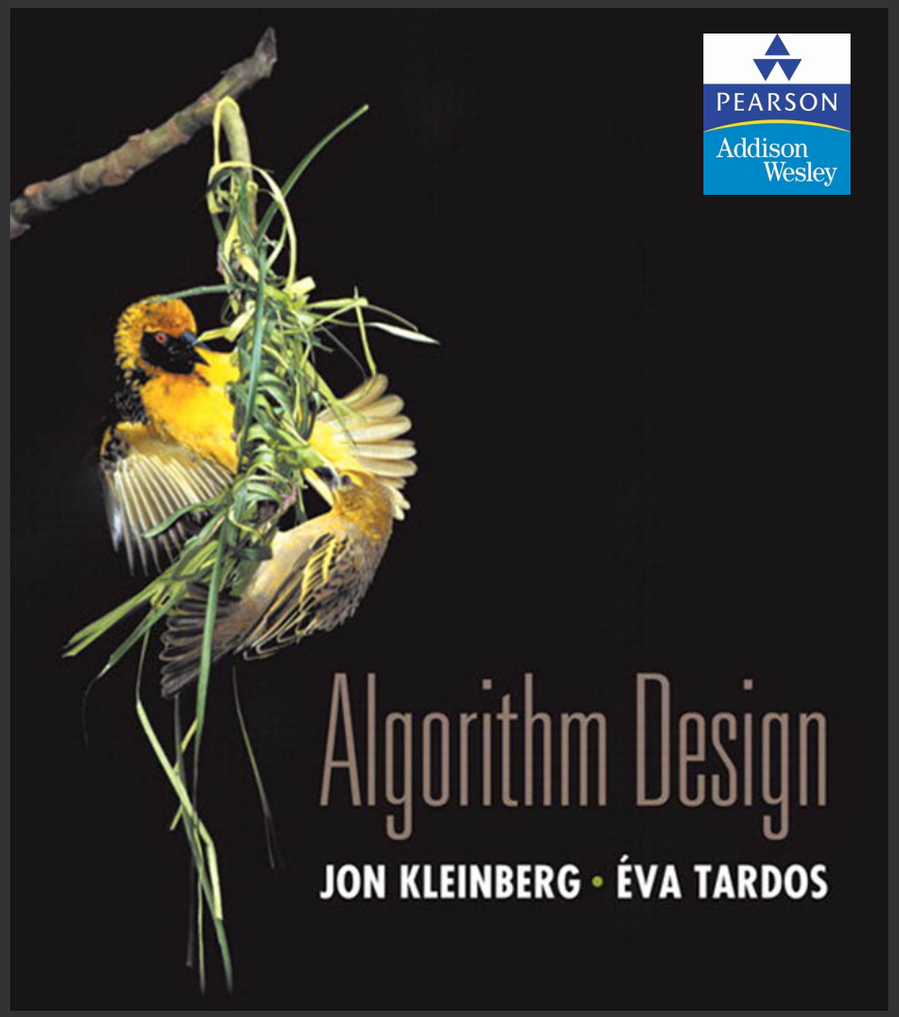
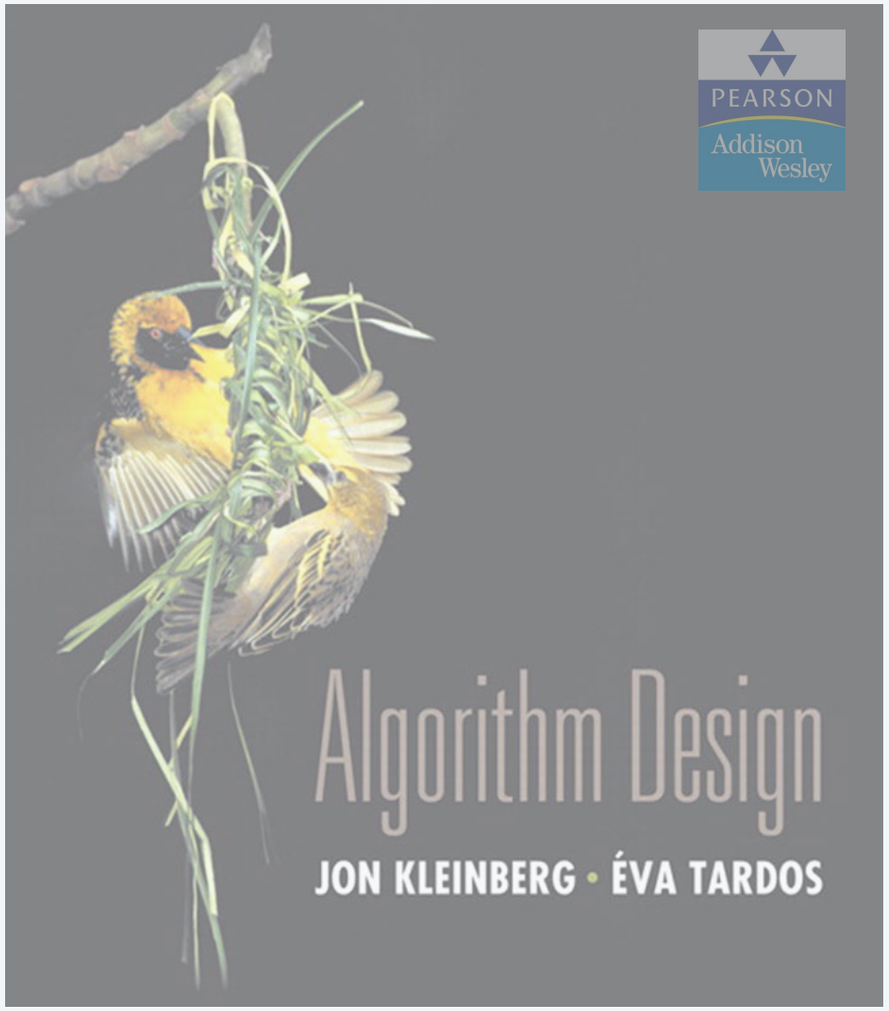


