

COL 351 : ANALYSIS & DESIGN OF ALGORITHMS

LECTURE 10

GRAPH ALGORITHMS III :

BFS , DFS , AND APPLICATIONS

AUG 14, 2026

|

ROHIT VAISH

GENERIC GRAPH SEARCH

input: an undirected or directed graph $G=(V,E)$, starting vertex s

goal: identify the vertices of V reachable from s efficiently

GENERIC GRAPH SEARCH

input: an undirected or directed graph $G=(V,E)$, starting vertex s

goal: identify the vertices of V reachable from s efficiently

$O(|V|+|E|)$ time

GENERIC GRAPH SEARCH

input: an undirected or directed graph $G=(V,E)$, starting vertex s

goal: identify the vertices of V reachable from s efficiently

Generic Algorithm

initially s is **explored**, other vertices **unexplored**

while some edge $(u,v) \in E$ with u **explored**
and v **unexplored**
└ mark v **explored**

GENERIC GRAPH SEARCH

input: an undirected or directed graph $G=(V,E)$, starting vertex s

goal: identify the vertices of V reachable from s efficiently

Generic Algorithm

initially s is **explored**, other vertices **unexplored**

while some edge $(u,v) \in E$ with u **explored**
and v **unexplored**
└ mark v **explored**

Claim: v is marked **explored** $\iff$ G has a path from s to v

Generic Algorithm

initially s is explored, other vertices unexplored

while some edge $(u, v) \in E$ with u explored
and v unexplored
| mark v explored

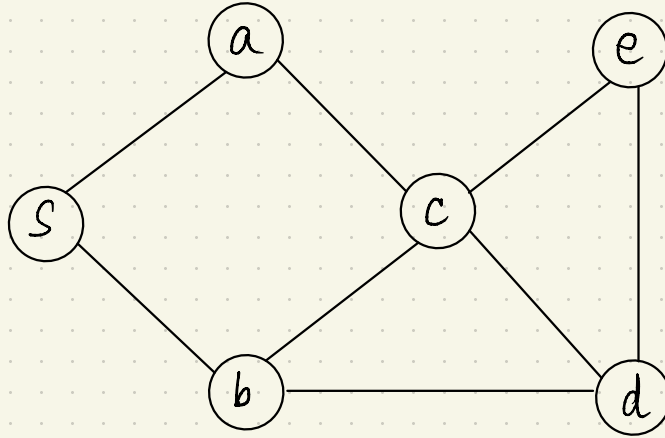
Breadth-first search (BFS)

explore layerwise

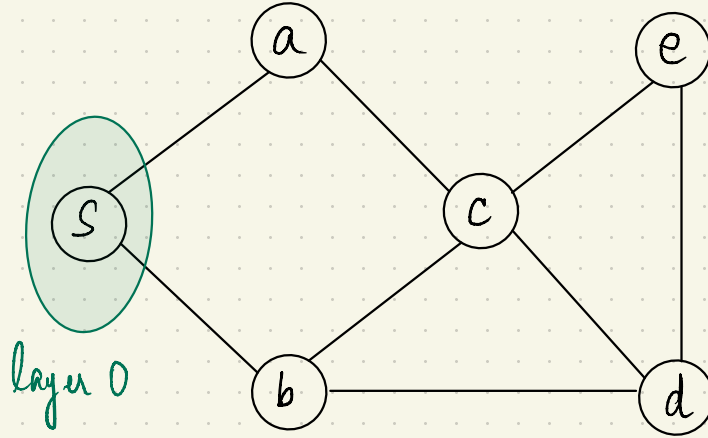
Depth-first search (DFS)

explore like a maze

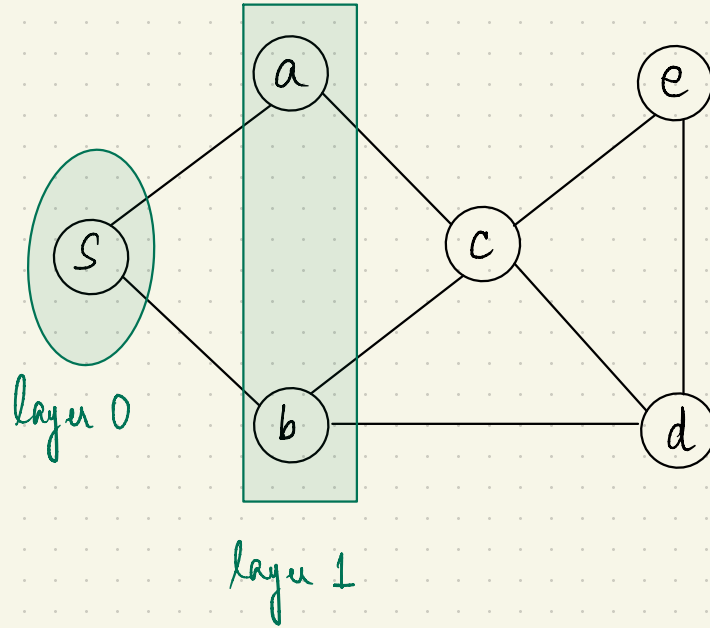
BREADTH - FIRST SEARCH



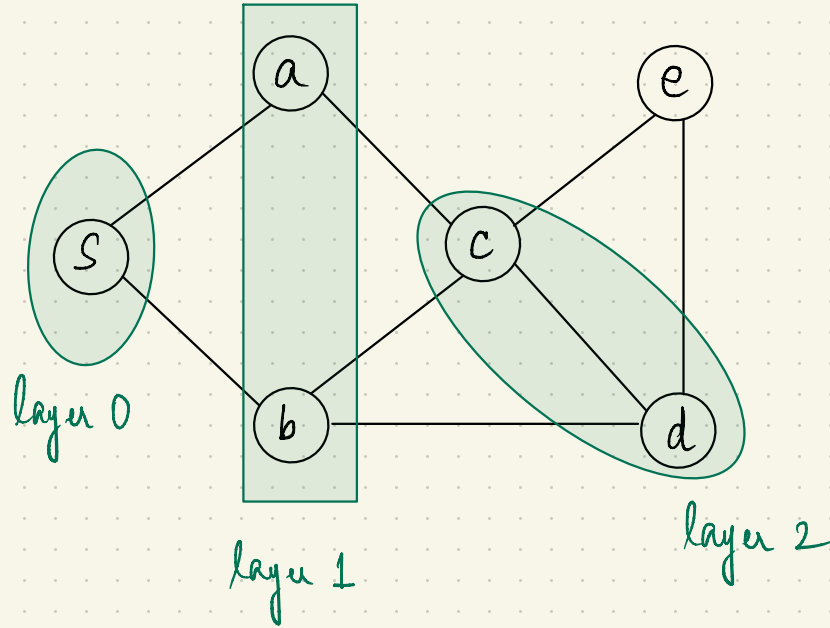
BREADTH - FIRST SEARCH



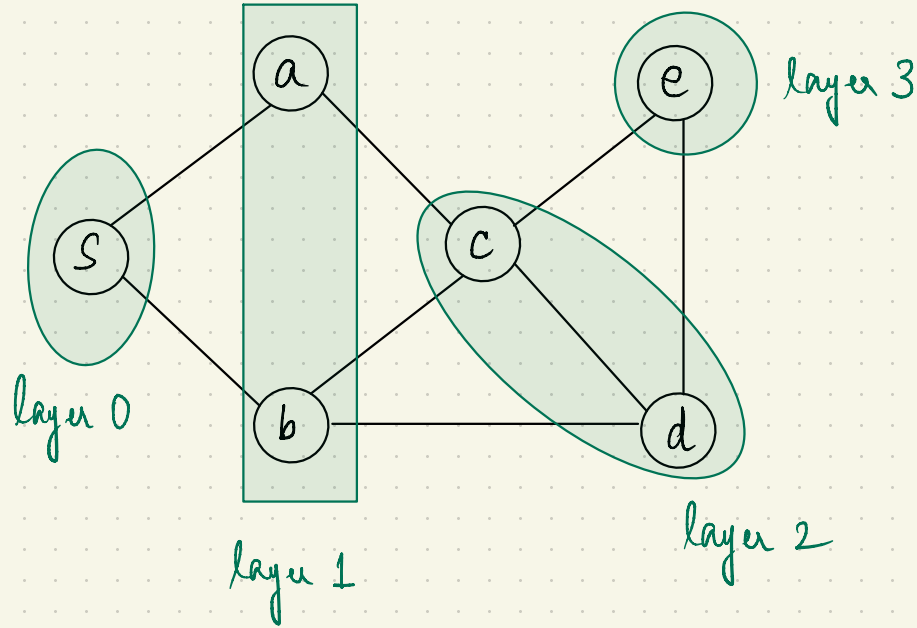
BREADTH - FIRST SEARCH



BREADTH - FIRST SEARCH



BREADTH - FIRST SEARCH



BREADTH - FIRST SEARCH

mark s as explored, all other vertices unexplored

$Q :=$ a queue data structure (FIFO), initialized with s

while $Q \neq \emptyset$

 remove the first node of Q , say v

 for each $(v, w) \in E$

 if w is unexplored

 mark w as explored

 add w to the end of Q

BREADTH - FIRST SEARCH

Claim 1: At the end of BFS

v explored $\iff$ there is a path from s to v

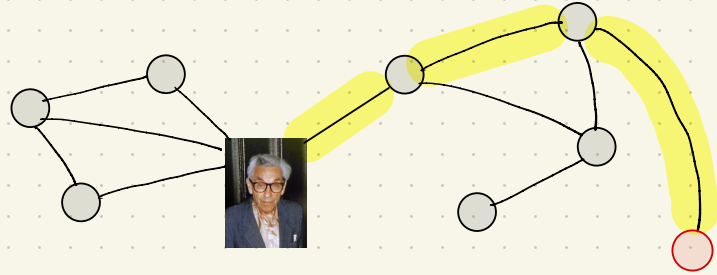
Claim 2: Running time of while-loop of BFS is $O(n_s + m_s)$

where $n_s = \#$ vertices reachable from s

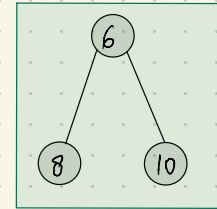
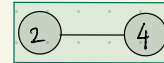
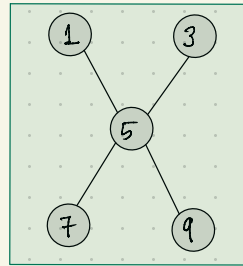
$m_s = \#$ edges $u \quad u \quad u$

APPLICATIONS OF BFS

Shortest paths

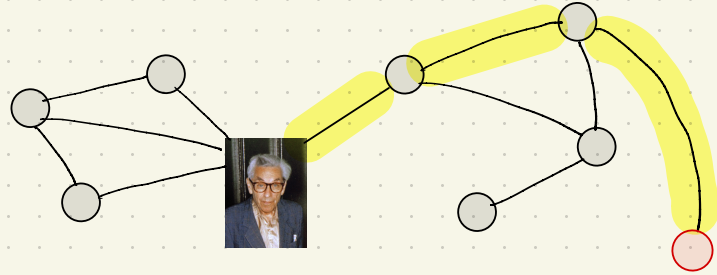


Connected Components

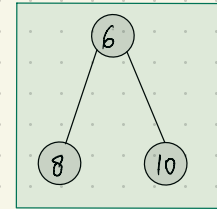
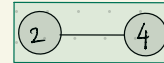
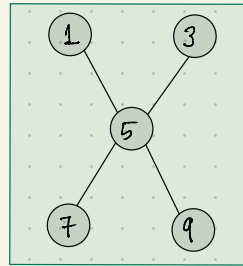


APPLICATIONS OF BFS

Shortest paths



Connected Components



SHORTEST PATHS

input: an undirected/directed graph $G=(V,E)$, a starting vertex s

output: $\text{dist}(s,v)$ for every vertex $v \in V$

SHORTEST PATHS

input: an undirected/directed graph $G=(V,E)$, a starting vertex s

output: $\text{dist}(s,v)$ for every vertex $v \in V$

Number of edges

$\text{dist}(s,v) =$ length of shortest path from s to v if one exists,
 ∞ otherwise

