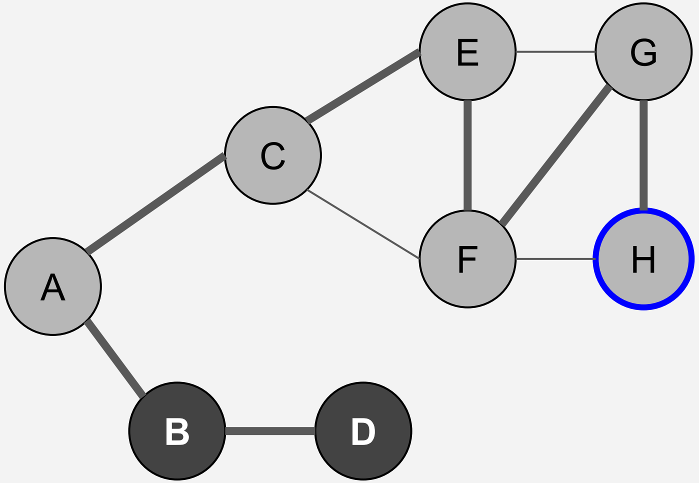


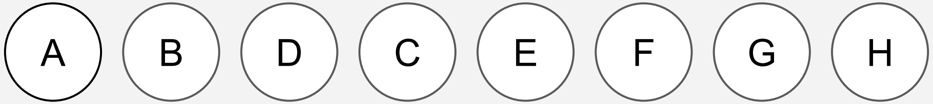


Depth-first search (DFS)

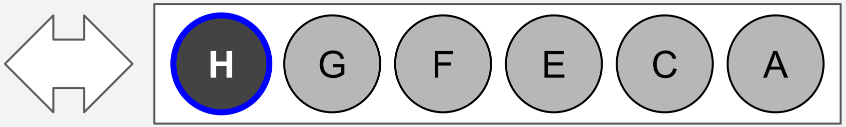


(explore edges from H)

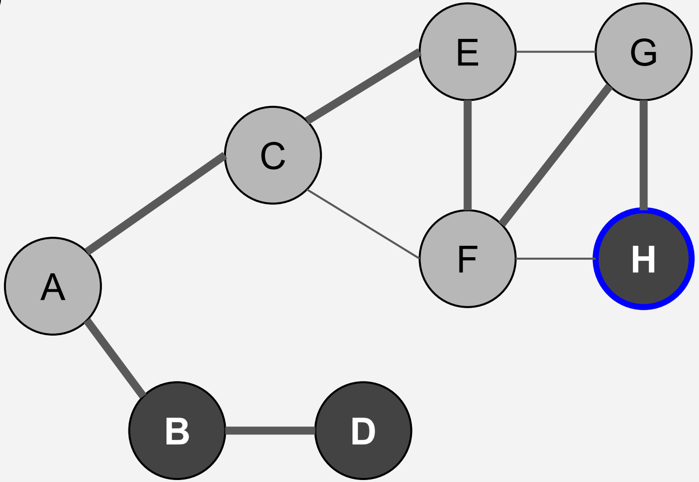


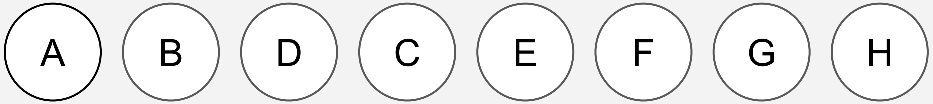


Depth-first search (DFS)

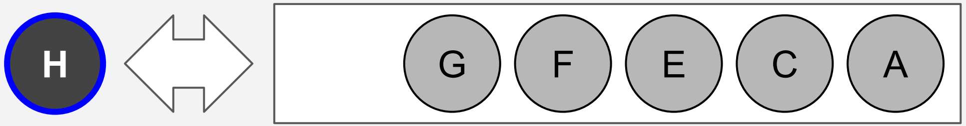


(color H Black; done with H)

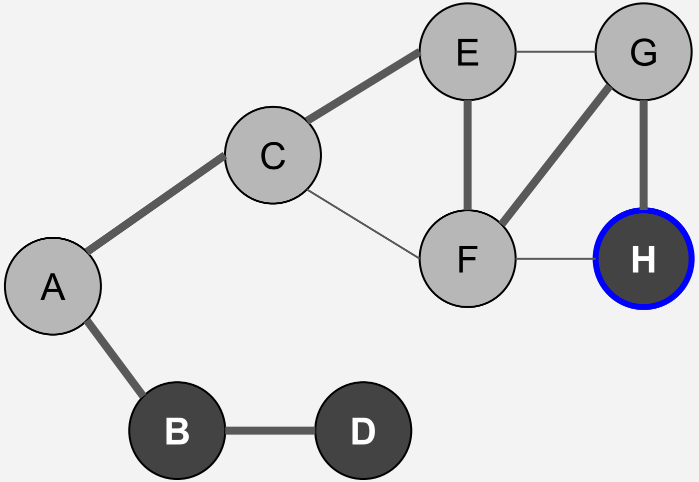


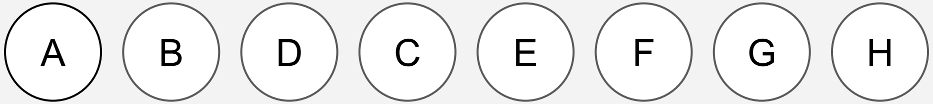


Depth-first search (DFS)

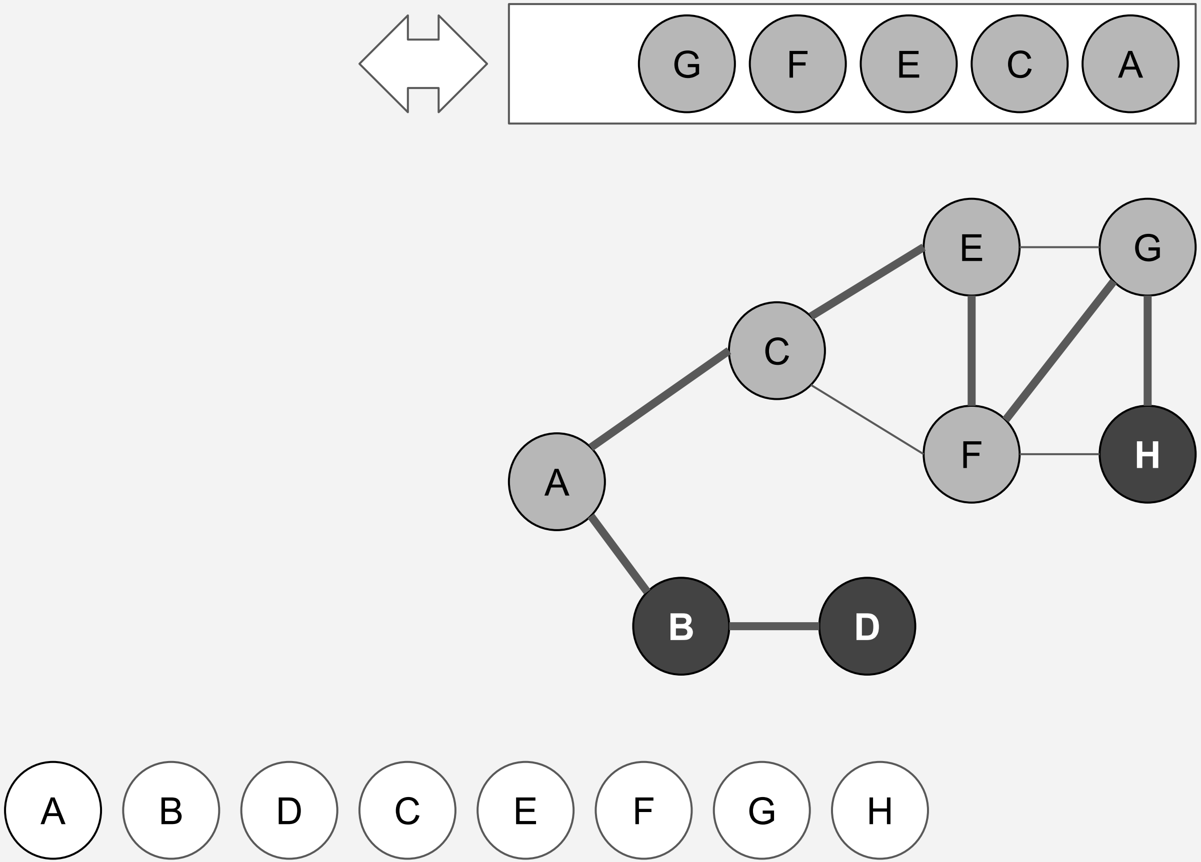


(pop H)