3. GPRAHS

- *BFS demo*
- *DFS demo*

Last updated on 2020/10/9 下午8:36

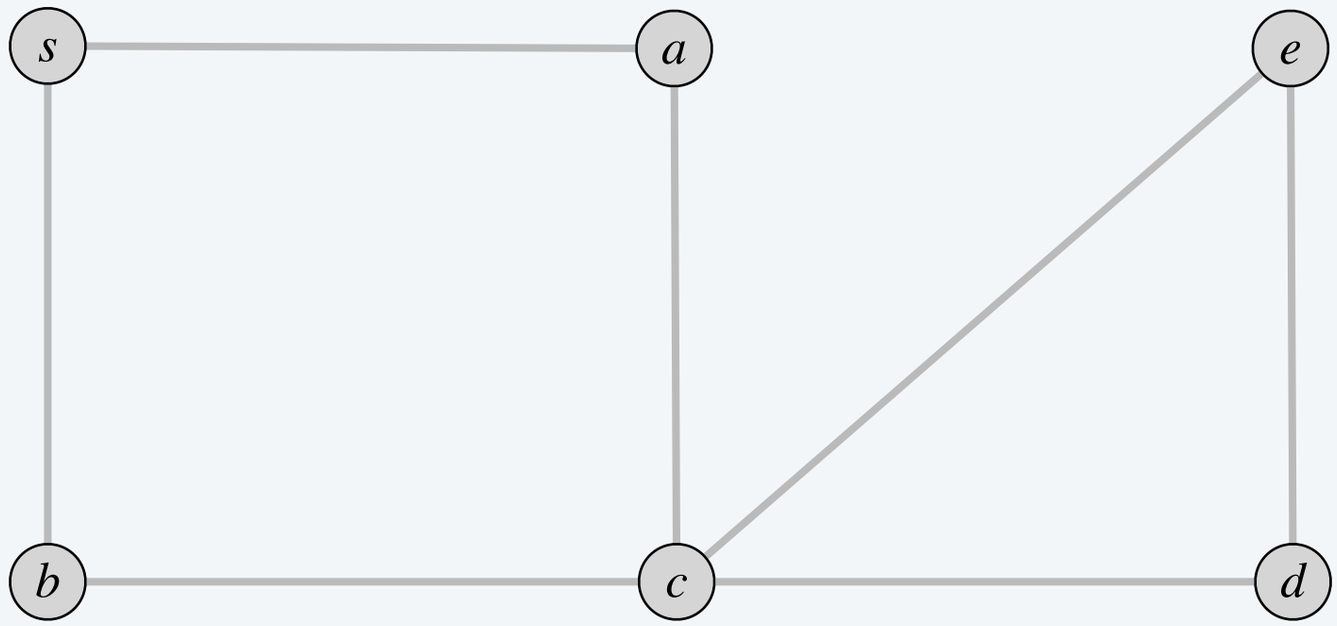


3. GRAPHS

- *BFS demo*
- *DFS demo*

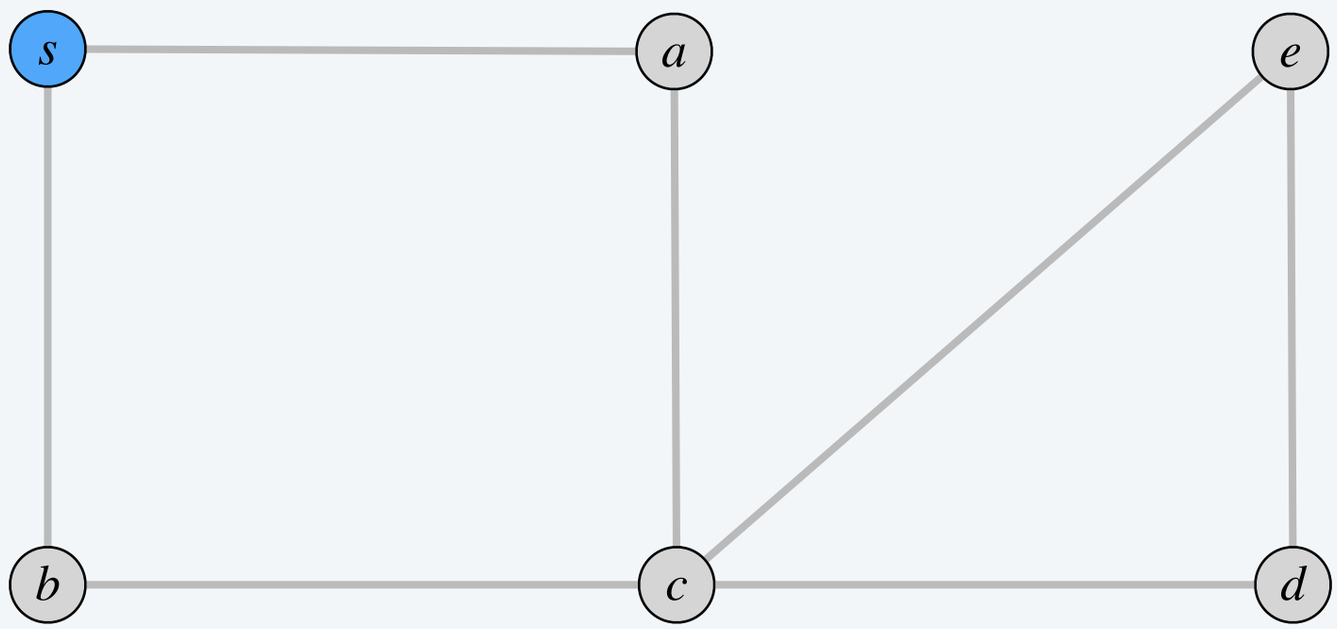
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