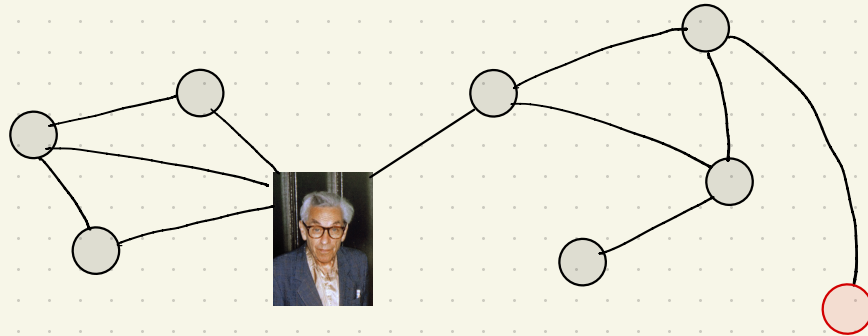
SHORTEST PATHS

input: an undirected/directed graph $G=(V,E)$, a starting vertex s

output: $\text{dist}(s,v)$ for every vertex $v \in V$

number of edges

$\text{dist}(s,v) =$ length of shortest path from s to v if one exists,
 ∞ otherwise



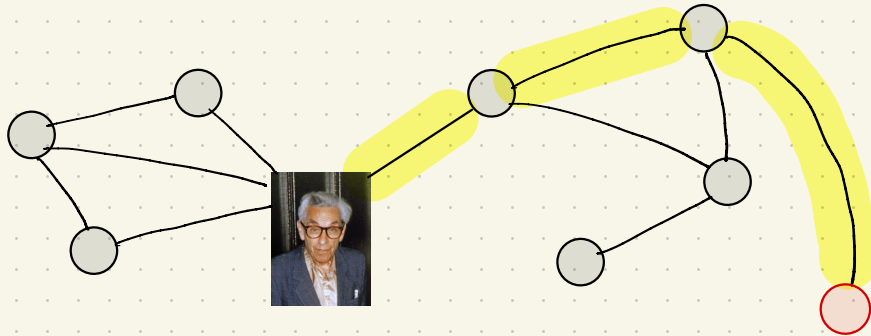
SHORTEST PATHS

input: an undirected/directed graph $G=(V,E)$, a starting vertex s

output: $\text{dist}(s,v)$ for every vertex $v \in V$

number of edges

$\text{dist}(s,v) =$ length of shortest path from s to v if one exists,
 ∞ otherwise



SHORTEST PATHS

mark s as **explored**, all other vertices **unexplored**

$\text{dist}(s) := 0$, $\text{dist}(v) = \infty$ for all $v \neq s$.

$Q :=$ a queue data structure (FIFO), initialized with s

while $Q \neq \emptyset$

 remove the first node of Q , say v

 for each $(v, w) \in E$

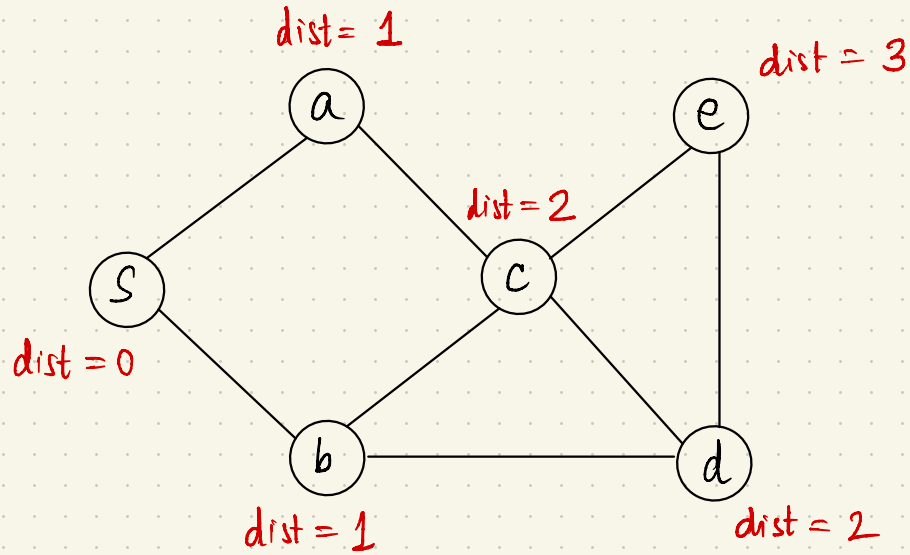
 if w is **unexplored**

 mark w as **explored**

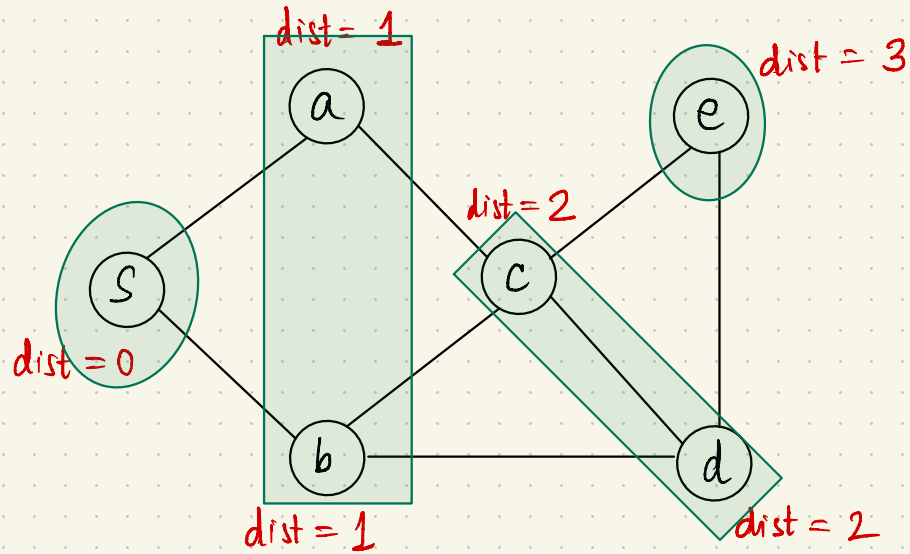
$\text{dist}(w) := \text{dist}(v) + 1$

 add w to the end of Q

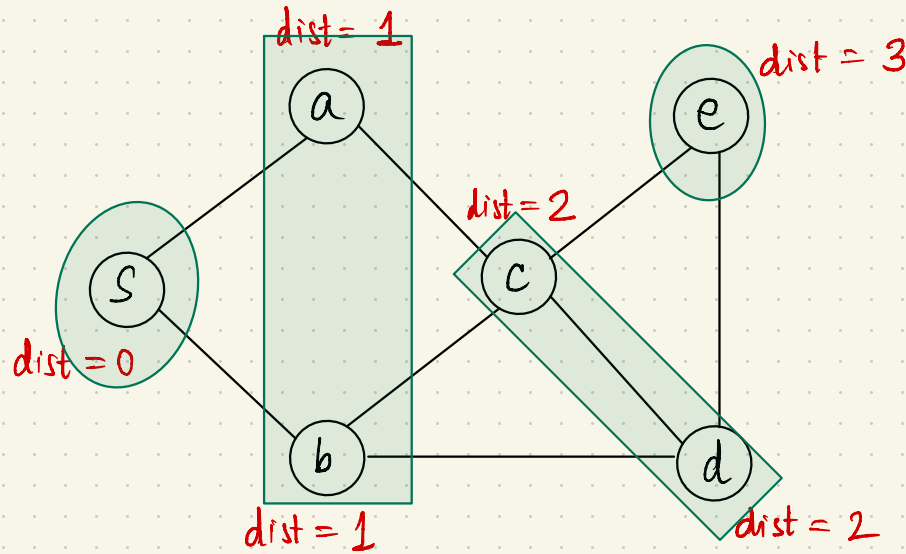
SHORTEST PATHS



SHORTEST PATHS

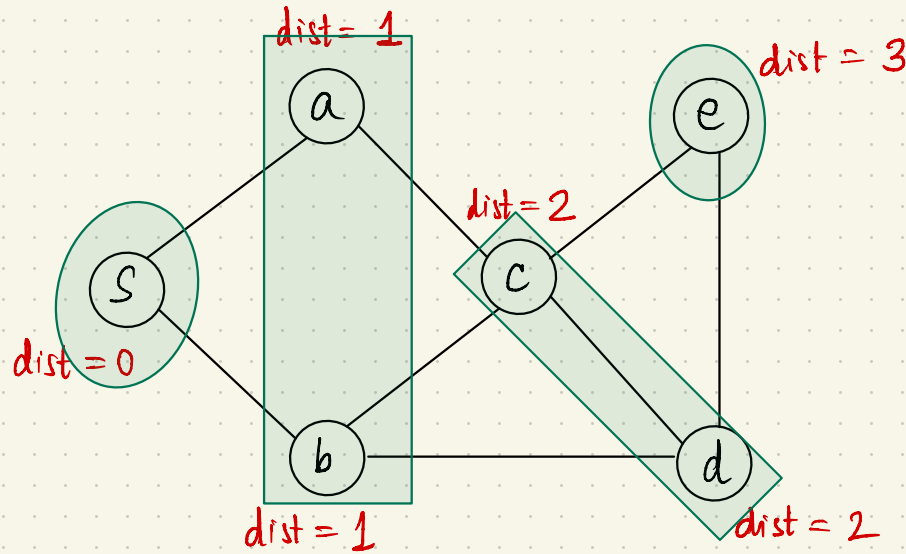


SHORTEST PATHS



Claim: at termination, $\text{dist}(v) = i \iff v$ is in i^{th} layer

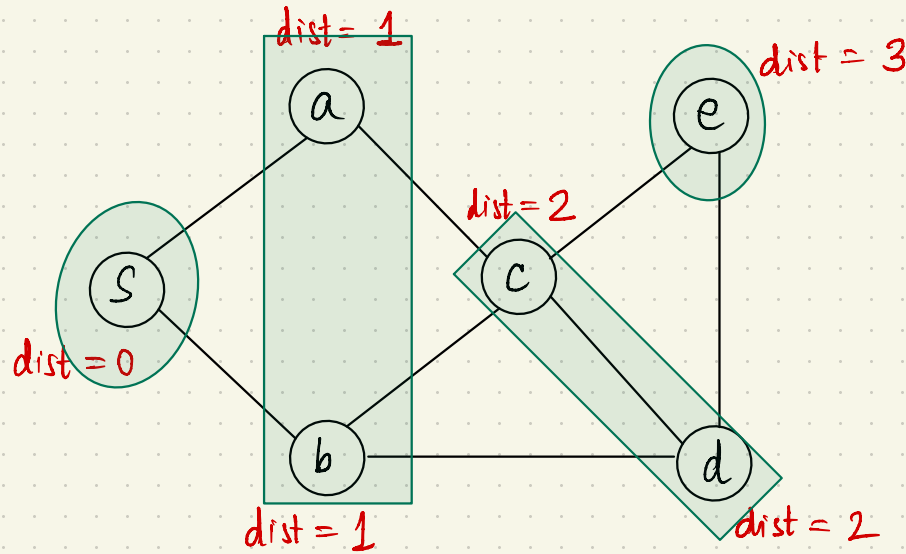
SHORTEST PATHS



Claim: at termination, $\text{dist}(v) = i \iff v$ is in i^{th} layer

shortest path $s \rightarrow v$ has i edges

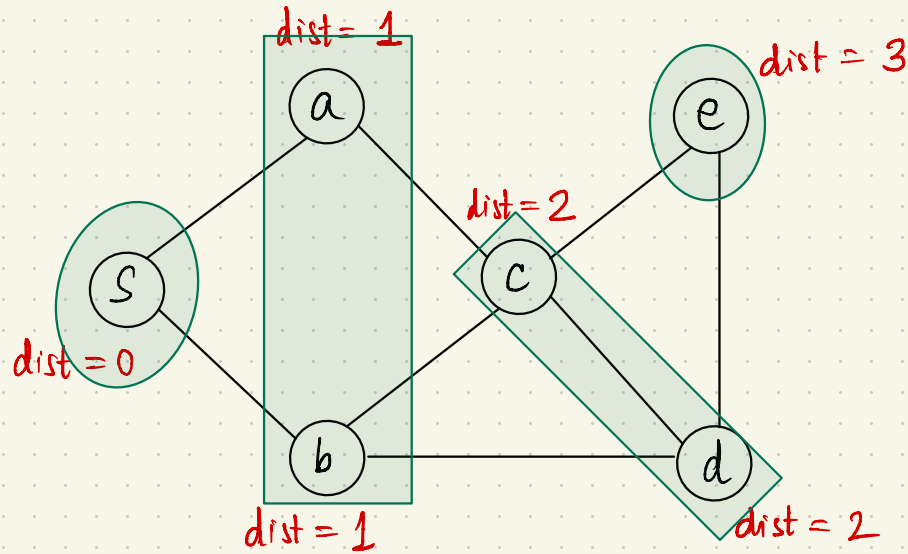
SHORTEST PATHS



Claim: at termination, $\text{dist}(v) = i \iff v$ is in i^{th} layer

Proof: Exercise (by induction)