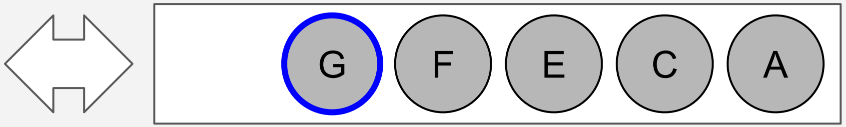




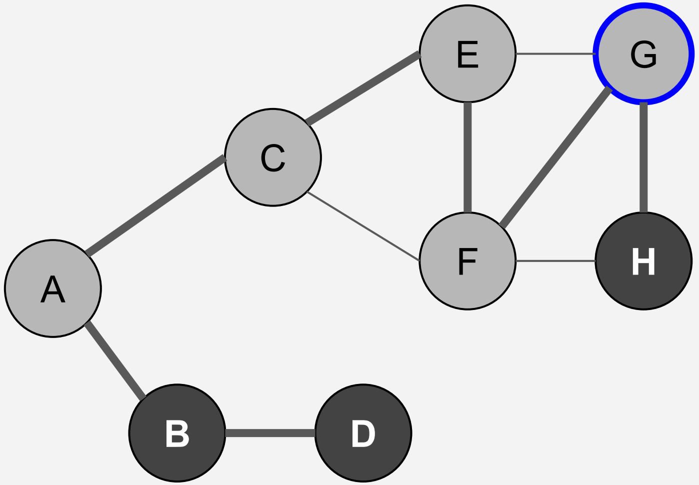
Depth-first search (DFS)

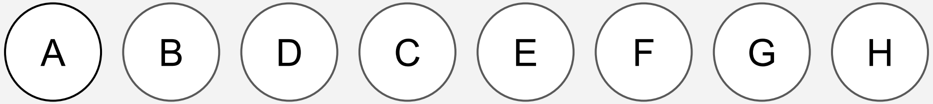


Depth-first search (DFS)

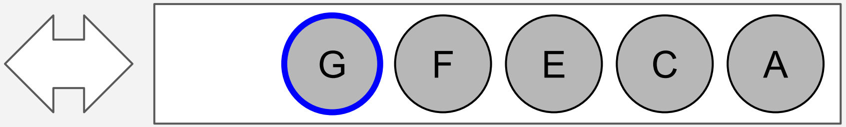


(top G)

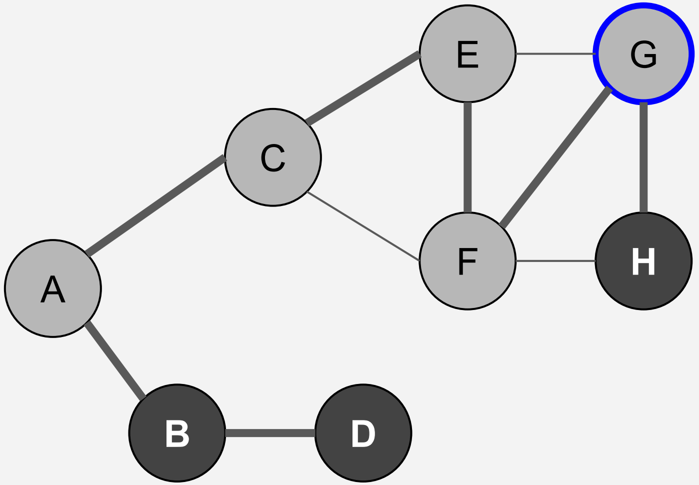


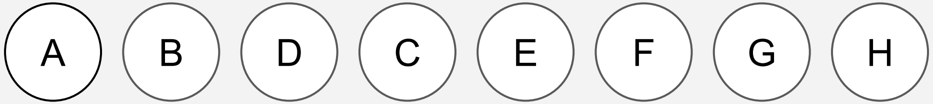


Depth-first search (DFS)

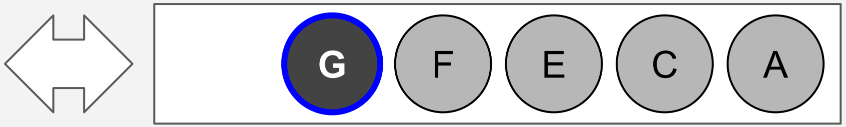


(explore edges from G)

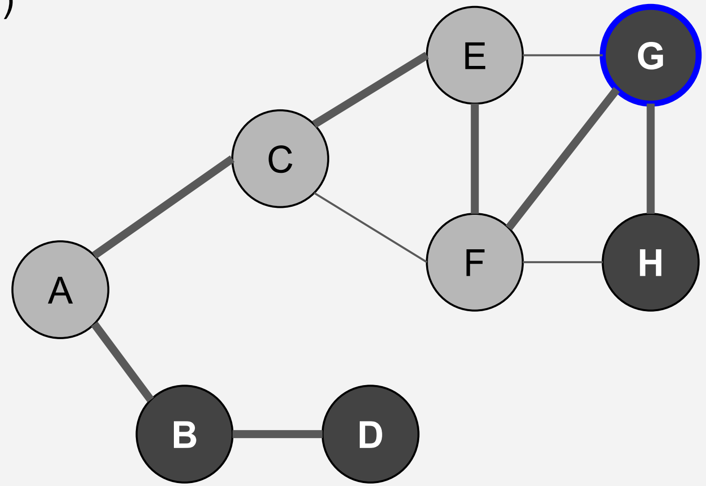


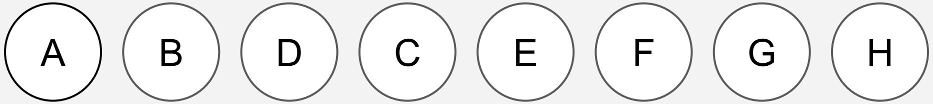


Depth-first search (DFS)

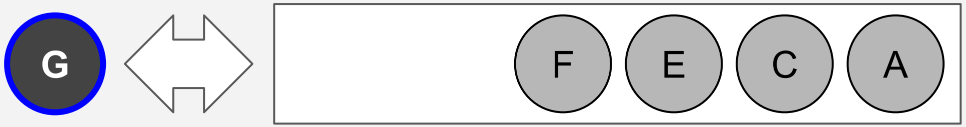


(color G Black; done with G)

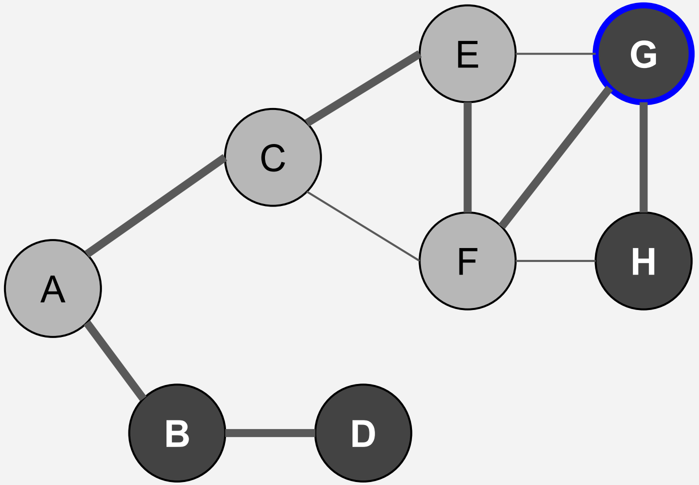


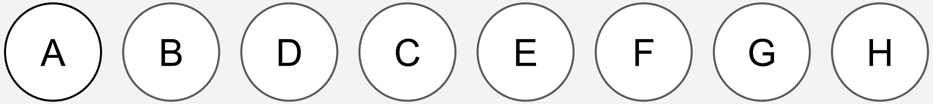


Depth-first search (DFS)