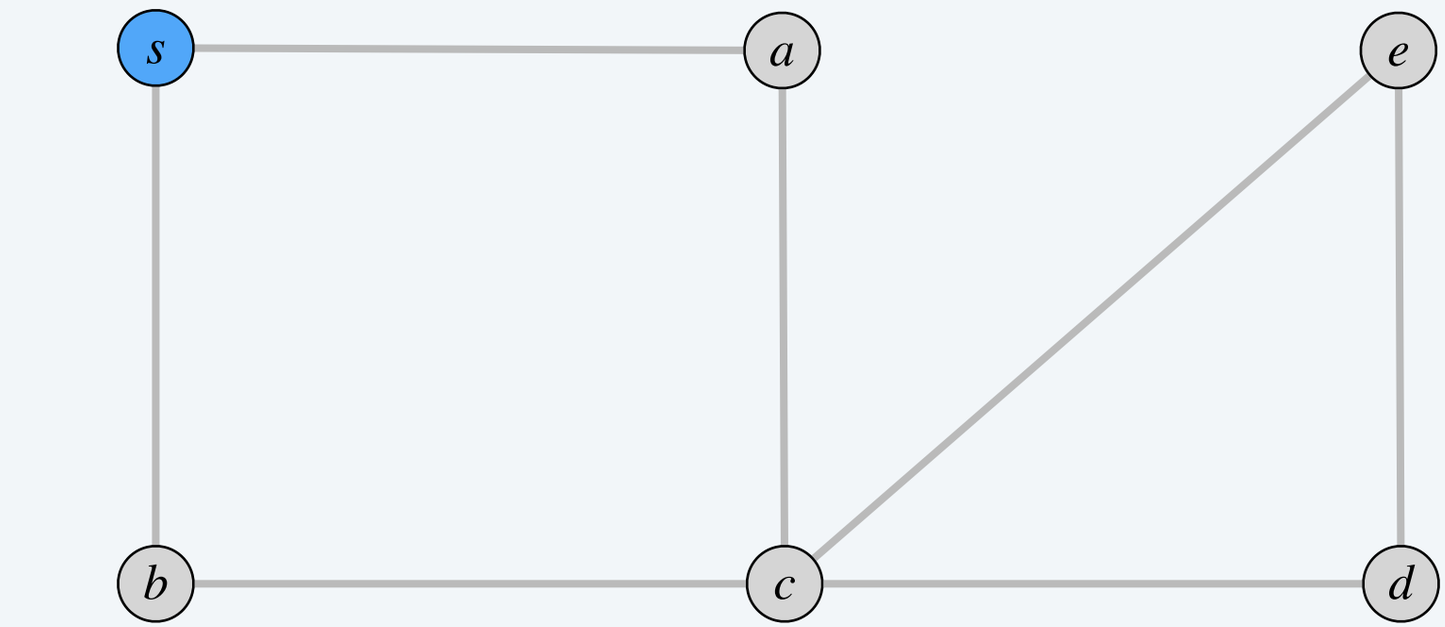
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



BFS demo

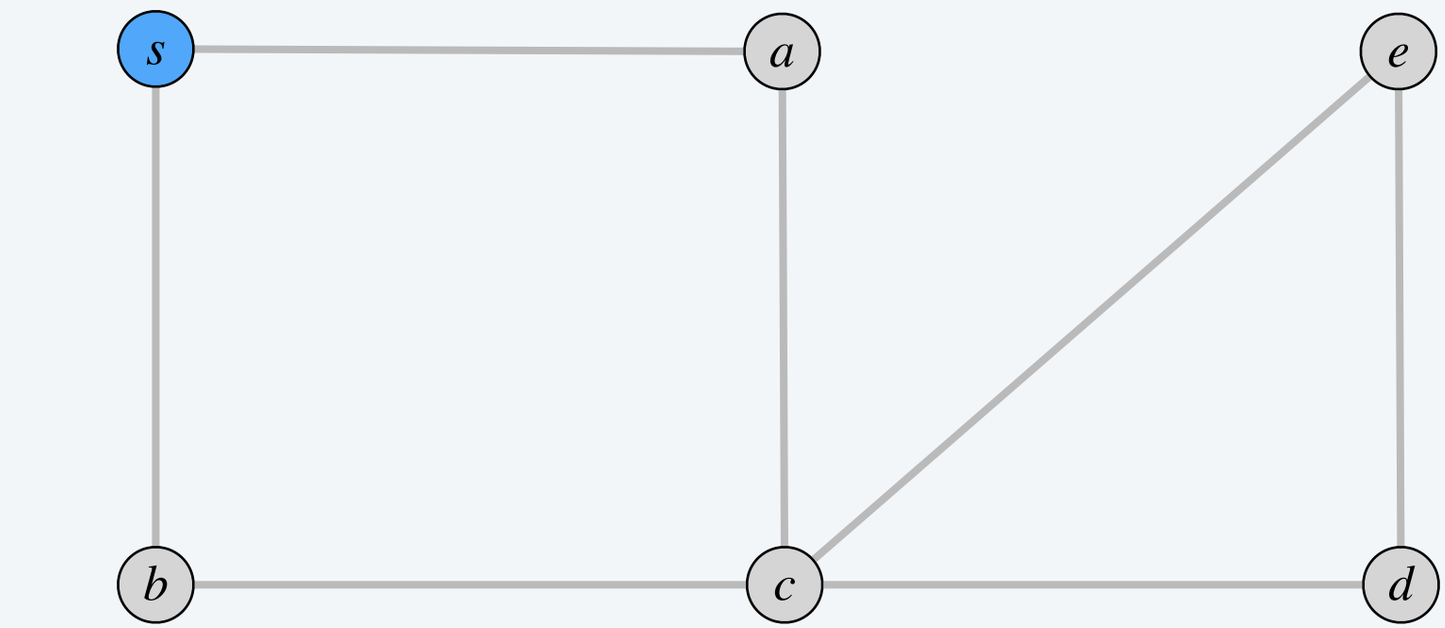
mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