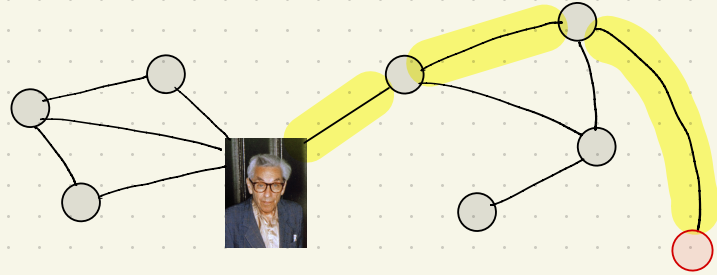
SHORTEST PATHS



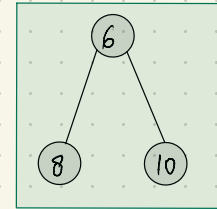
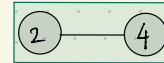
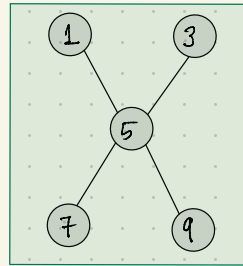
Claim: The augmented BFS has running time $O(m+n)$.

APPLICATIONS OF BFS

Shortest paths



Connected Components

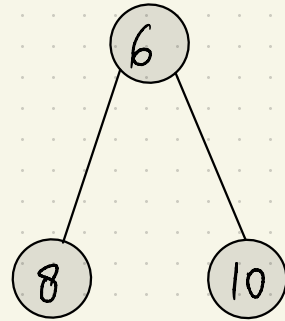
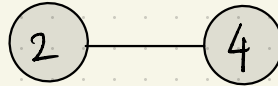
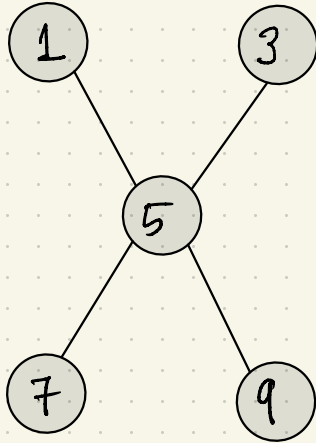


CONNECTED COMPONENTS

Undirected graph $G = (V, E)$

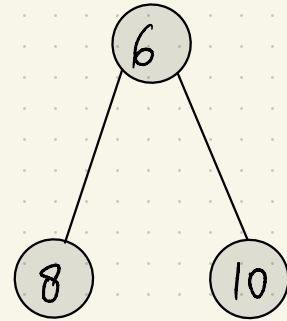
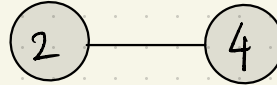
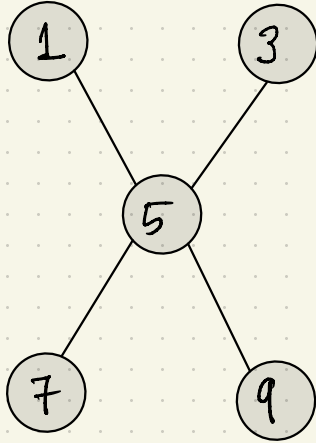
CONNECTED COMPONENTS

Undirected graph $G = (V, E)$



CONNECTED COMPONENTS

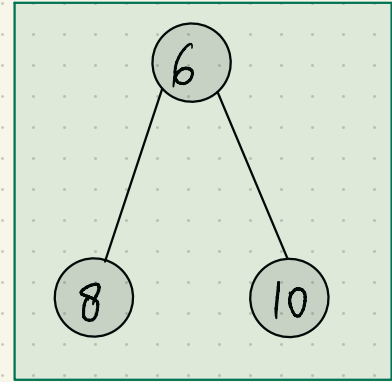
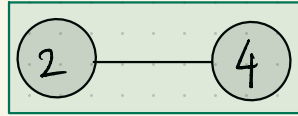
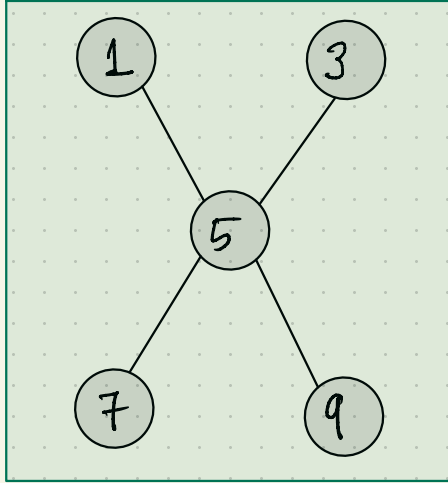
Undirected graph $G = (V, E)$



Connected components: maximal subsets of reachable vertices
(or equivalence class)

CONNECTED COMPONENTS

Undirected graph $G = (V, E)$



Connected components: maximal subsets of reachable vertices
(or equivalence class)

CONNECTED COMPONENTS

input: an undirected graph $G = (V, E)$

output: identify the connected components of G

CONNECTED COMPONENTS

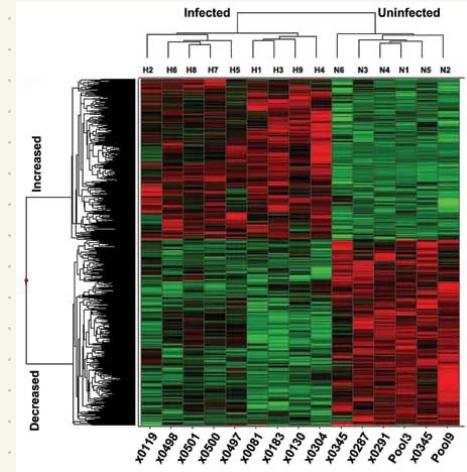
input: an undirected graph $G = (V, E)$

output: identify the connected components of G



Destination	Airline	Flight	Status
ATLANTA	DELTA	DL 2634	Cancelled
ATLANTA	DELTA	DL 2633	Cancelled
ATLANTA	DELTA	DL 2052	C1 Delayed
ATLANTA	DELTA	DL 1488	D5 Delayed
AUSTIN	DELTA	DL 4153	D3 Delayed
BOSTON	DELTA	DL 1054	D3 Delayed
BOSTON	DELTA	DL 1022	D5 Delayed
BOSTON	DELTA	DL 1033	D10 Delayed
BOSTON	jetBlue	B6 2184	C3 On Time

Detecting network failures



Genome clustering

CONNECTED COMPONENTS via BFS

CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

for $i = 1$ to n // try all vertices

CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

for $i = 1$ to n // try all vertices

 if i is **unexplored**

 increment CC count by 1

CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue

CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

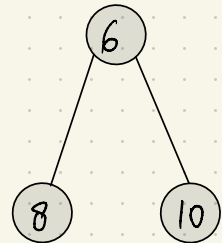
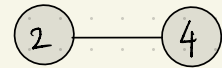
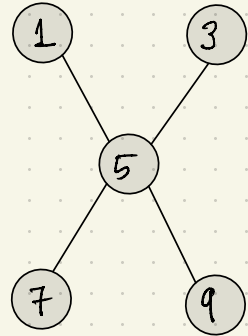
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as *unexplored*

initialize CC count $:= 0$

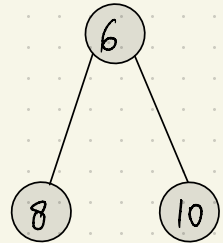
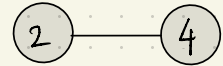
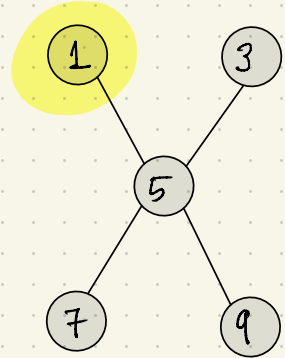
for $i = 1$ to n // try all vertices

if i is *unexplored*

increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as *unexplored*

initialize CC count $:= 0$

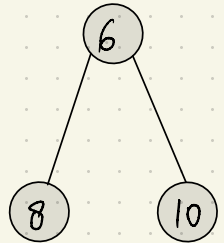
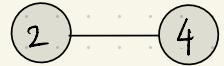
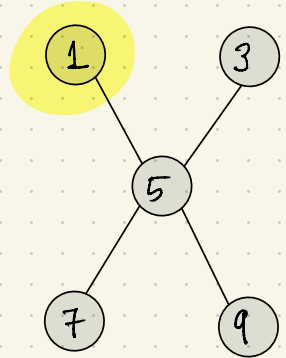
for $i = 1$ to n // try all vertices

if i is *unexplored*

increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

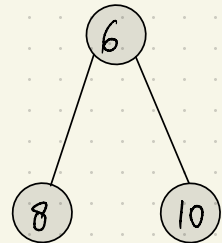
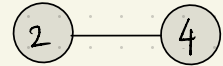
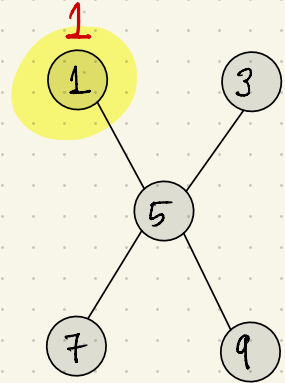
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

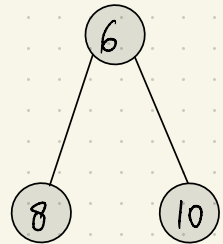
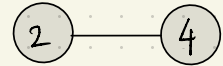
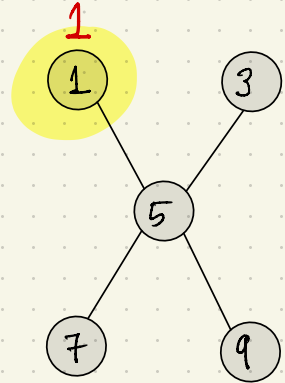
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS (G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

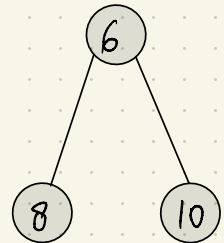
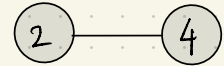
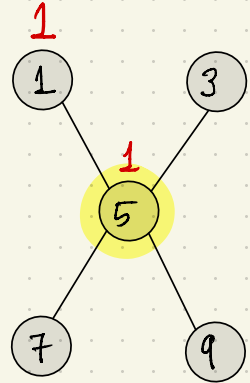
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS (G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

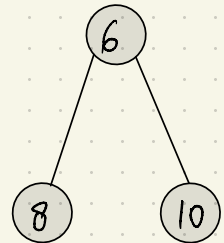
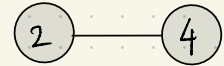
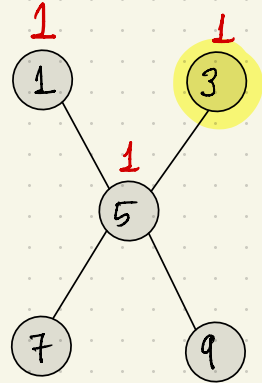
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS (G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