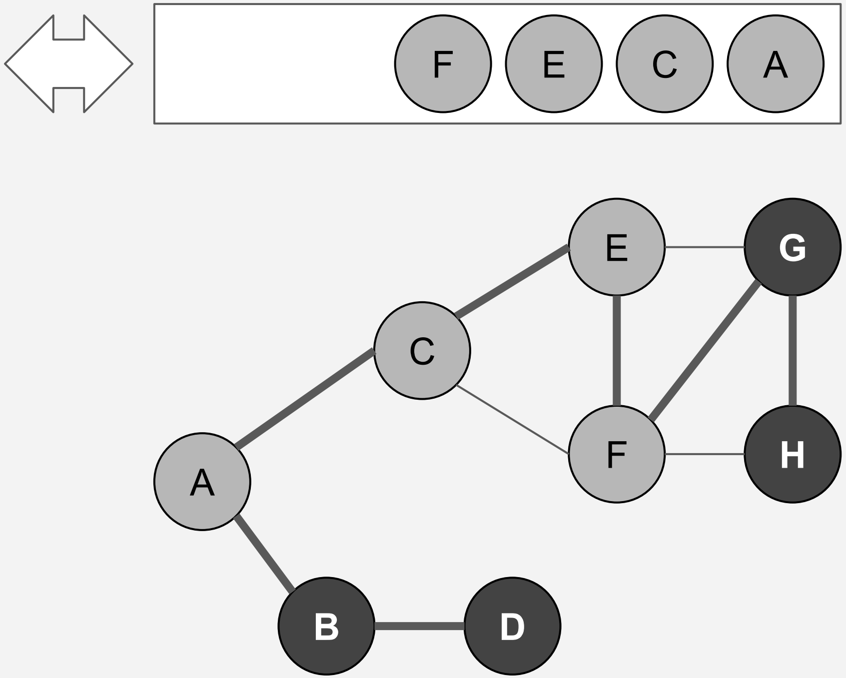


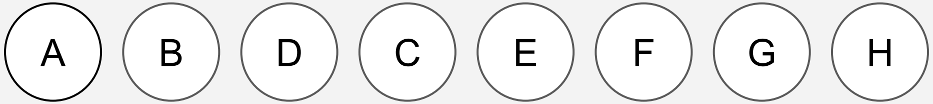
(pop G)



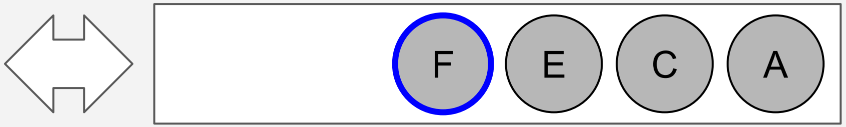


Depth-first search (DFS)

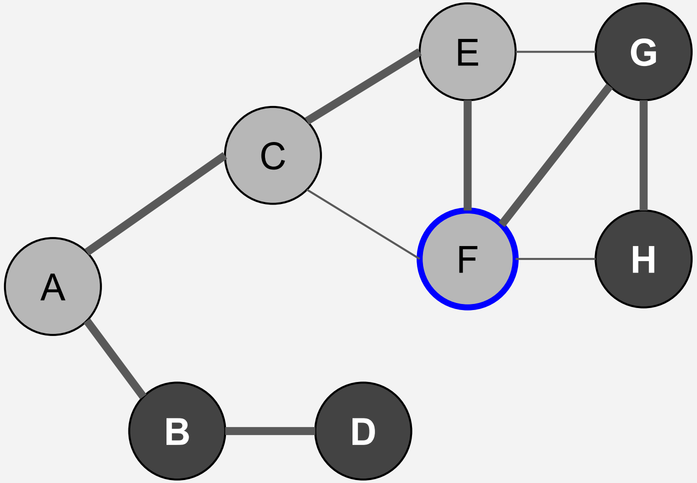


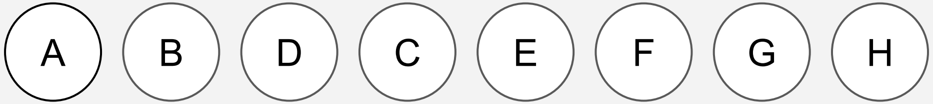


Depth-first search (DFS)

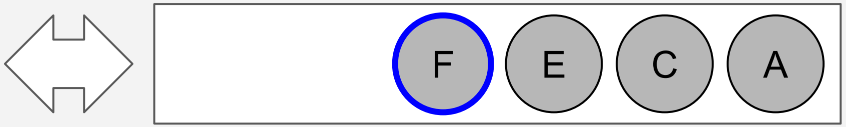


(top F)

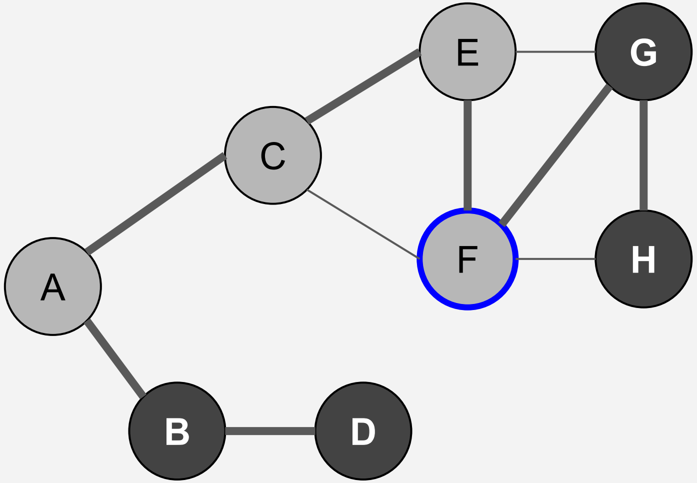


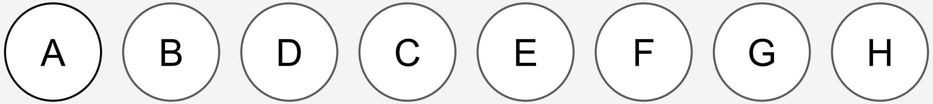


Depth-first search (DFS)

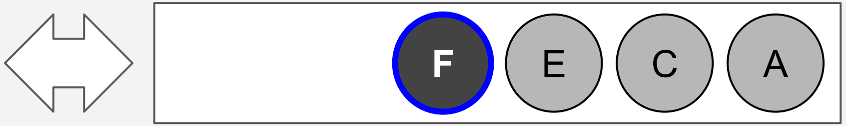


(explore edges from F)

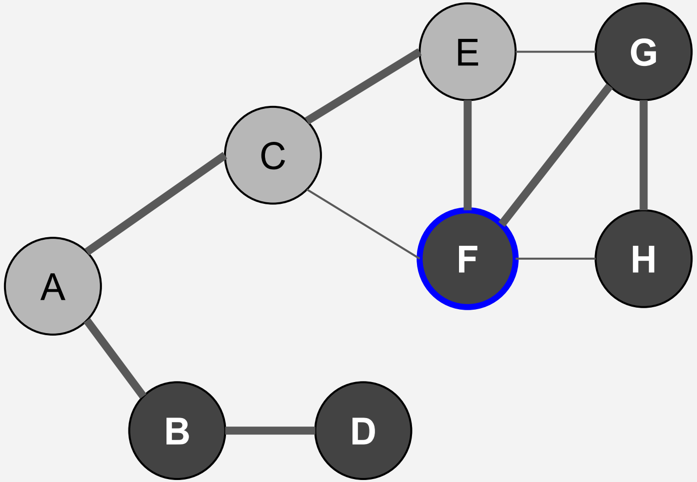


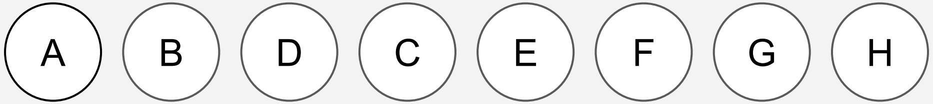


Depth-first search (DFS)