Q | s |

BFS demo

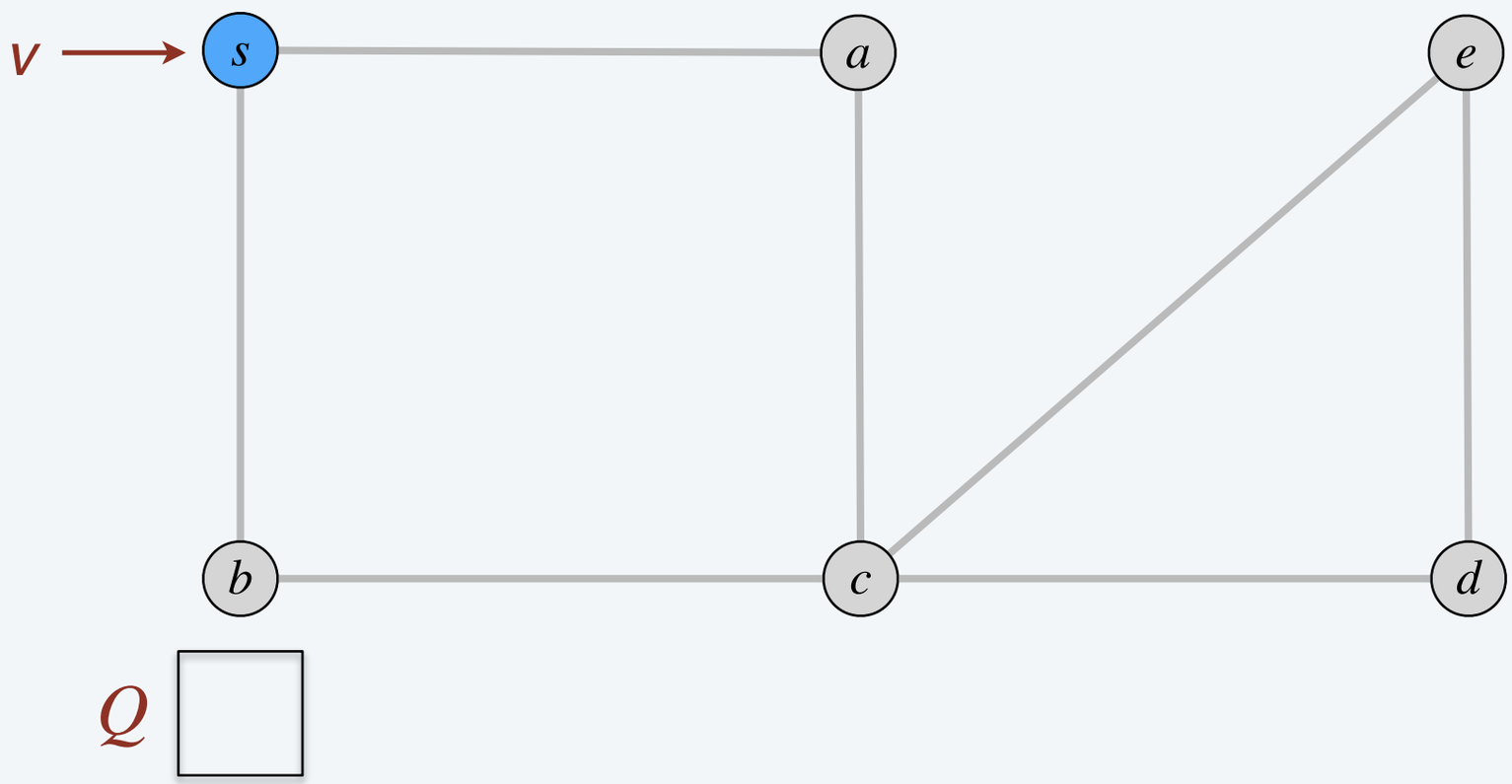
mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



Q | s |

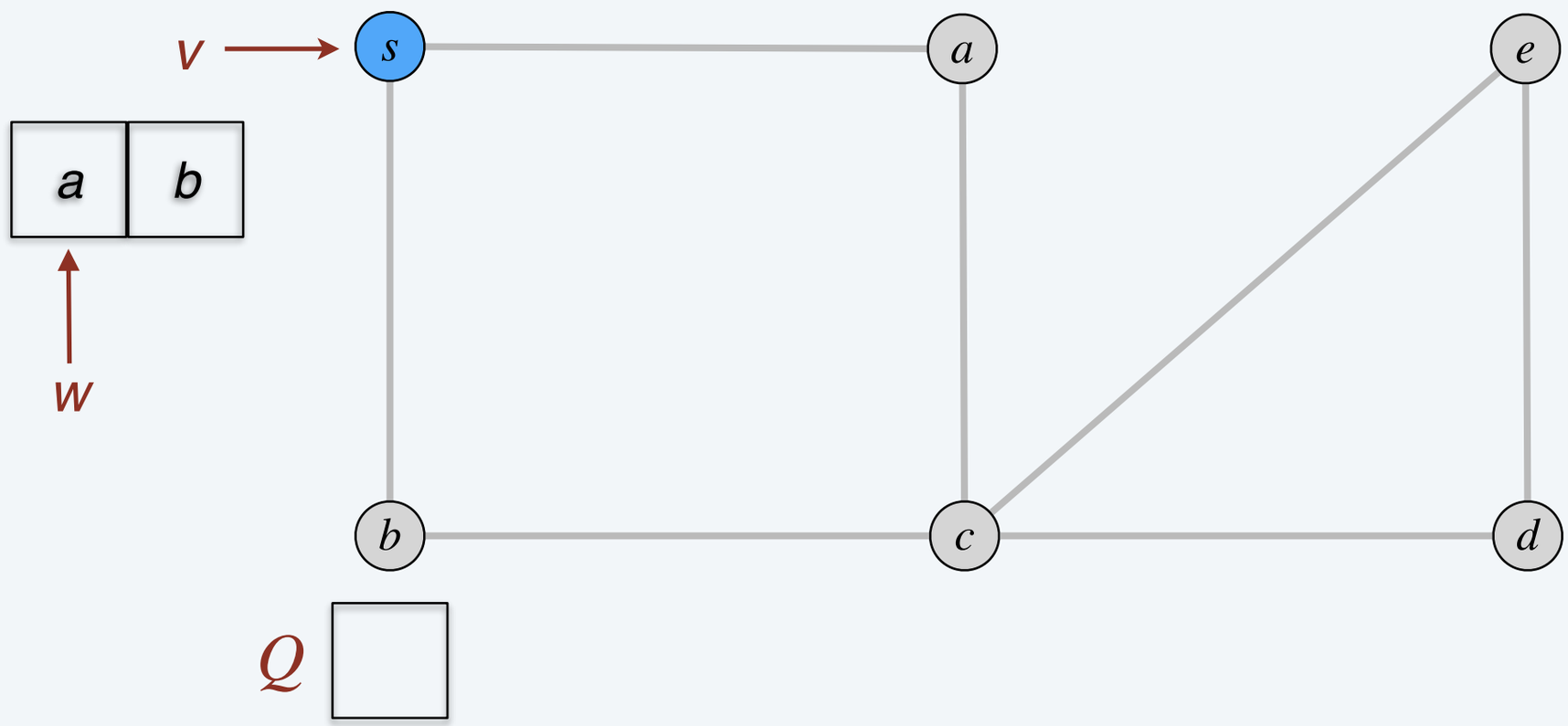
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