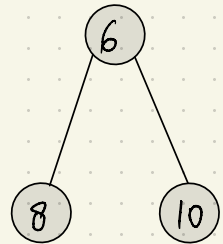
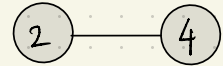
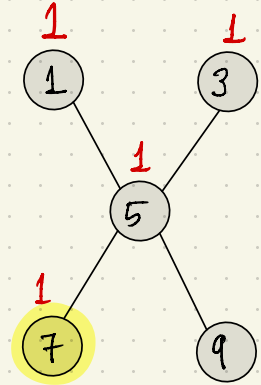
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS (G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

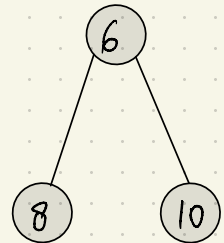
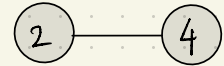
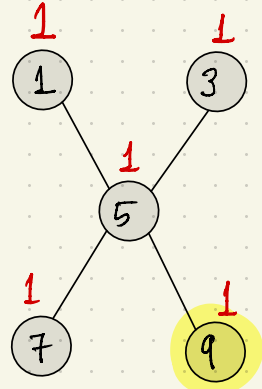
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS (G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

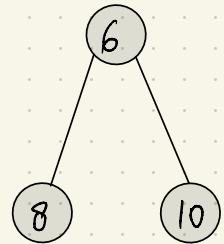
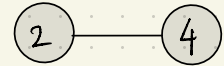
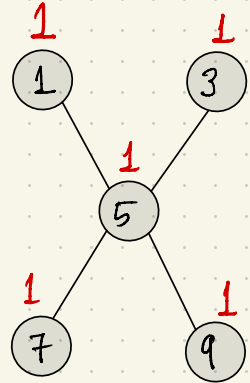
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS (G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as *unexplored*

initialize CC count $:= 0$

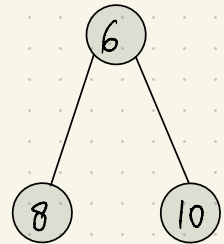
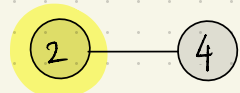
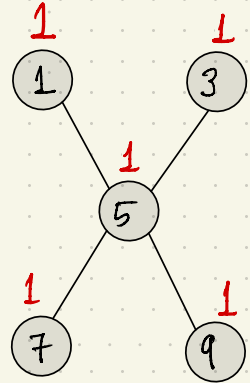
for $i = 1$ to n // try all vertices

if i is *unexplored*

increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

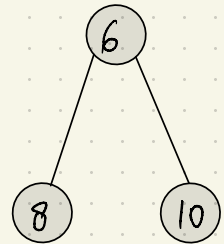
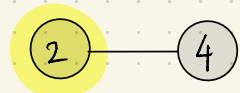
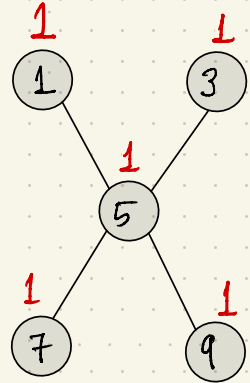
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

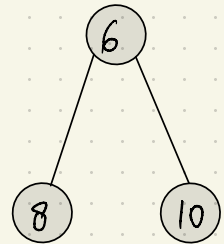
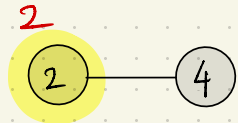
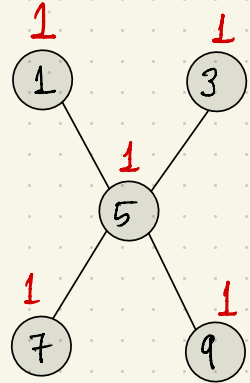
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

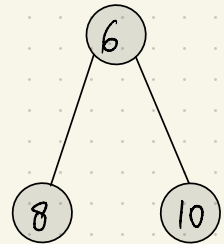
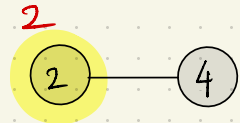
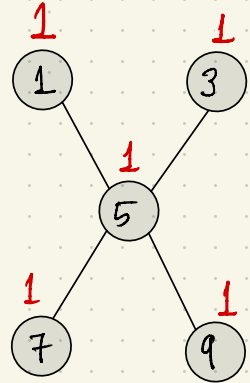
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS (G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

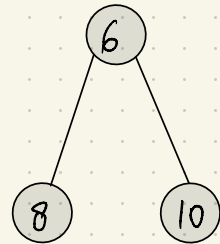
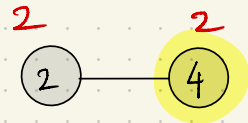
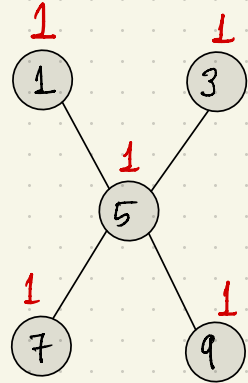
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS (G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

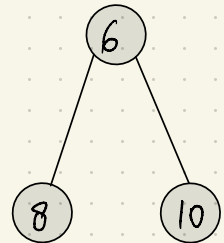
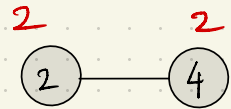
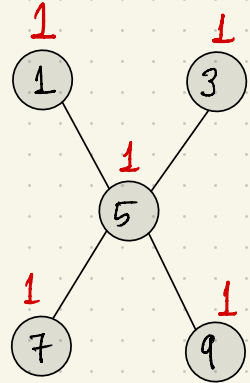
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS (G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as *unexplored*

initialize CC count $:= 0$

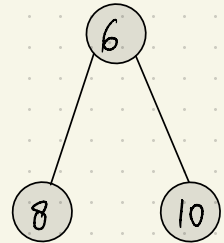
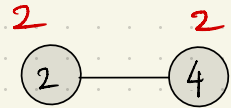
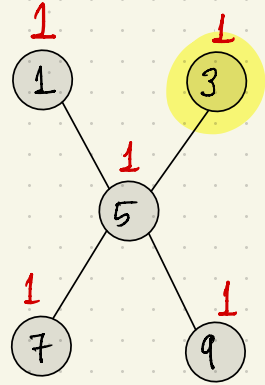
for $i = 1$ to n // try all vertices

if i is *unexplored*

increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

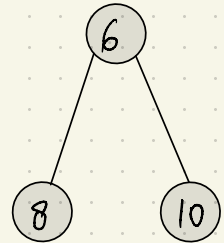
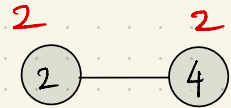
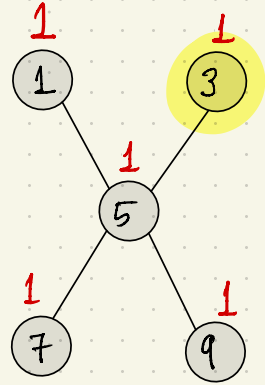
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as *unexplored*

initialize CC count $:= 0$

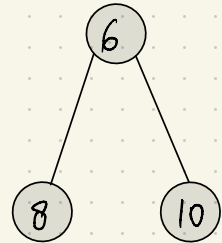
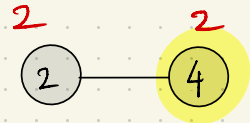
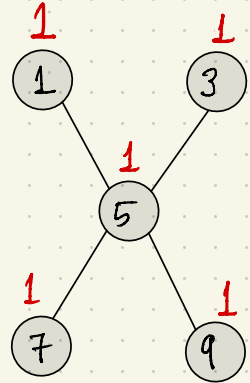
for $i = 1$ to n // try all vertices

if i is *unexplored*

increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

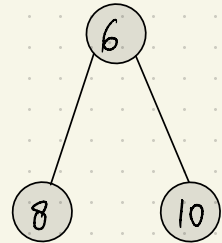
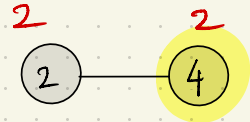
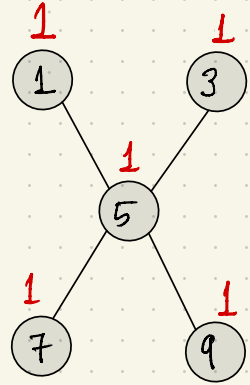
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as *unexplored*

initialize CC count $:= 0$

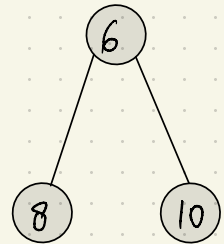
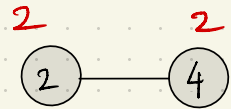
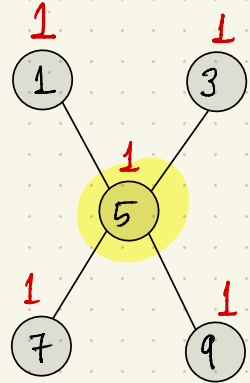
for $i = 1$ to n // try all vertices

if i is *unexplored*

increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as *unexplored*

initialize CC count $:= 0$

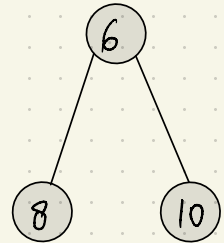
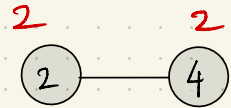
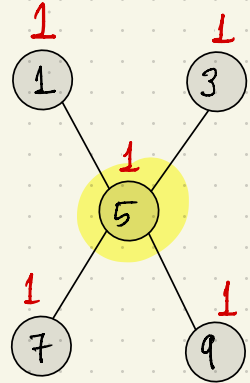
for $i = 1$ to n // try all vertices

if i is *unexplored*

increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as *unexplored*

initialize CC count $:= 0$

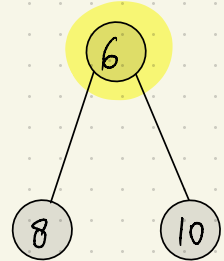
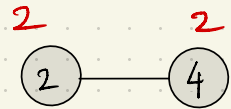
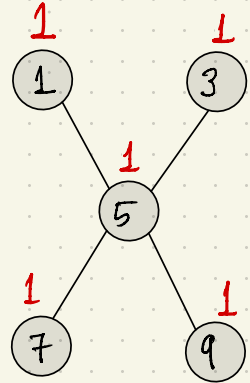
for $i = 1$ to n // try all vertices

if i is *unexplored*

increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

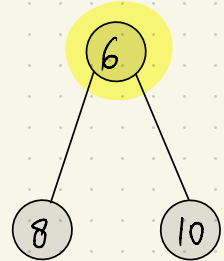
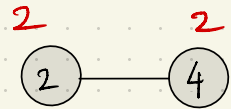
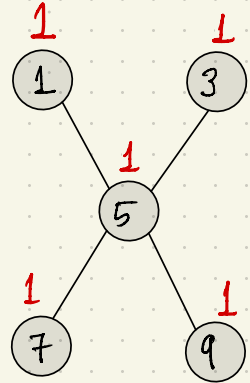
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

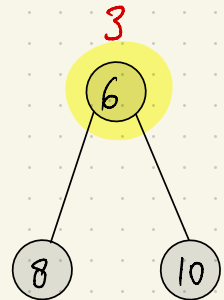
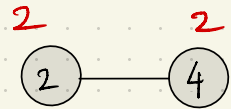
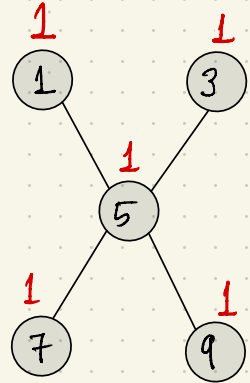
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