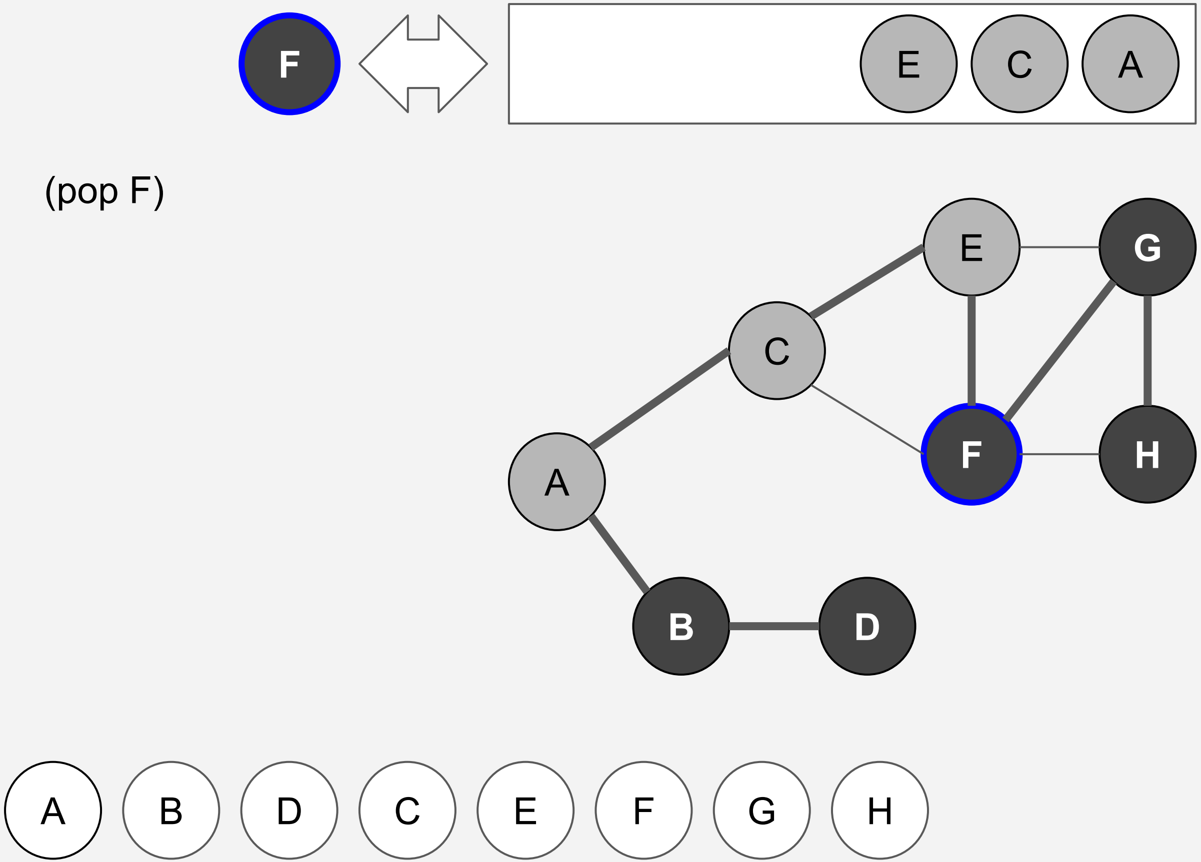
(color F Black; done with F)



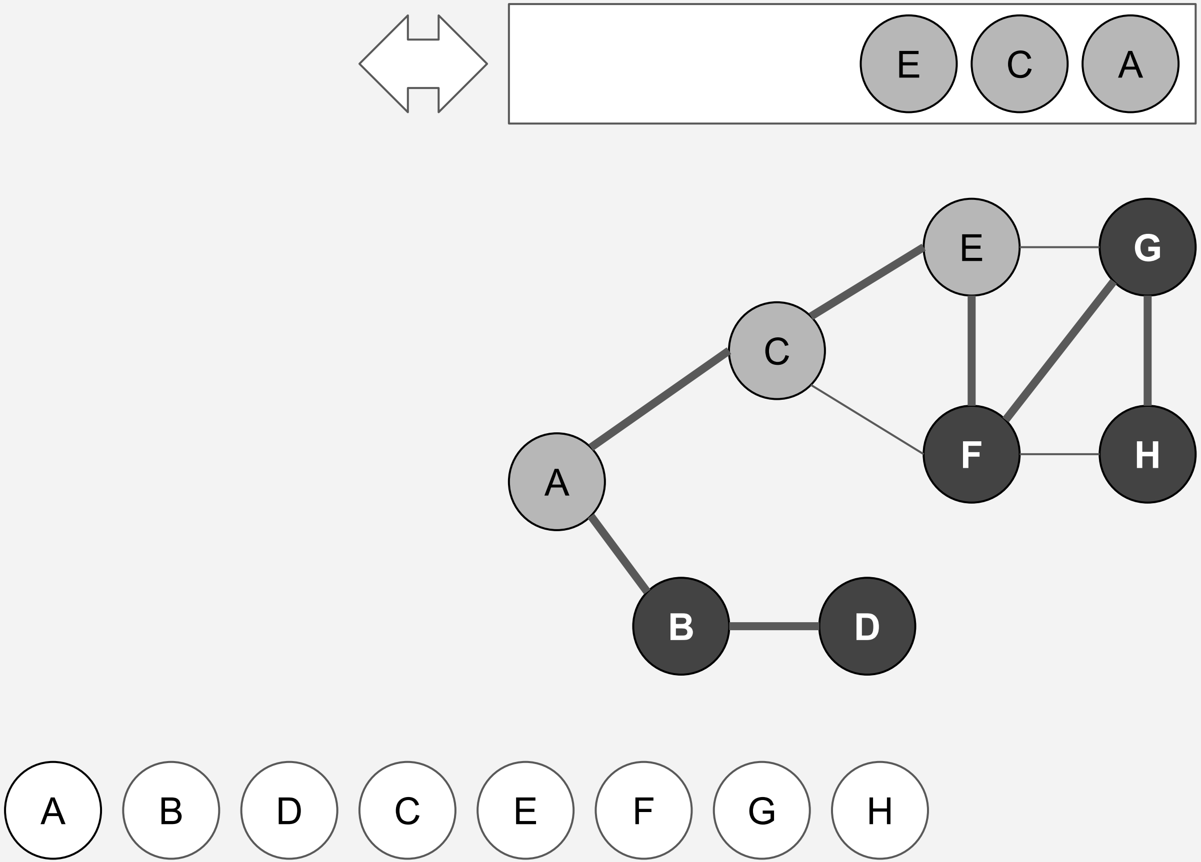


Depth-first search (DFS)

(pop F)

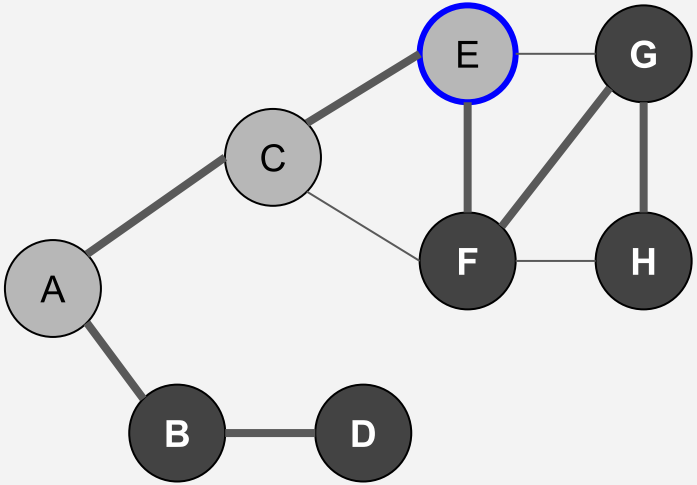


Depth-first search (DFS)



Depth-first search (DFS)

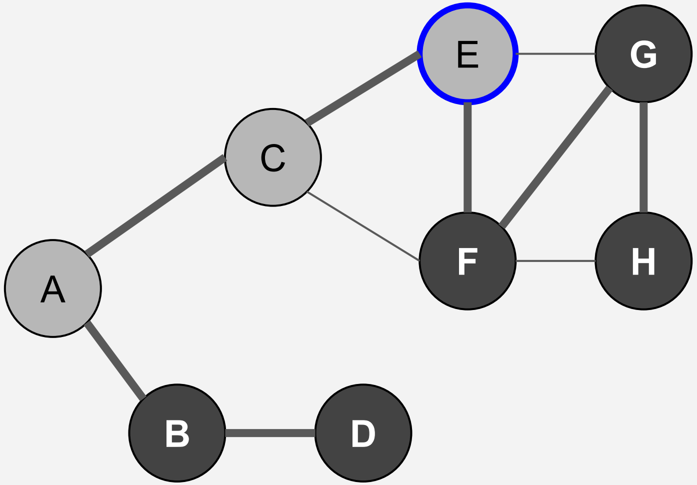
(top E)

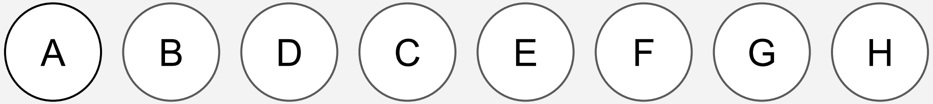


A B D C E F G H

Depth-first search (DFS)