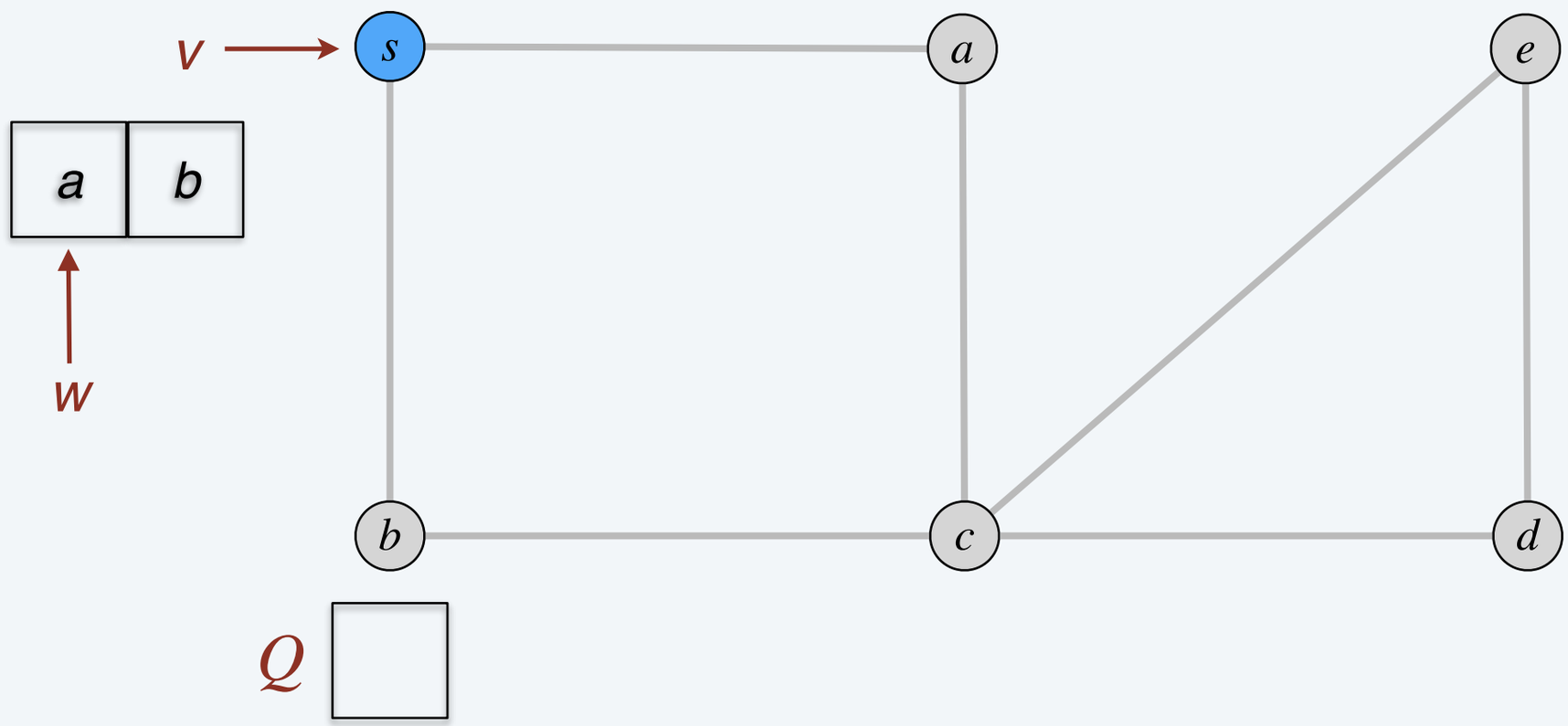
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



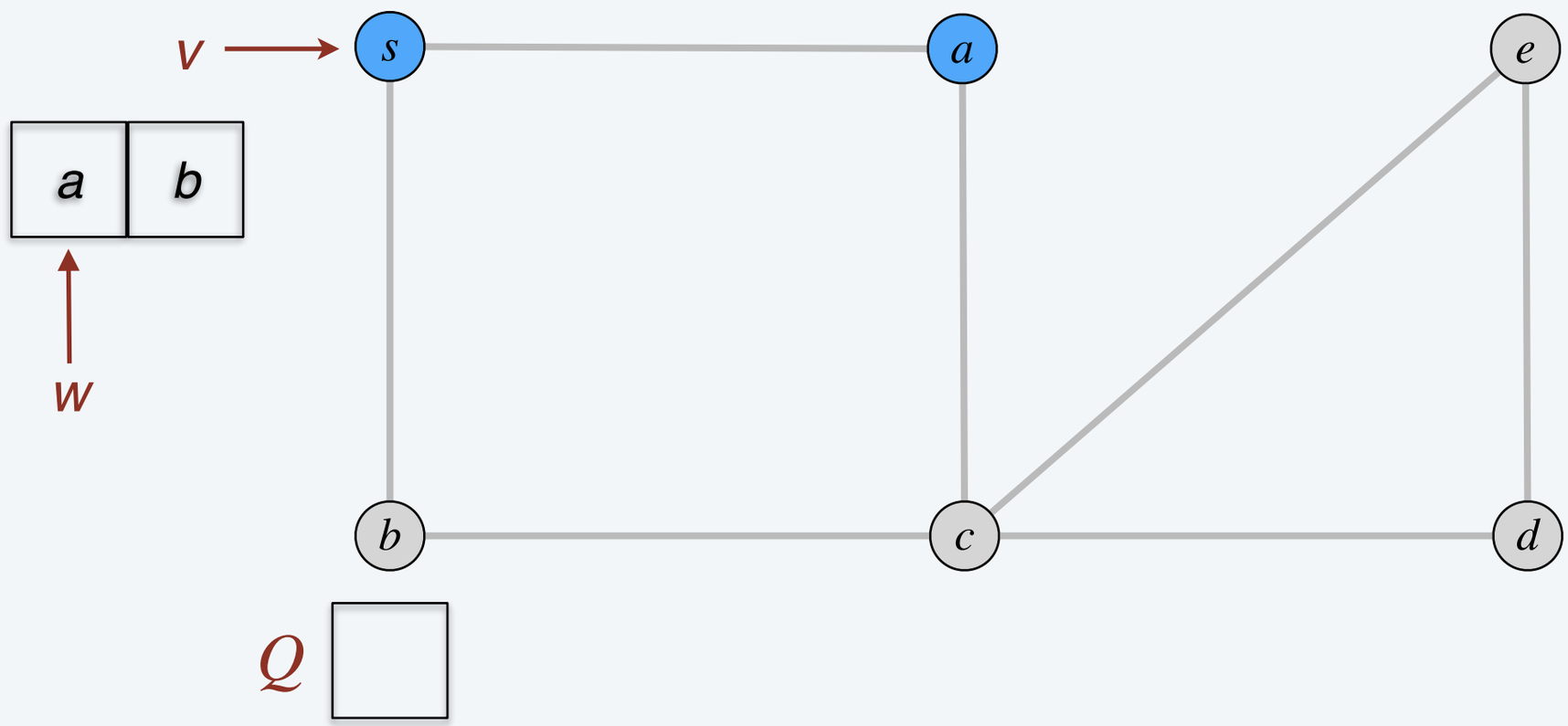
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