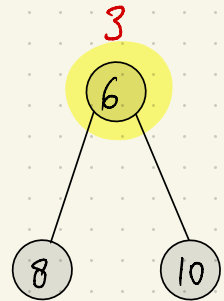
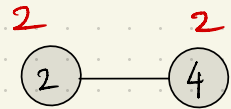
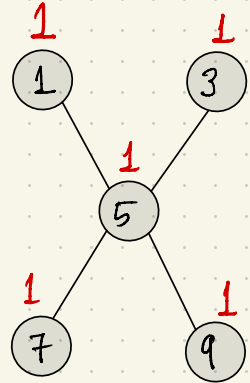
for $i = 1$ to n // try all vertices

if i is **unexplored**

increment CC count by 1

BFS (G, i)

(while updating CC number of
a vertex extracted from queue



CONNECTED COMPONENTS via BFS

mark all vertices $1, 2, \dots, n$ as **unexplored**

initialize CC count $:= 0$

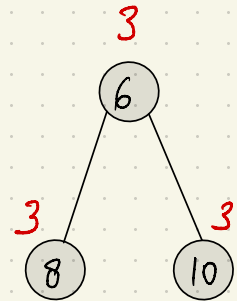
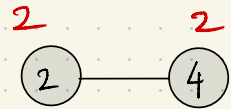
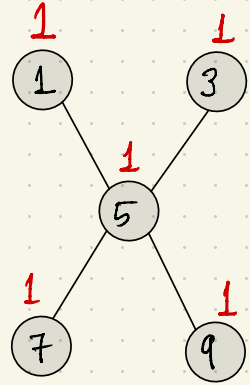
for $i = 1$ to n // try all vertices

if i is **unexplored**

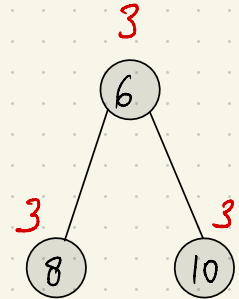
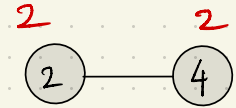
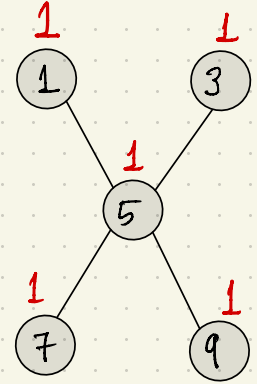
increment CC count by 1

BFS(G, i)

(while updating CC number of
a vertex extracted from queue



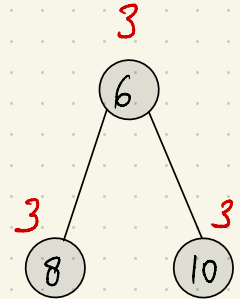
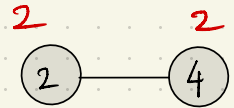
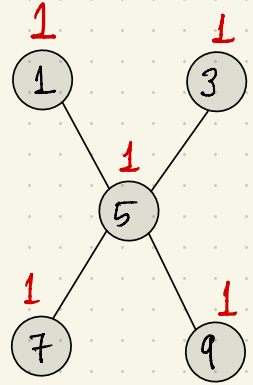
CONNECTED COMPONENTS via BFS



CONNECTED COMPONENTS via BFS

Claim: (a) At termination,

$cc[u] = cc[v] \iff u$ and v in same
connected component

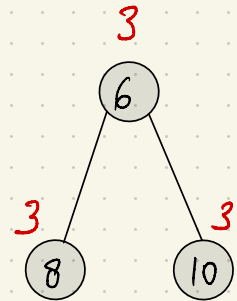
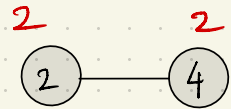
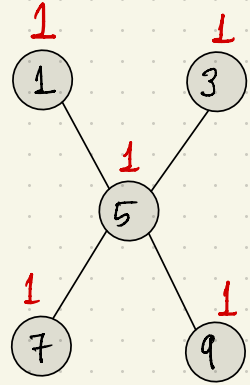


CONNECTED COMPONENTS via BFS

Claim: (a) At termination,

$cc[u] = cc[v] \iff u$ and v in same
Connected component

Idea: correctness of BFS +
explored vertex is never marked **unexplored**



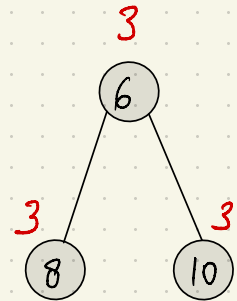
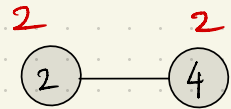
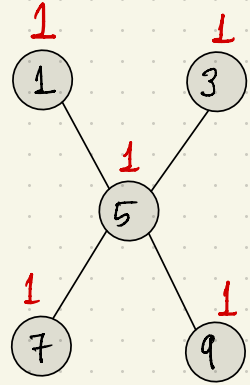
CONNECTED COMPONENTS via BFS

Claim: (a) At termination,

$cc[u] = cc[v] \iff u \text{ and } v \text{ in same}$
Connected component

Idea: correctness of BFS +
explored vertex is never marked **unexplored**

(b) Algo runs in $O(m+n)$ time



CONNECTED COMPONENTS via BFS

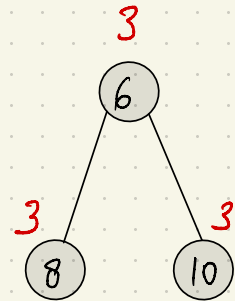
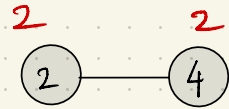
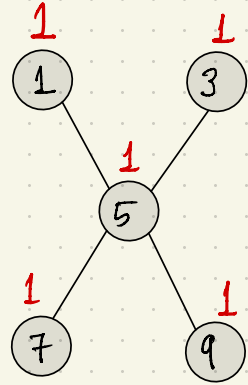
Claim: (a) At termination,

$cc[u] = cc[v] \iff u \text{ and } v \text{ in same}$
Connected component

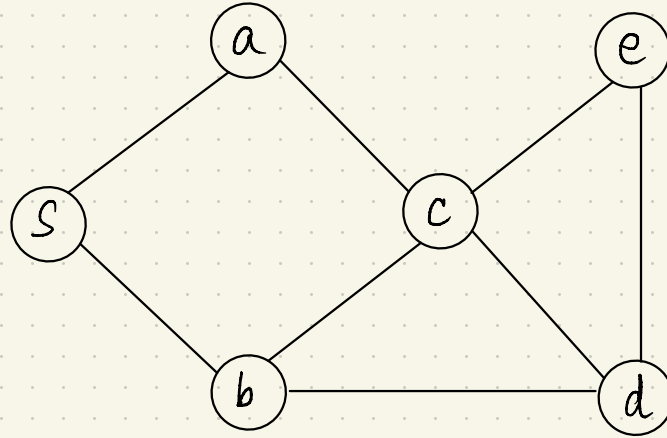
Idea: correctness of BFS +
explored vertex is never marked **unexplored**

(b) Algo runs in $O(m+n)$ time

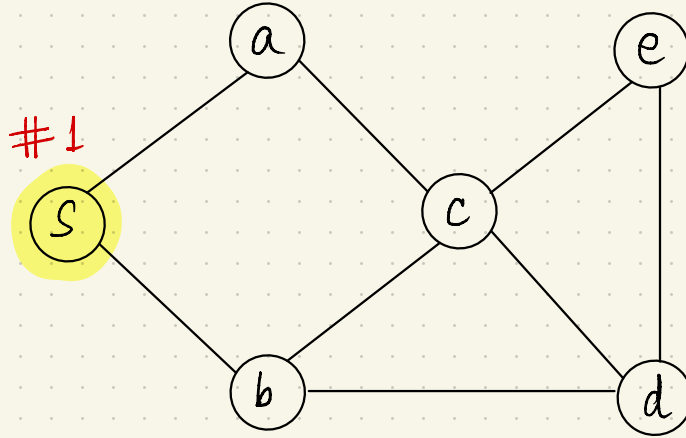
Idea: Time per component = $O(m_s + n_s)$.



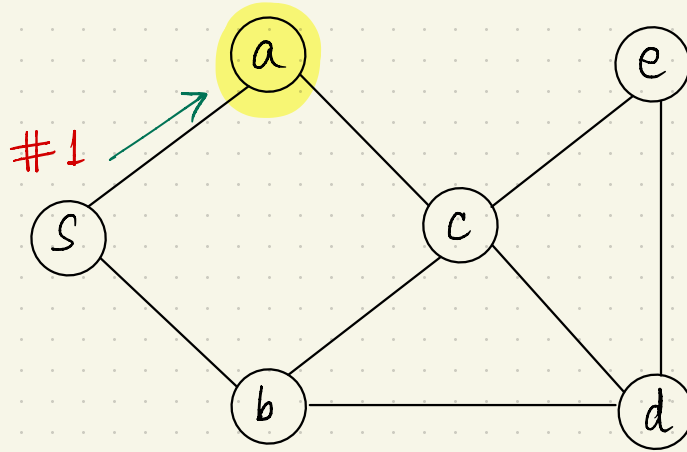
DEPTH-FIRST SEARCH



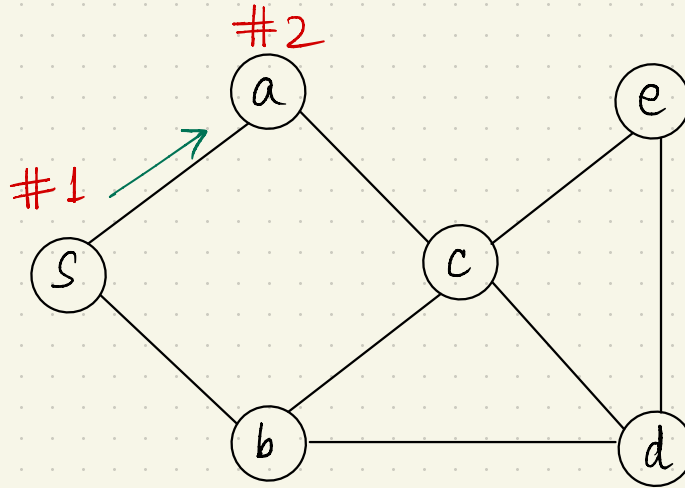
DEPTH-FIRST SEARCH



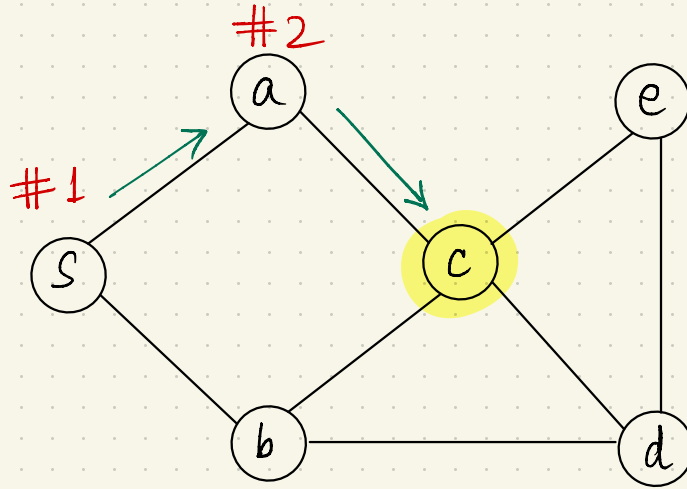
DEPTH-FIRST SEARCH



DEPTH-FIRST SEARCH

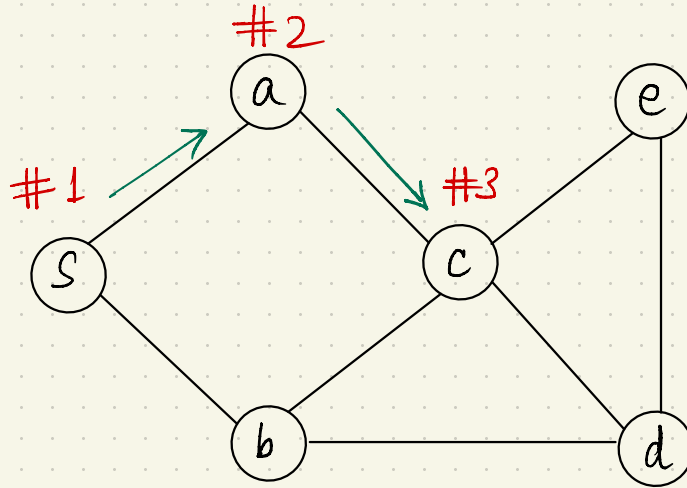


DEPTH-FIRST SEARCH



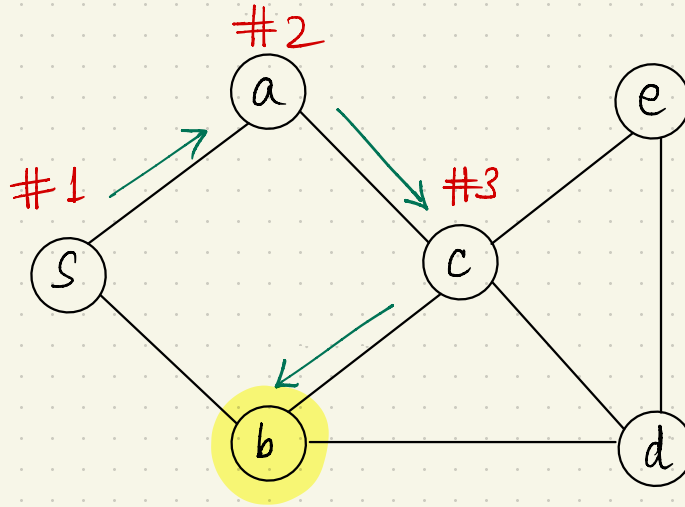
Differs from BFS !