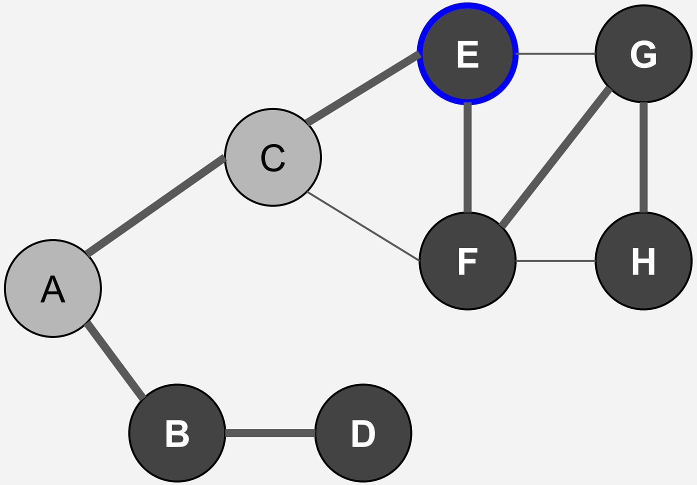
(explore edges from E)

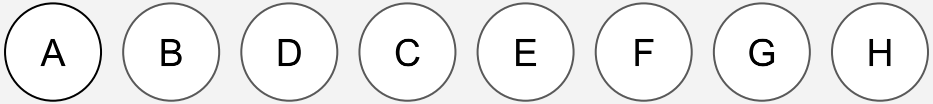




Depth-first search (DFS)

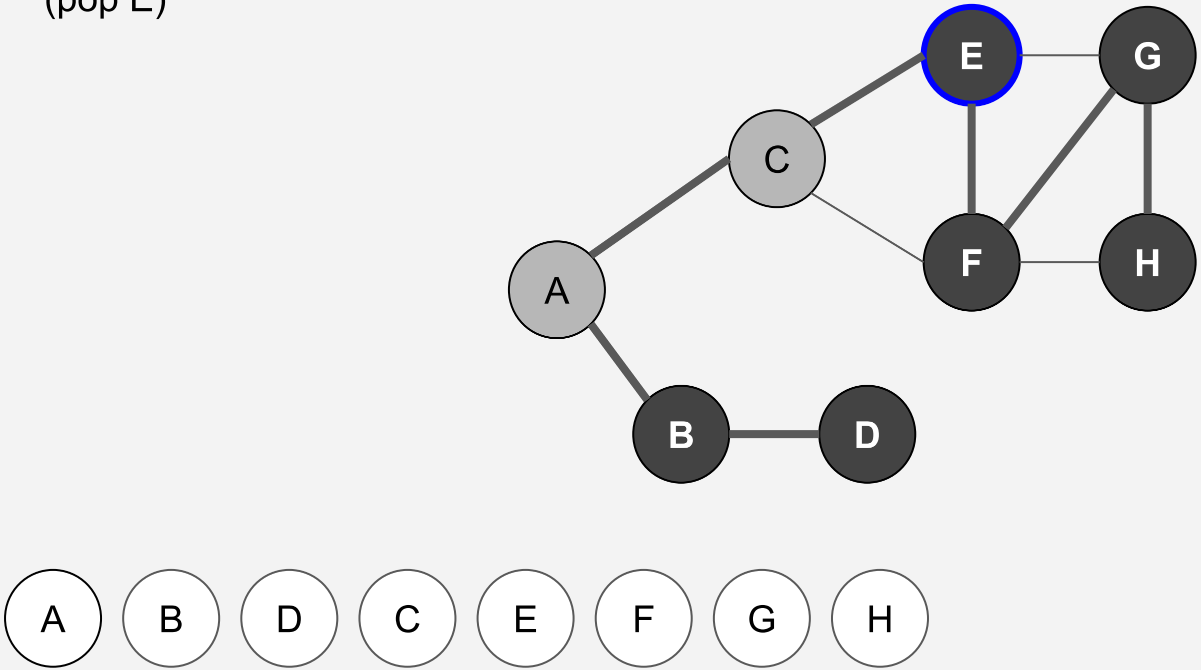
(color E Black; done with E)



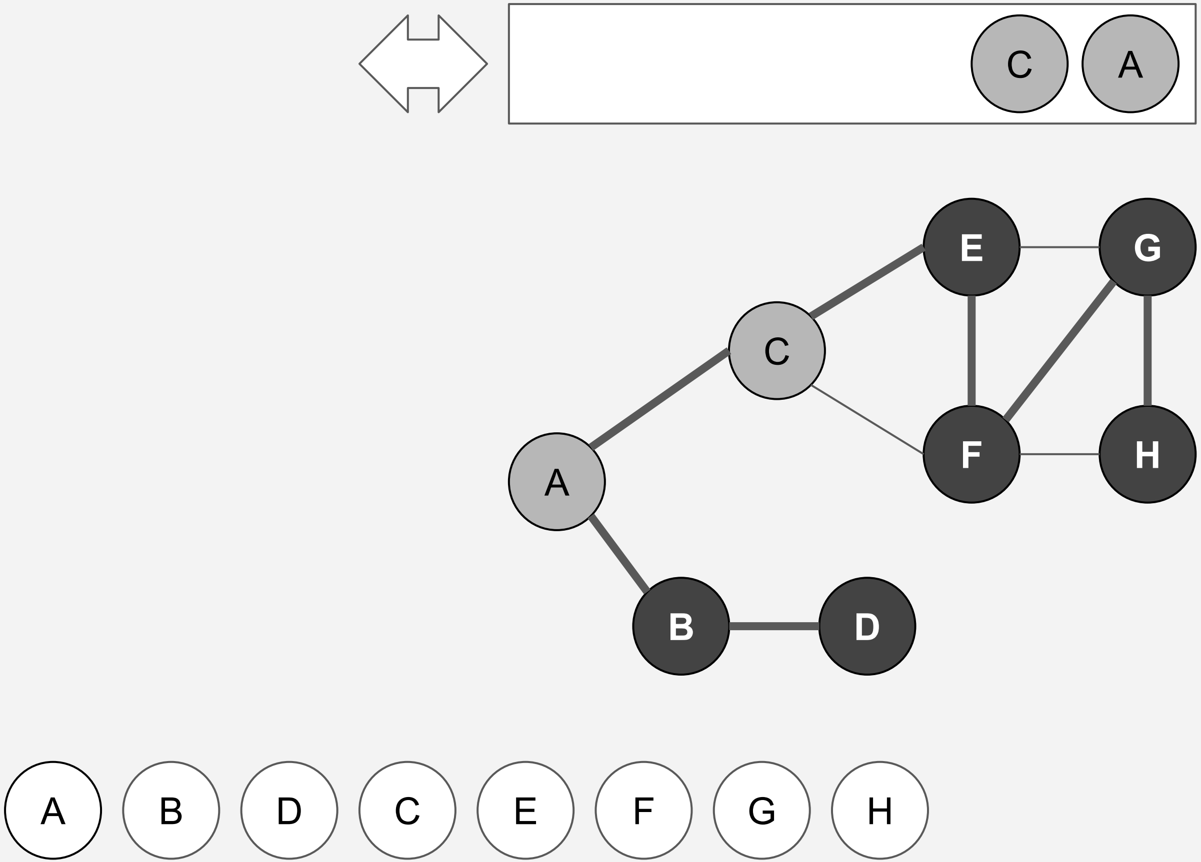


Depth-first search (DFS)

(pop E)

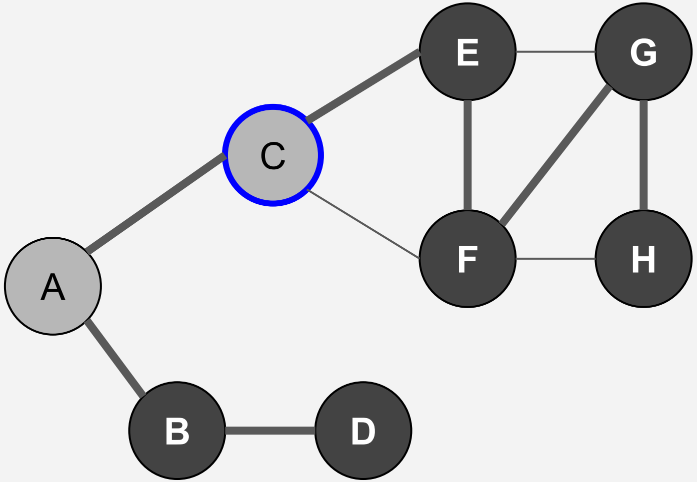


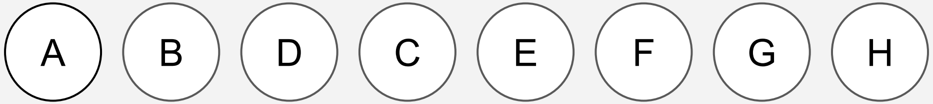
Depth-first search (DFS)