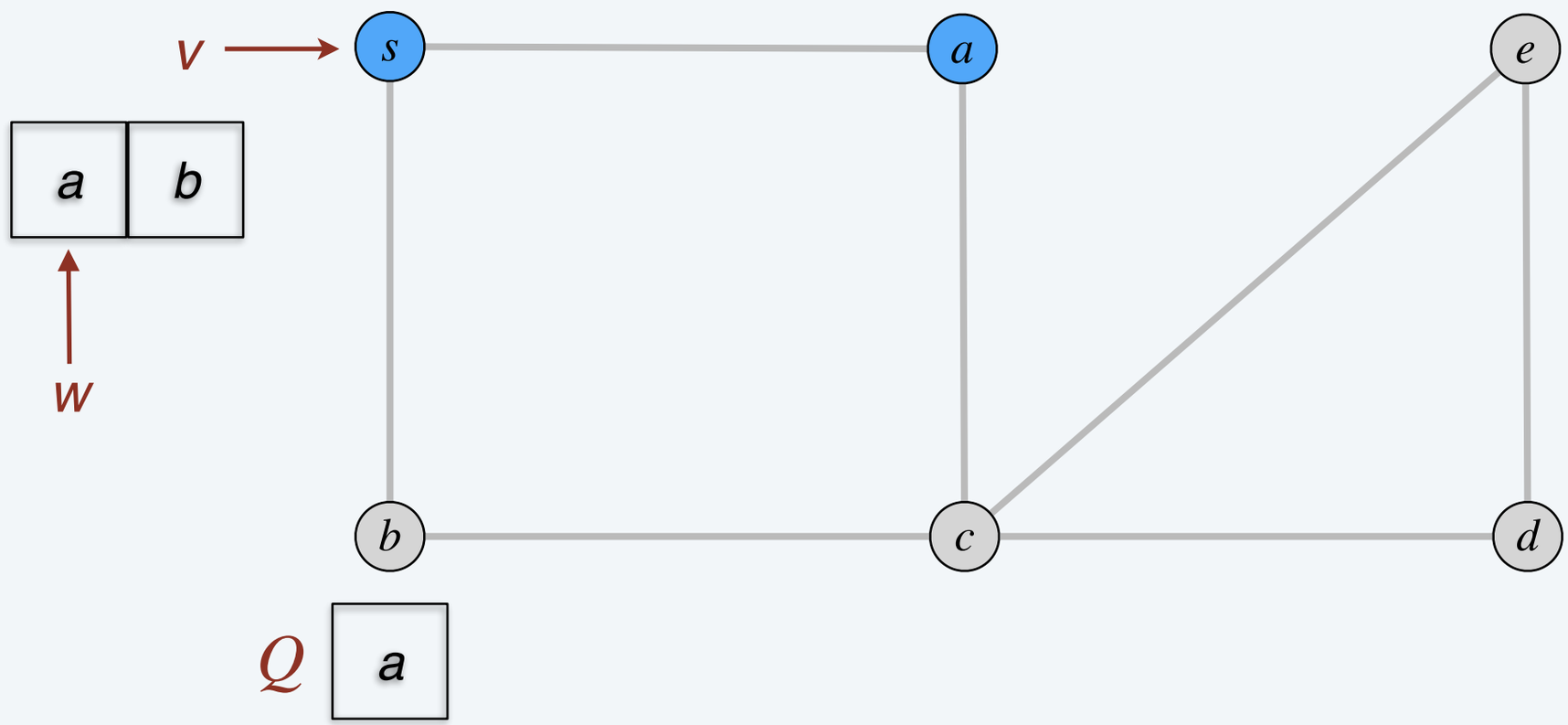
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



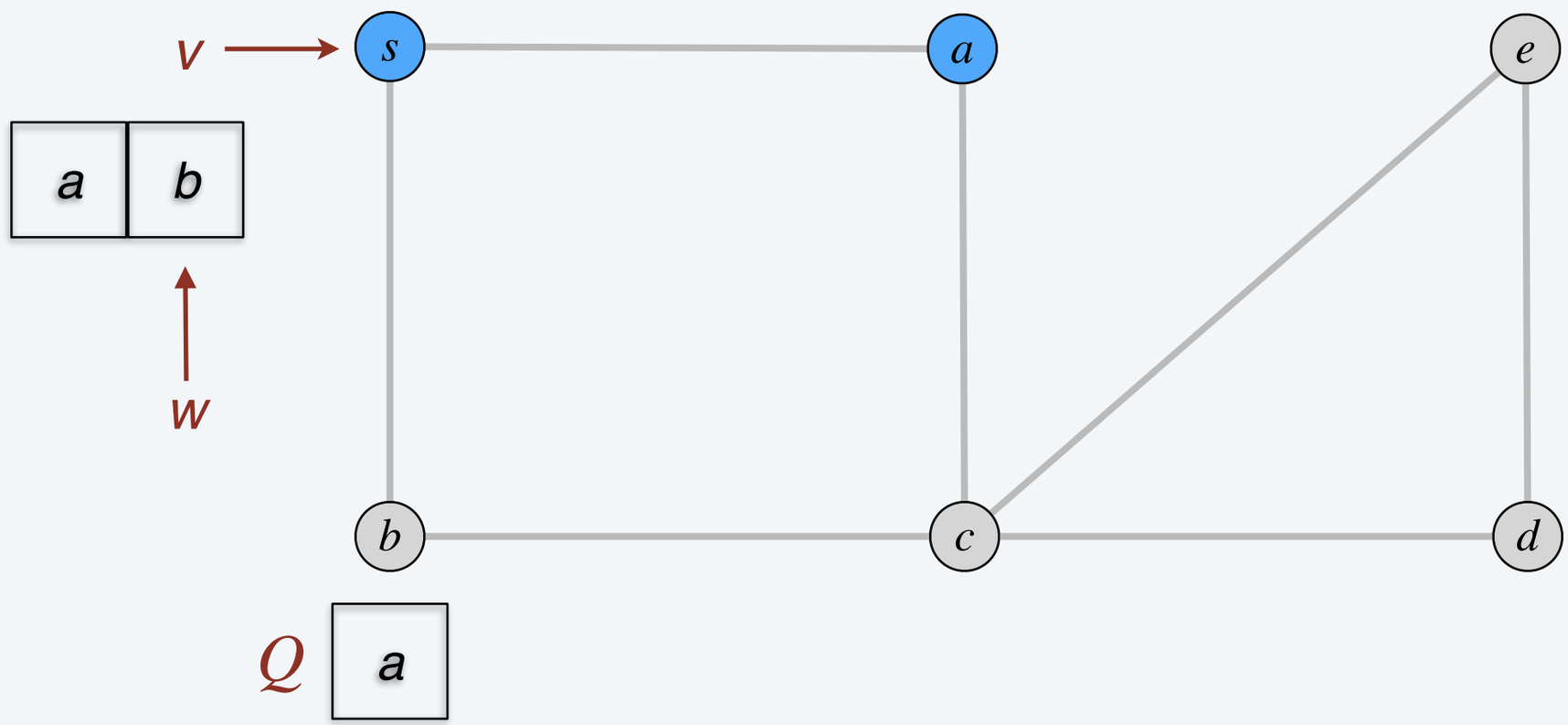
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