DEPTH-FIRST SEARCH

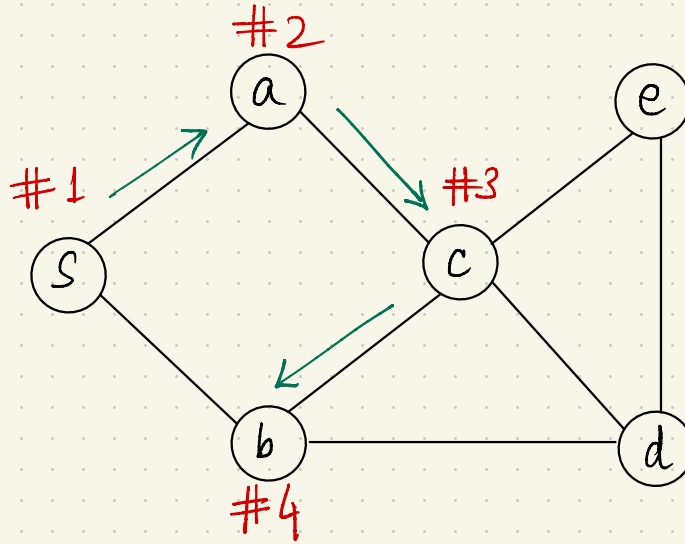


Differs from BFS !

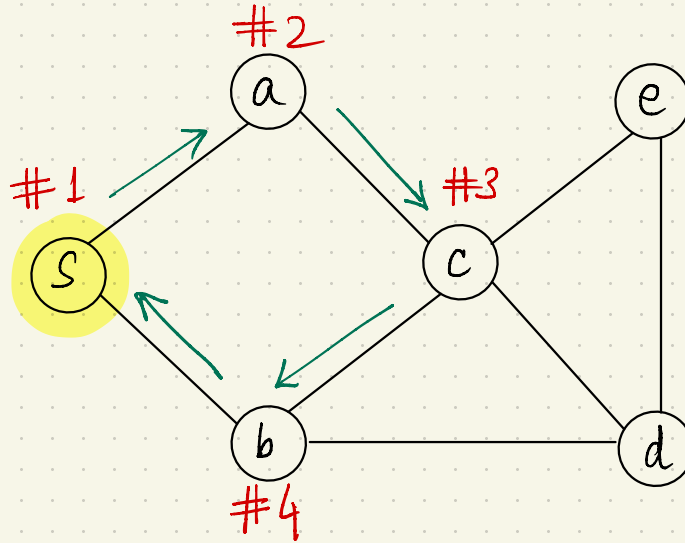
DEPTH-FIRST SEARCH



DEPTH-FIRST SEARCH

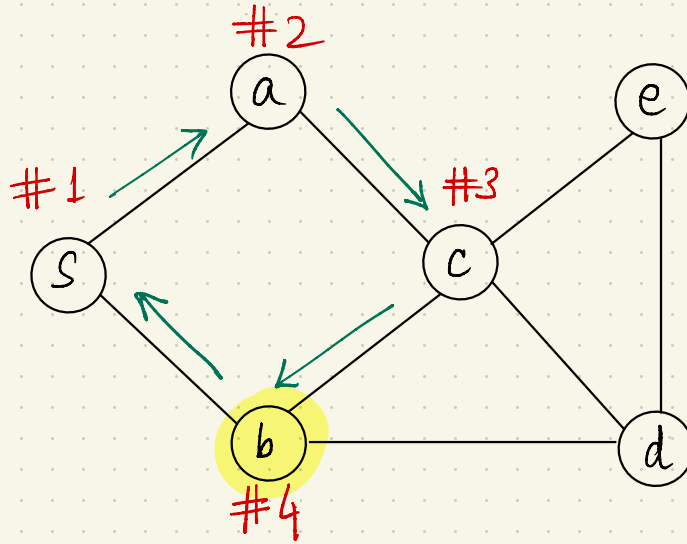


DEPTH-FIRST SEARCH



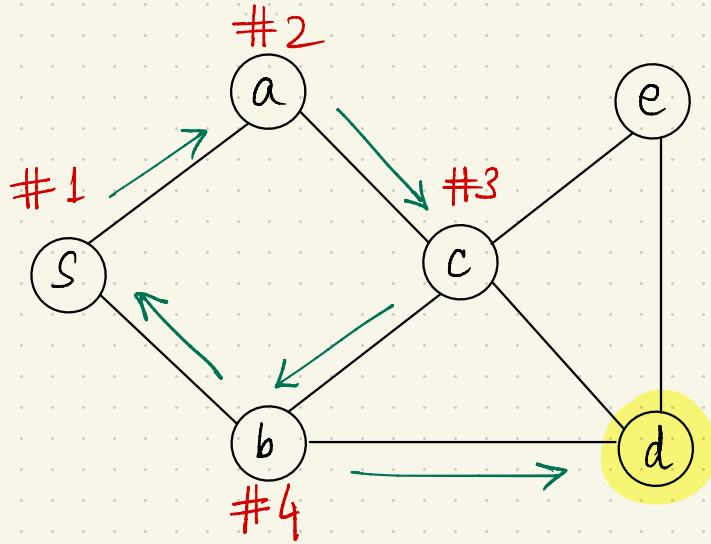
s is already explored $\Rightarrow$ backtrack!

DEPTH-FIRST SEARCH

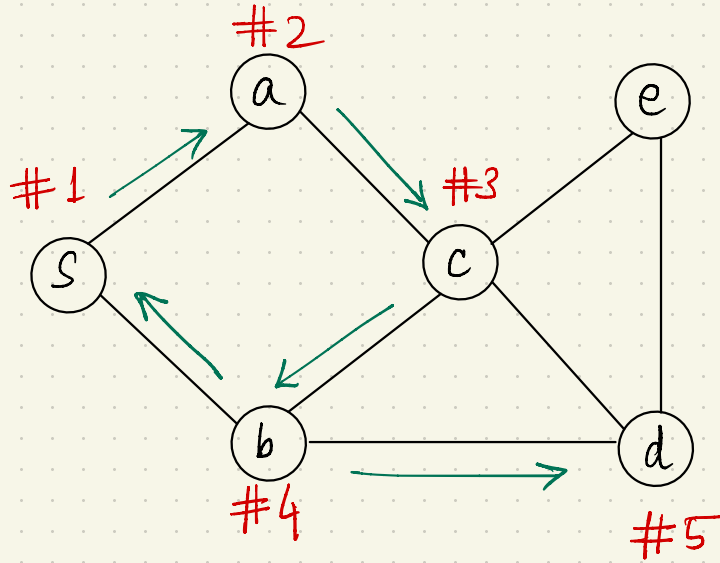


s is already explored $\Rightarrow$ backtrack!