Depth-first search (DFS)

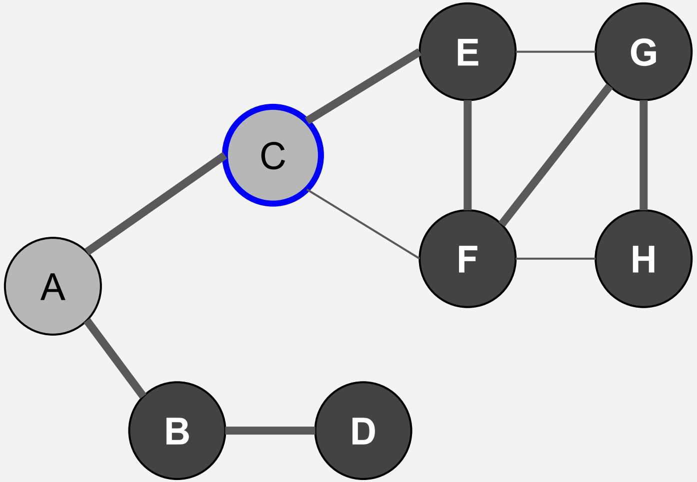
(top C)

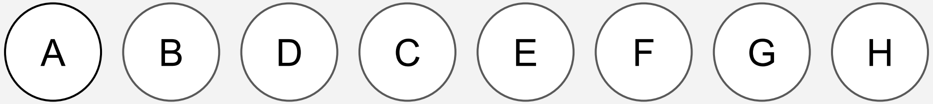




Depth-first search (DFS)

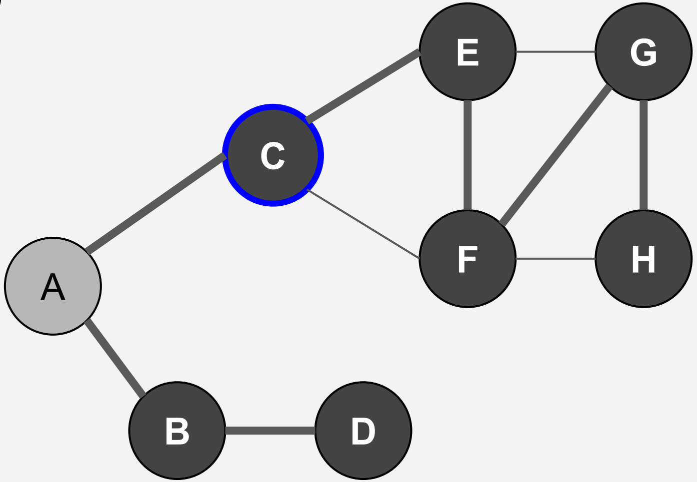
(explore edges from C)

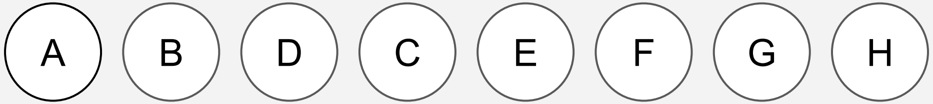




Depth-first search (DFS)

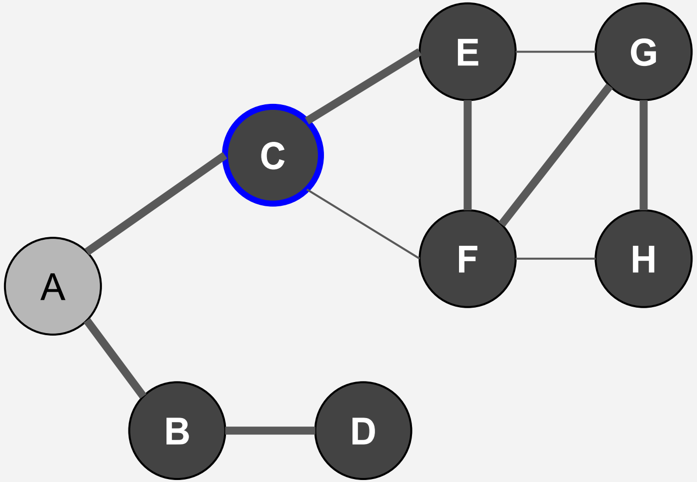
(color C Black; done with C)

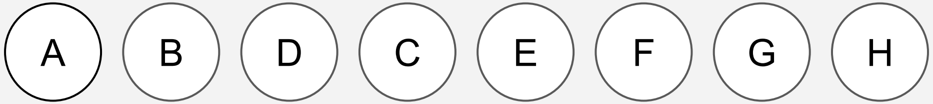




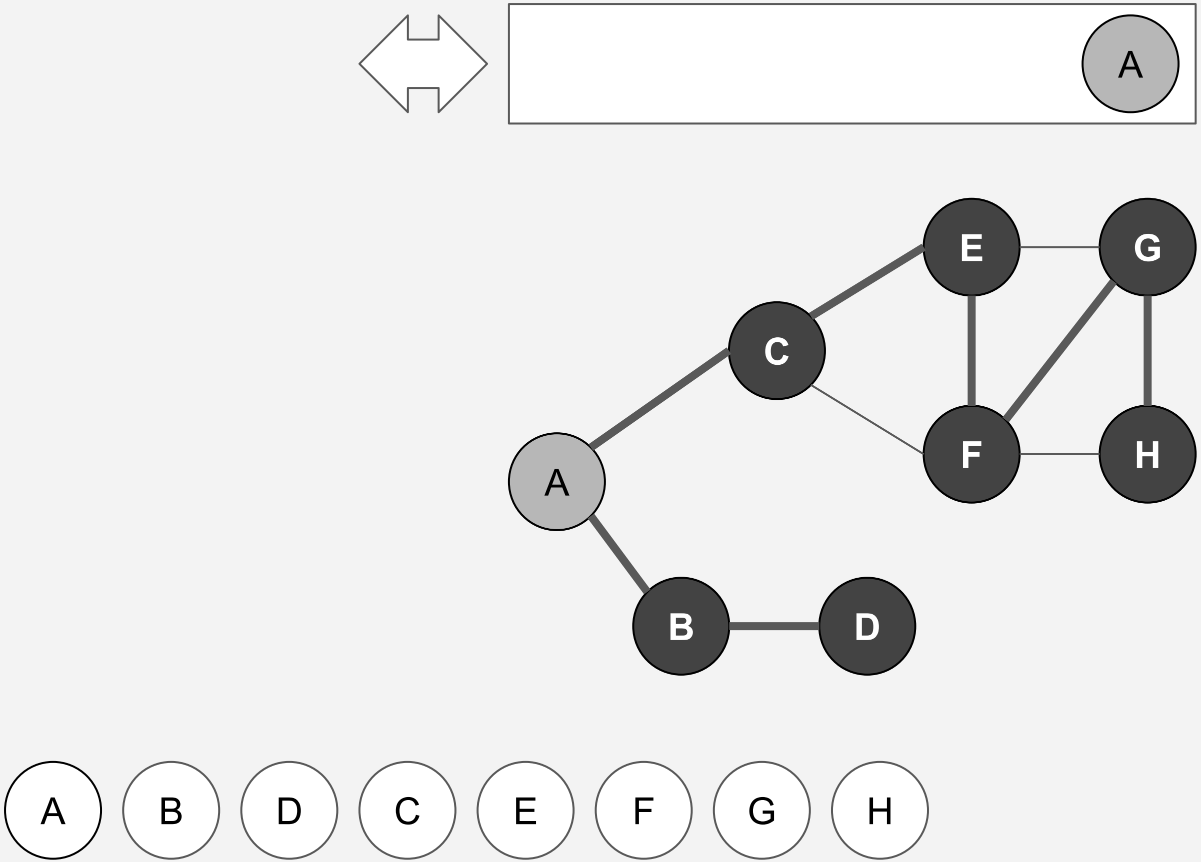
Depth-first search (DFS)

(pop C)