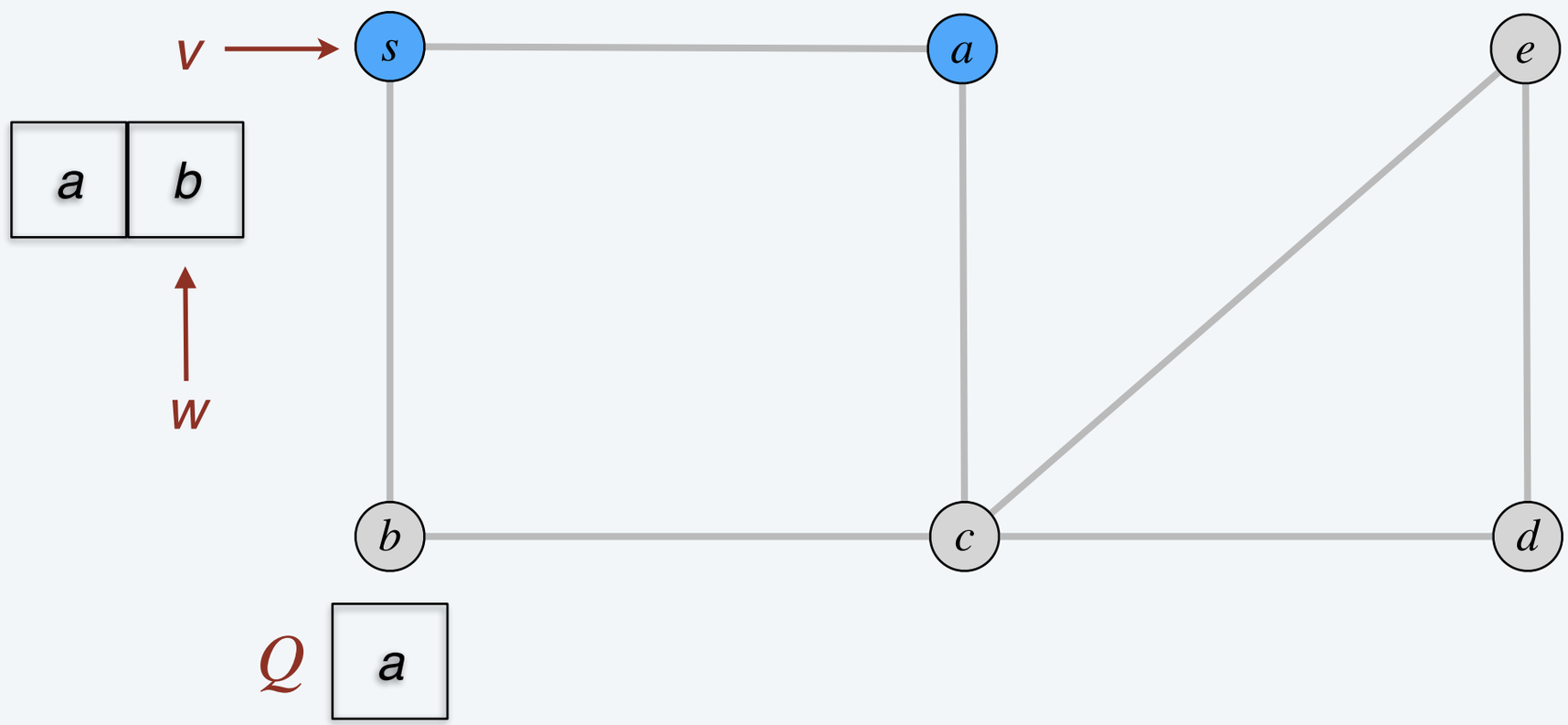
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