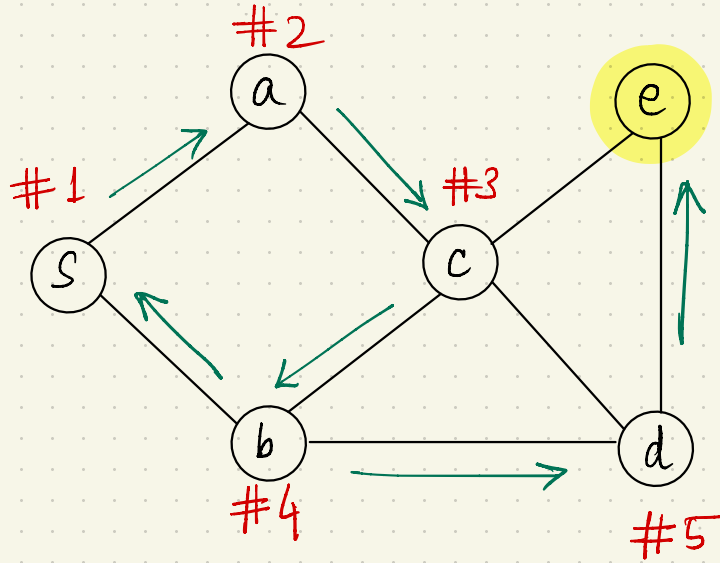
DEPTH-FIRST SEARCH



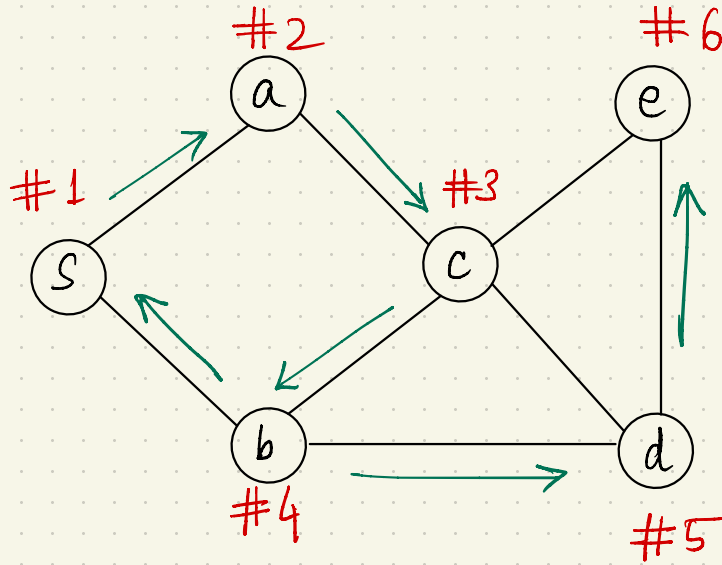
DEPTH-FIRST SEARCH



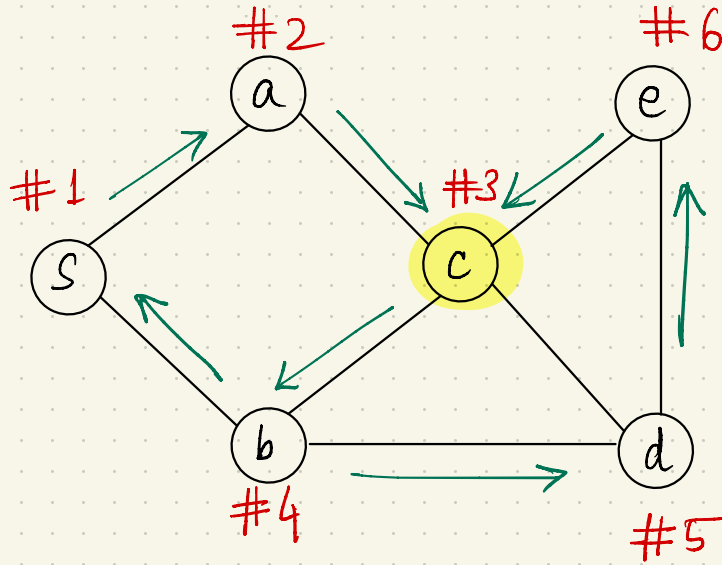
DEPTH-FIRST SEARCH



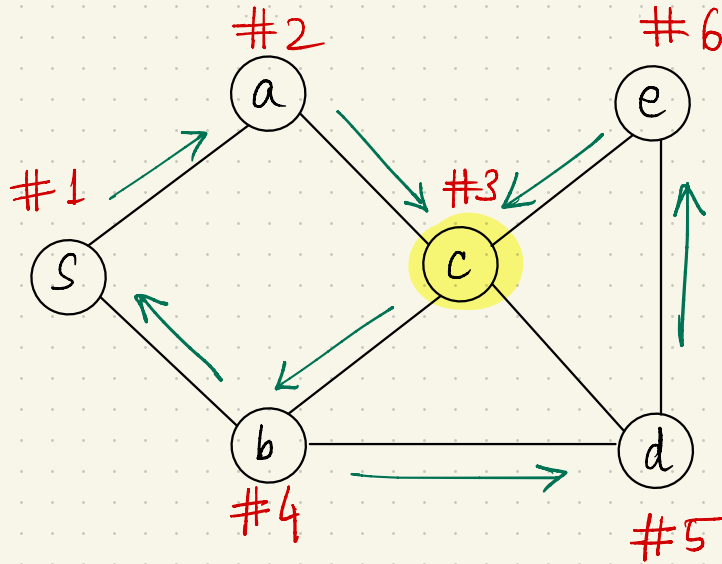
DEPTH-FIRST SEARCH



DEPTH-FIRST SEARCH

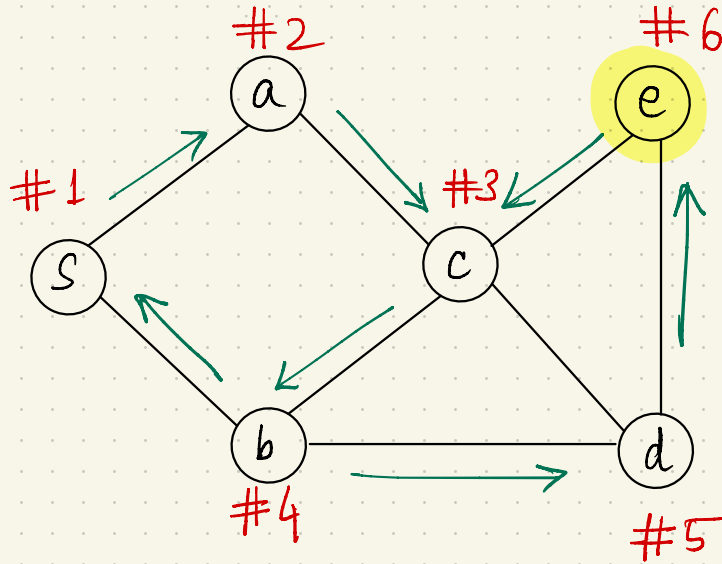


DEPTH-FIRST SEARCH



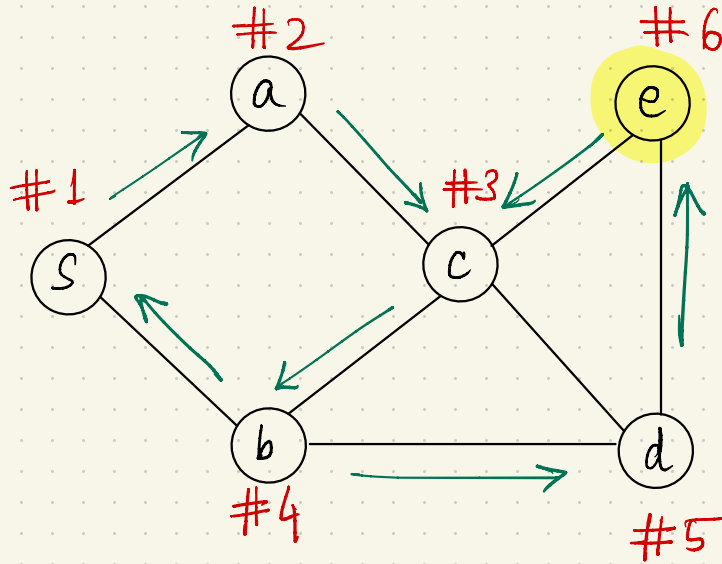
c is already explored $\Rightarrow$ backtrack!

DEPTH-FIRST SEARCH



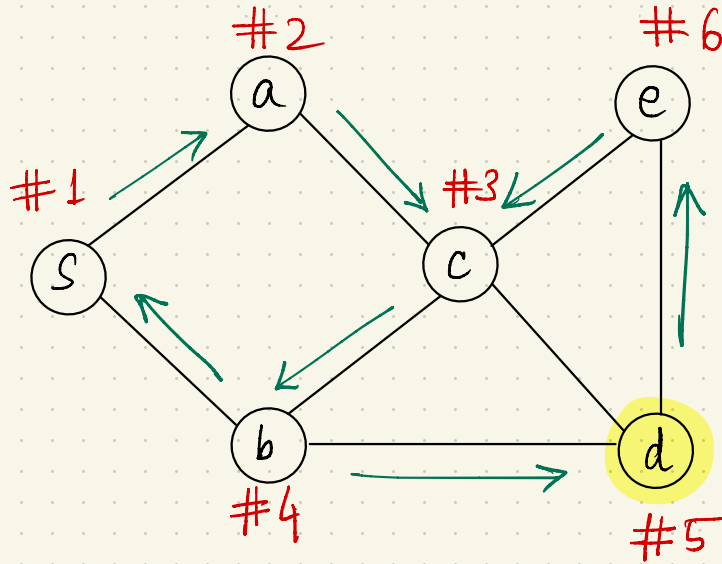
c is already explored $\Rightarrow$ backtrack!

DEPTH-FIRST SEARCH



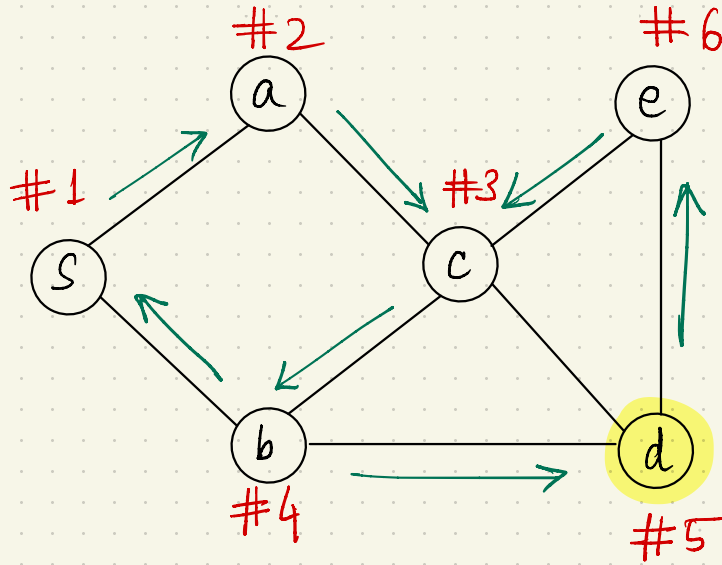
No unexplored neighbors of e $\Rightarrow$ backtrack!

DEPTH-FIRST SEARCH

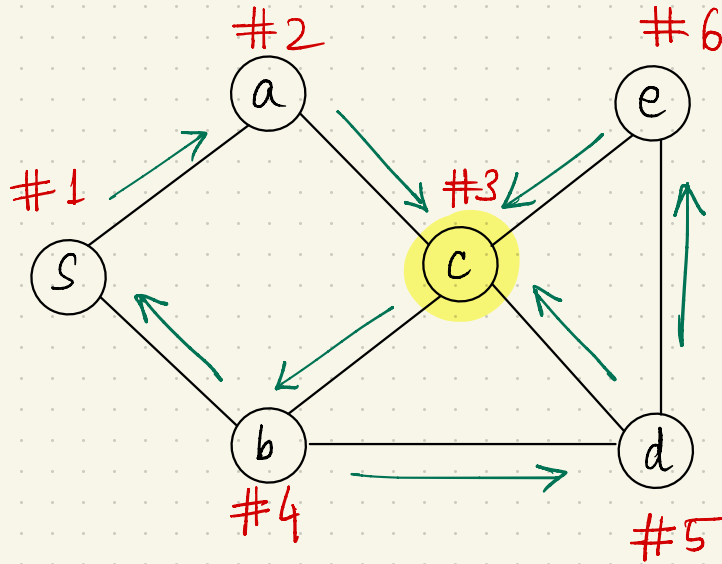


No unexplored neighbors of e $\Rightarrow$ backtrack!

DEPTH-FIRST SEARCH

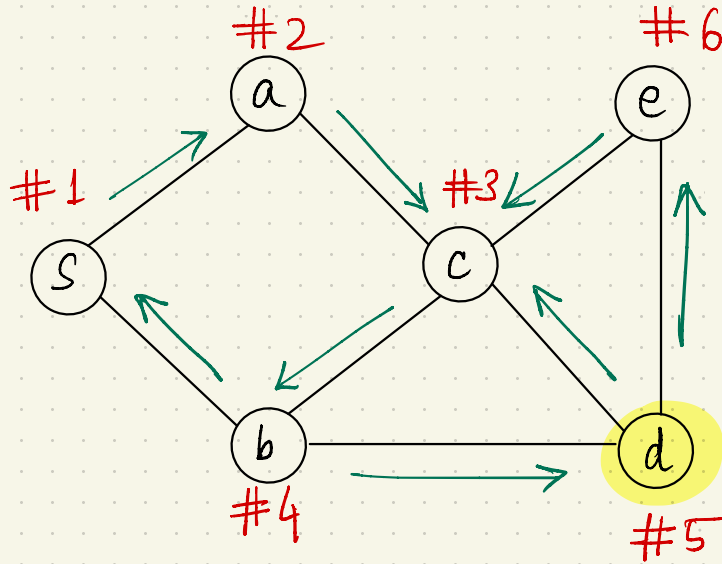


DEPTH-FIRST SEARCH



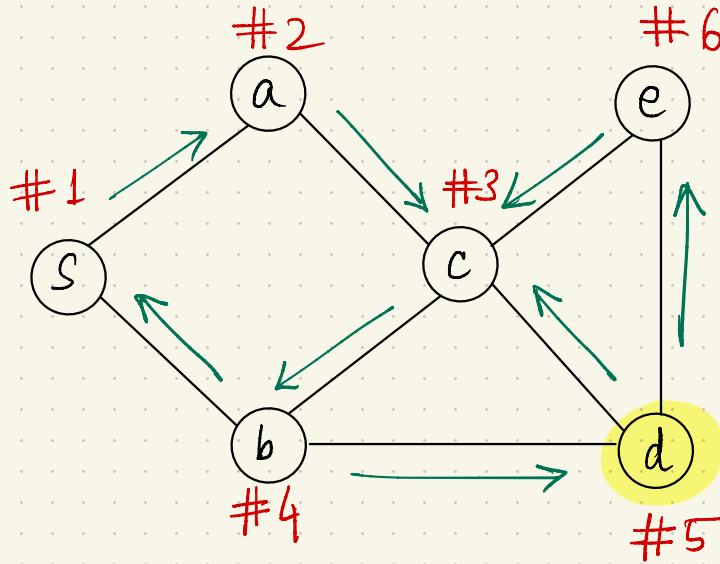
c is already explored $\Rightarrow$ backtrack!