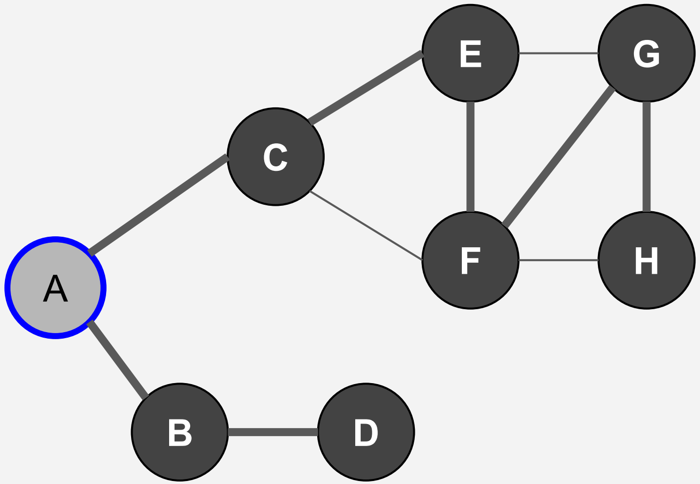


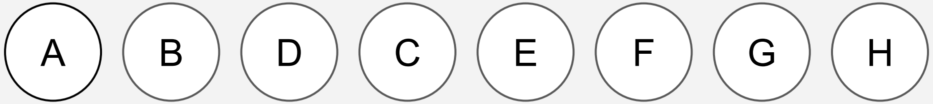
Depth-first search (DFS)



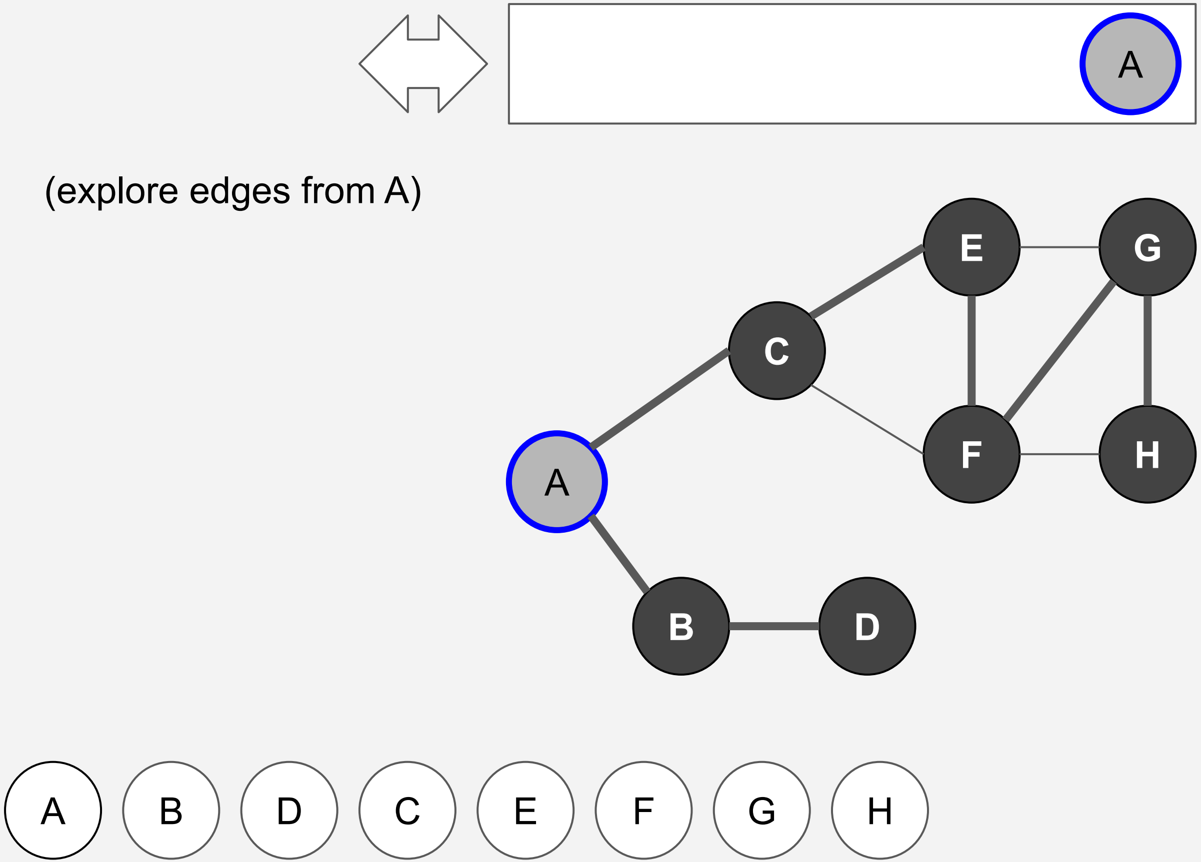
Depth-first search (DFS)

(top A)



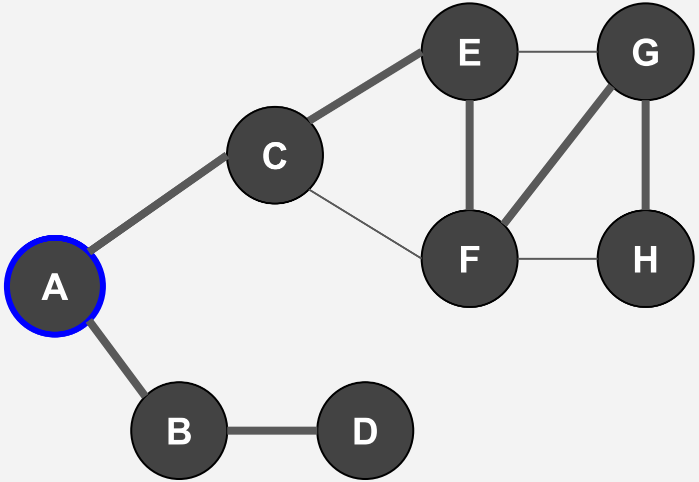


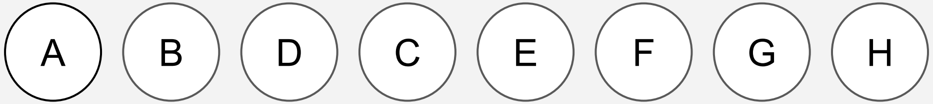
Depth-first search (DFS)



Depth-first search (DFS)

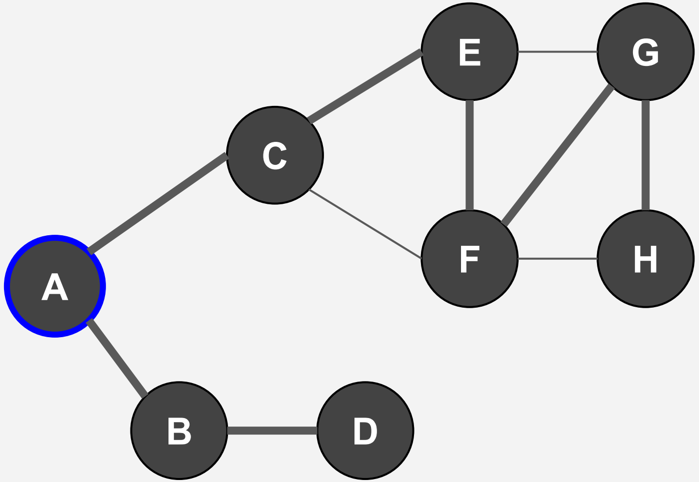
(color A Black; done with A)

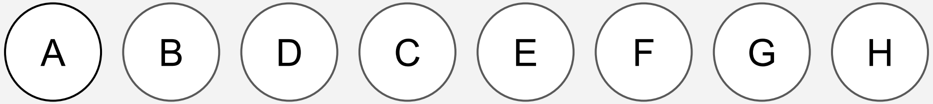




Depth-first search (DFS)

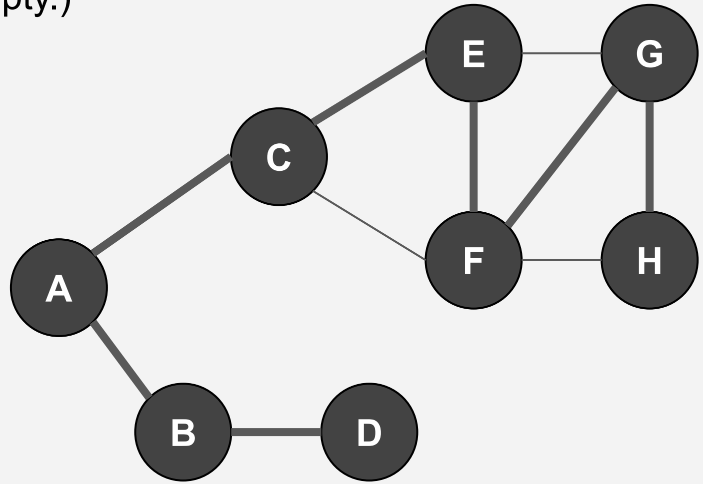
(pop A)

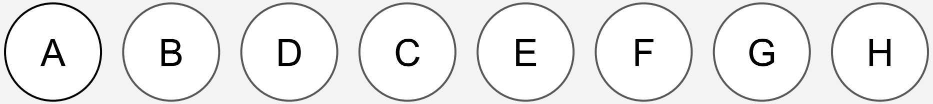




Depth-first search (DFS)

(done; The stack is now empty.)





In-Class Activity

Operator Overloading

(Note: We didn't have time for this; will do next week.)

Do Now Exercise

To prepare you for the lecture today, please do the following exercise.

Try searching (walking) routes
from Halligan Hall
to Museum of Fine Arts, Boston
using Google Maps.

Discussion:
How could we find the shortest path from Location A to Location B?

(watching a video)

(watching a video)

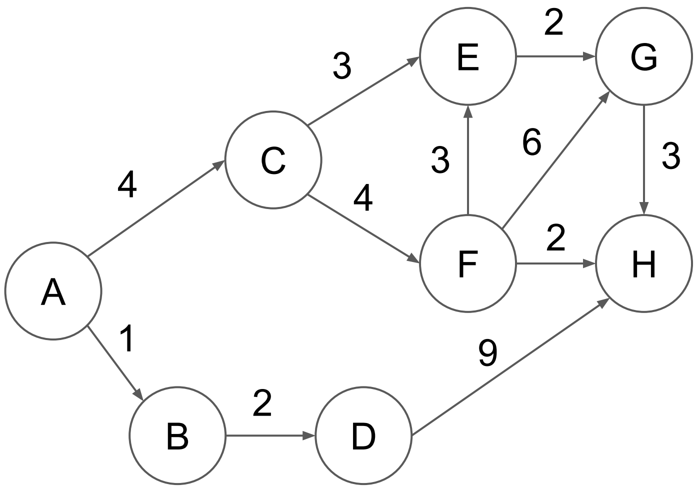
(Brute-force approach)

Single-source shortest-paths problem

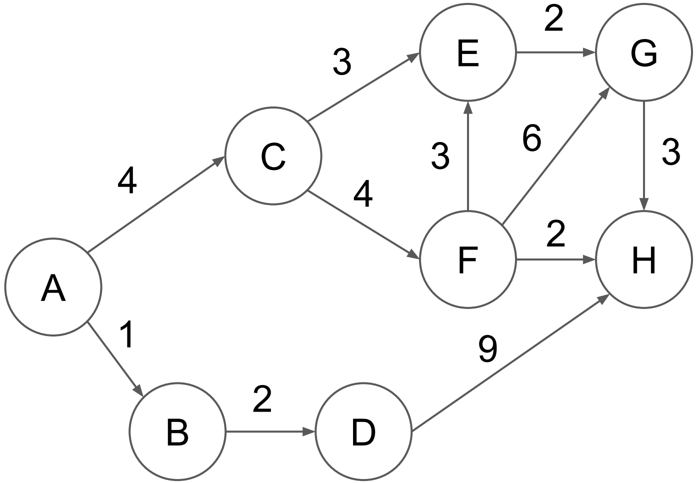
Dijkstra’s algorithm

(Greedy algorithm)

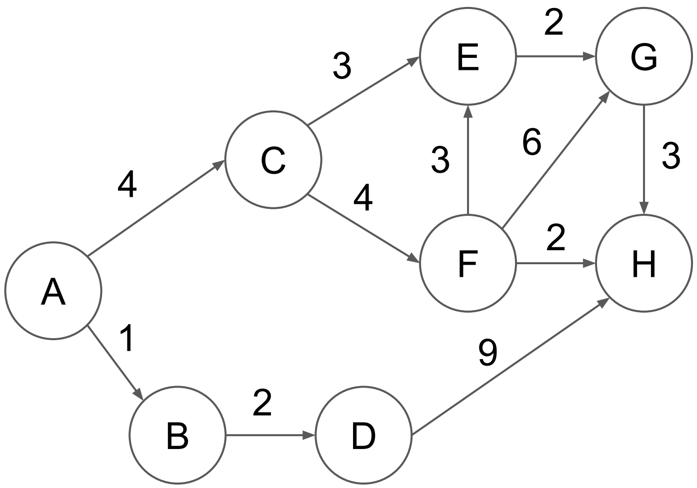
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

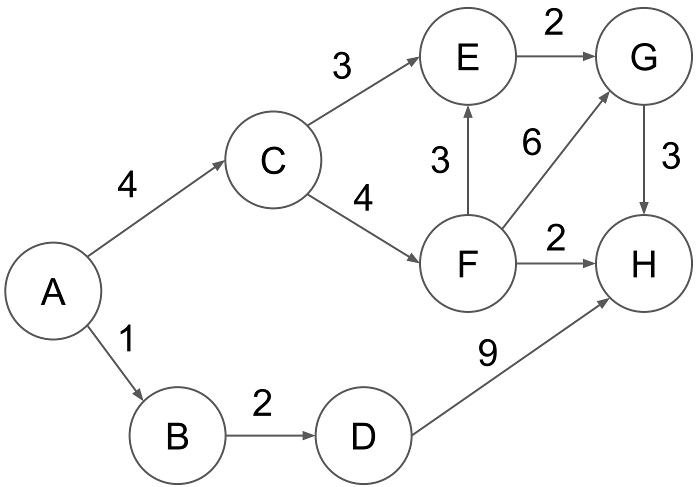


(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



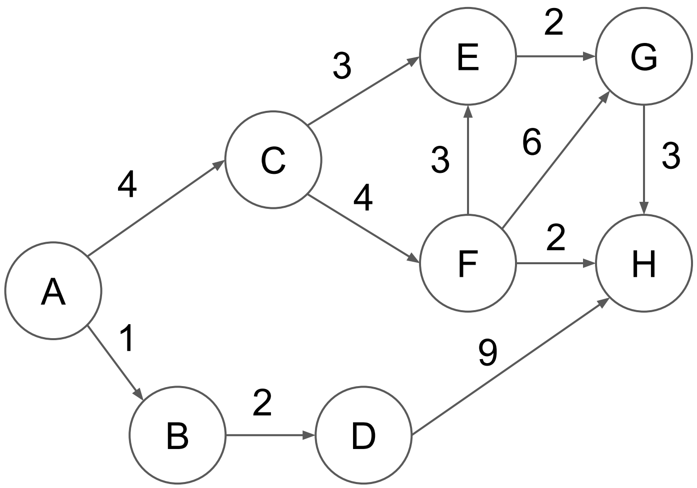
	distance	previous

(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



	distance	previous
A		
B		
C		
D		
E		
F		
G		
H		

(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



	distance	previous
(A)	~~INFINITY~~ 0	NONE
(B)	~~INFINITY~~ 1	~~NONE~~ A
(C)	~~INFINITY~~ 4	~~NONE~~ A
(D)	~~INFINITY~~ 3	~~NONE~~ B
(E)	~~INFINITY~~ 7	~~NONE~~ C
(F)	~~INFINITY~~ 8	~~NONE~~ C
(G)	~~INFINITY~~ 9	~~NONE~~ E
(H)	~~INFINITY~~ ~~12~~ 10	~~NONE~~ ~~D~~ F

In Your Pocket

arrays
linked lists
stacks
queues
trees
heaps
hash tables
graphs

man ssh exit pwd cd ls valgrind touch mkdir cp rm rmdir mv cat head tail less redirect (>, >>, <) pipe (|) (echo, sort, uniq, wc) diff grep clear clang++ valgrind make

Sorting Algorithms
- Selection sort
- Insertion sort
- Merge sort
- Quicksort
- Heapsort
- Counting sort

Some keywords from today's lecture:

- hash table, put, get, remove
- load factor, rehashing
- open addressing, linear probing, (quadratic probing, double hashing)
- graph, vertex, edge
- undirected graph, directed graph (digraph), connected/disconnected graph, (self loop edge)
- cycle
- adjacency matrix, adjacency list
- breadth-first search (BFS), depth-first search (DFS)
    - using 3 colors (White, Gray, Black)
- Single-source shortest-paths problem
- (brute-force approach)
- Dijkstra's algorithm, (Greedy algorithm)

To the lab!