DEPTH-FIRST SEARCH



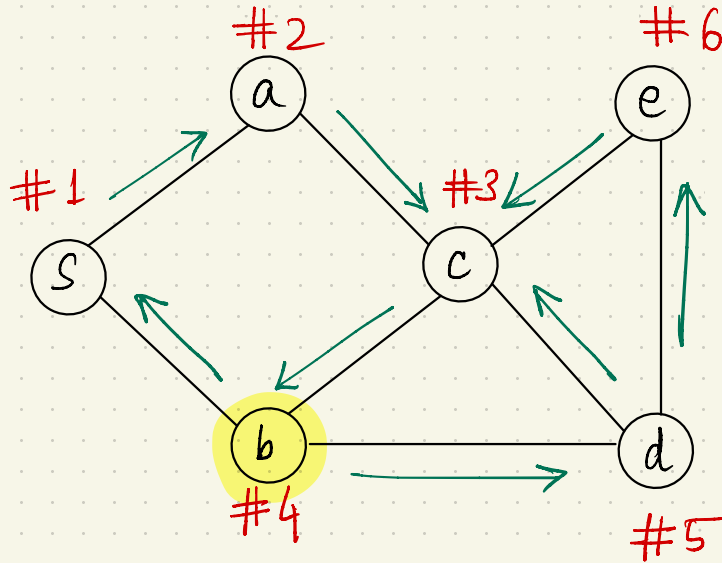
c is already explored $\Rightarrow$ backtrack!

DEPTH-FIRST SEARCH



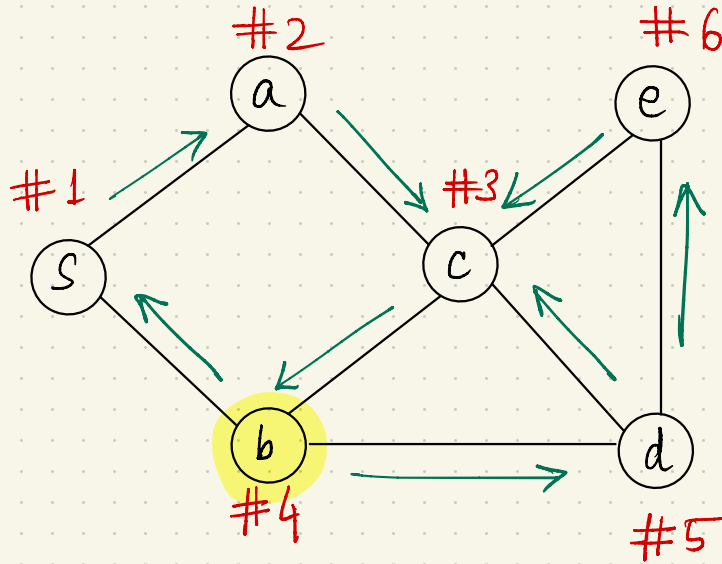
No unexplored neighbors of d $\Rightarrow$ backtrack!

DEPTH-FIRST SEARCH



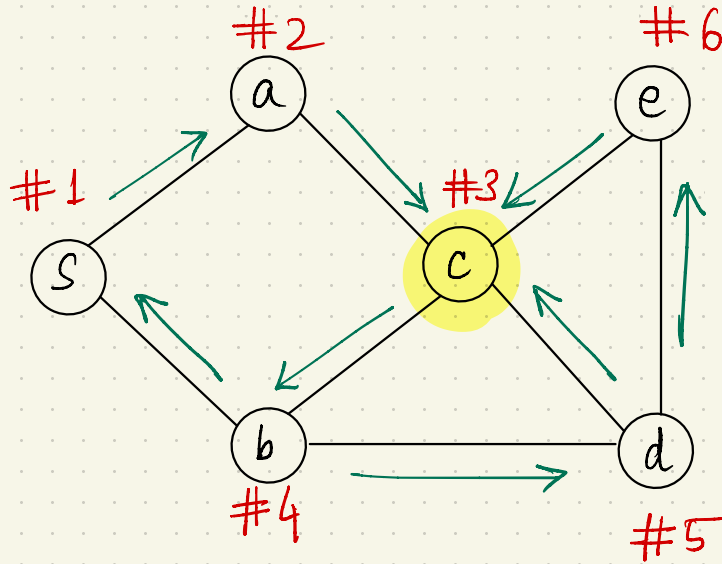
No unexplored neighbors of d $\Rightarrow$ backtrack!

DEPTH-FIRST SEARCH



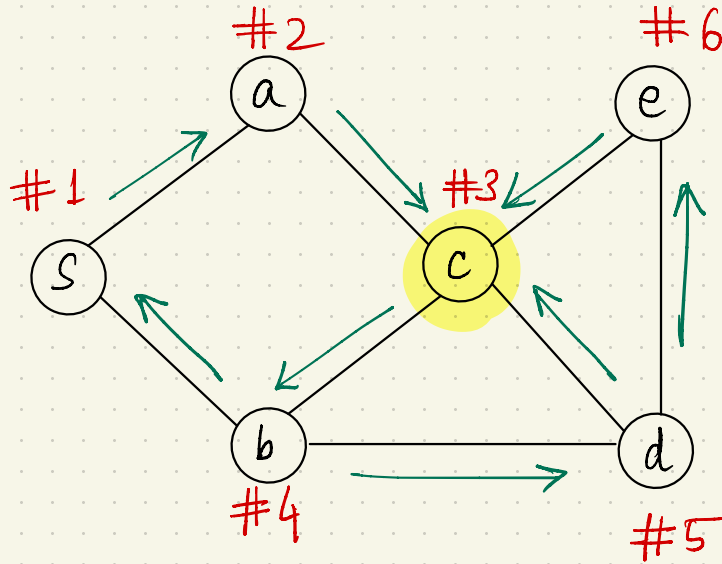
No unexplored neighbors of b $\Rightarrow$ backtrack!

DEPTH-FIRST SEARCH



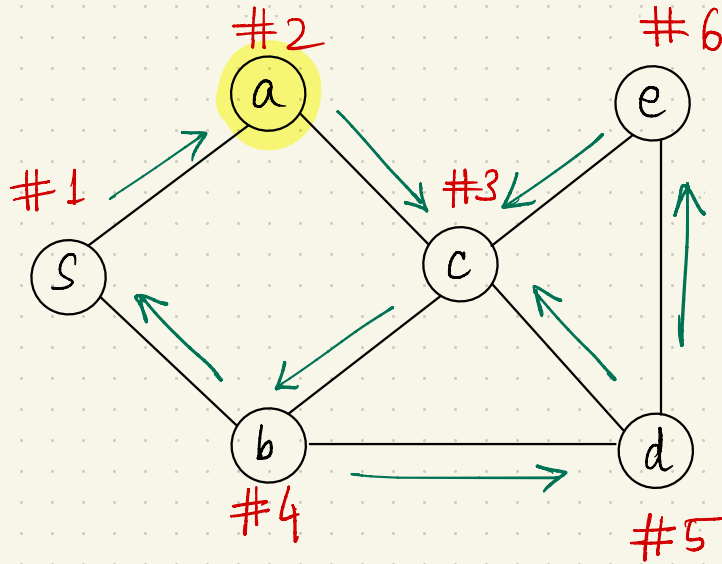
No unexplored neighbors of b $\Rightarrow$ backtrack!

DEPTH-FIRST SEARCH



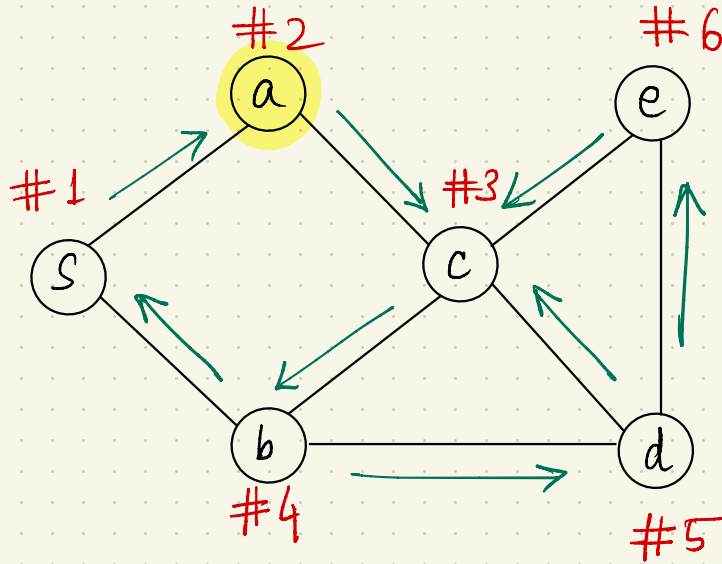
No unexplored neighbors of c $\Rightarrow$ backtrack!

DEPTH-FIRST SEARCH



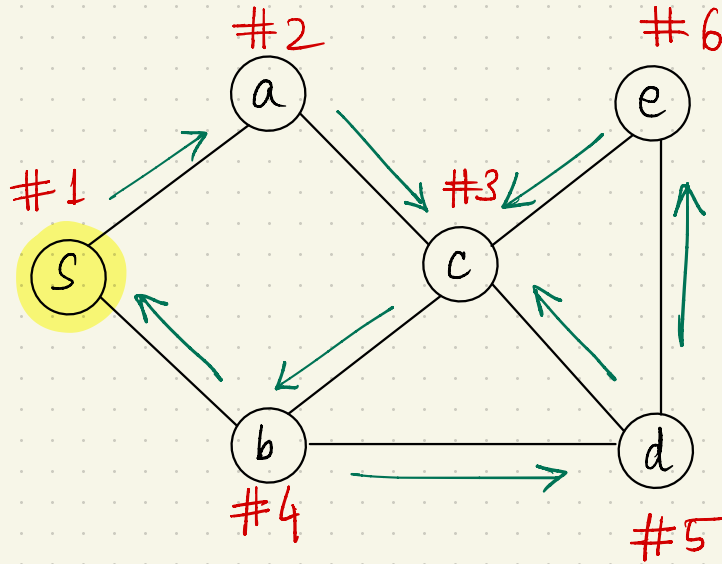
No unexplored neighbors of c $\Rightarrow$ backtrack!

DEPTH-FIRST SEARCH



No unexplored neighbors of a $\Rightarrow$ backtrack!

DEPTH-FIRST SEARCH



Back home!

DEPTH-FIRST SEARCH

Iterative version

DEPTH-FIRST SEARCH

Iterative version

mark all vertices as **unexplored**

$S :=$ a **stack** data structure (LIFO), initialized with s

while $S \neq \emptyset$

- | remove the top node of S , say v ("pop")
 - | if v is **unexplored**
 - | mark v as **explored**
 - | for each (v, w) in adj. list of v
 - | add w to the front of S ("push")

DEPTH-FIRST SEARCH

Iterative version

mark all vertices as **unexplored**

$S :=$ a **stack** data structure (**LIFO**), initialized with s

while $S \neq \emptyset$

- | remove the top node of S , say v ("pop")
 - | if v is **unexplored**
 - | mark v as **explored**
 - | for each (v, w) in adj. list of v
 - | add w to the **front** of S ("push")

DEPTH-FIRST SEARCH

Recursive version

DEPTH-FIRST SEARCH

Recursive version

DFS (G, s) // all vertices **unexplored** before the call

mark s as **explored**

for each (s, v) in adj. list of s

└ if v is **unexplored**
└└ DFS (G, v)

DEPTH-FIRST SEARCH

Claim 1: At the end of DFS

v is explored $\iff \exists$ path from s to v in G

DEPTH-FIRST SEARCH

Claim 1: At the end of DFS

v is explored $\iff \exists$ path from s to v in G

Reason: DFS is a special case of Generic Algorithm.

DEPTH-FIRST SEARCH

Claim 1: At the end of DFS

v is explored $\iff \exists$ path from s to v in G

Reason: DFS is a special case of Generic Algorithm.

Claim 2: Running time is $O(n_s + m_s)$.

reachable vertices

reachable edges

DEPTH-FIRST SEARCH

Claim 1: At the end of DFS

v is explored $\iff \exists$ path from s to v in G

Reason: DFS is a special case of Generic Algorithm.

Claim 2: Running time is $O(n_s + m_s)$.

reachable vertices

reachable edges

Corollary: DFS can compute connected components in linear time.