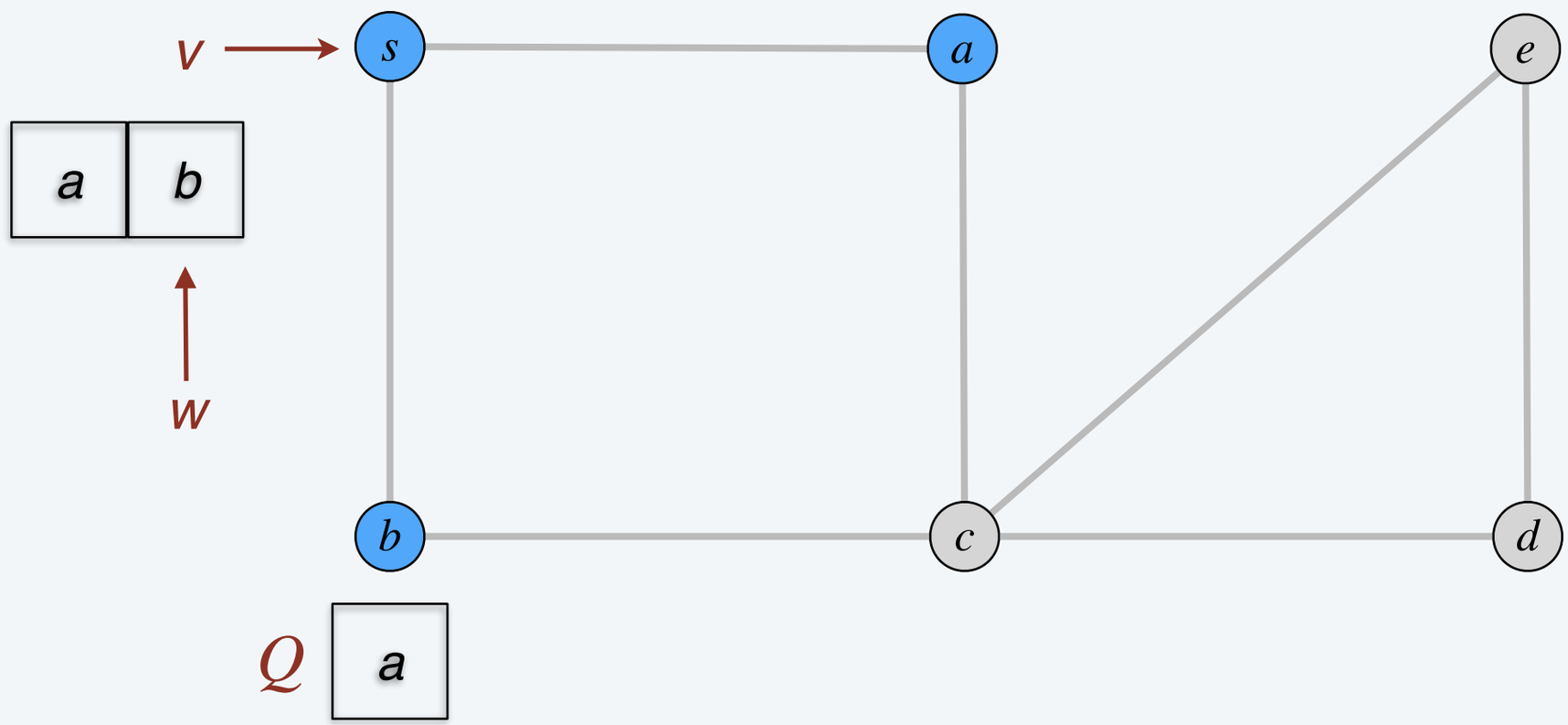
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



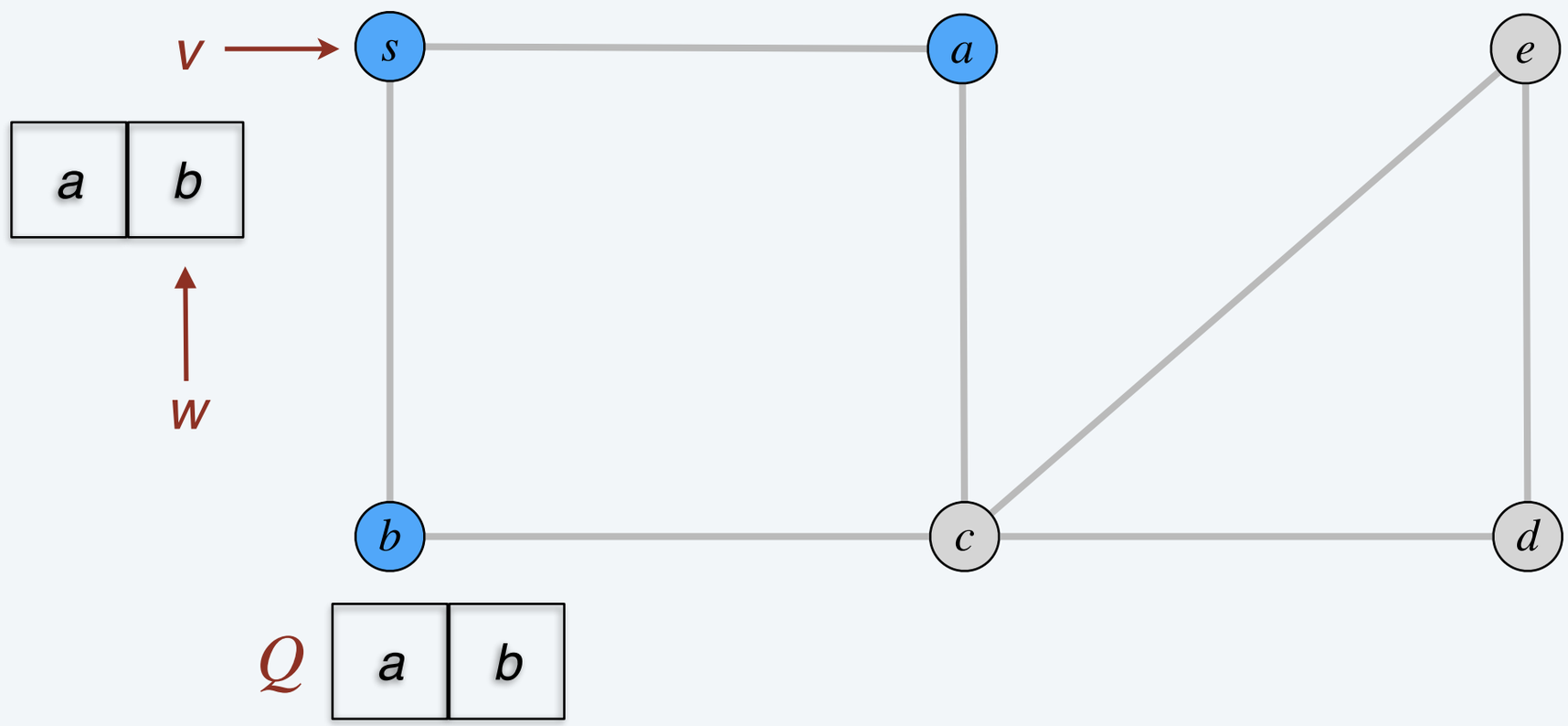
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



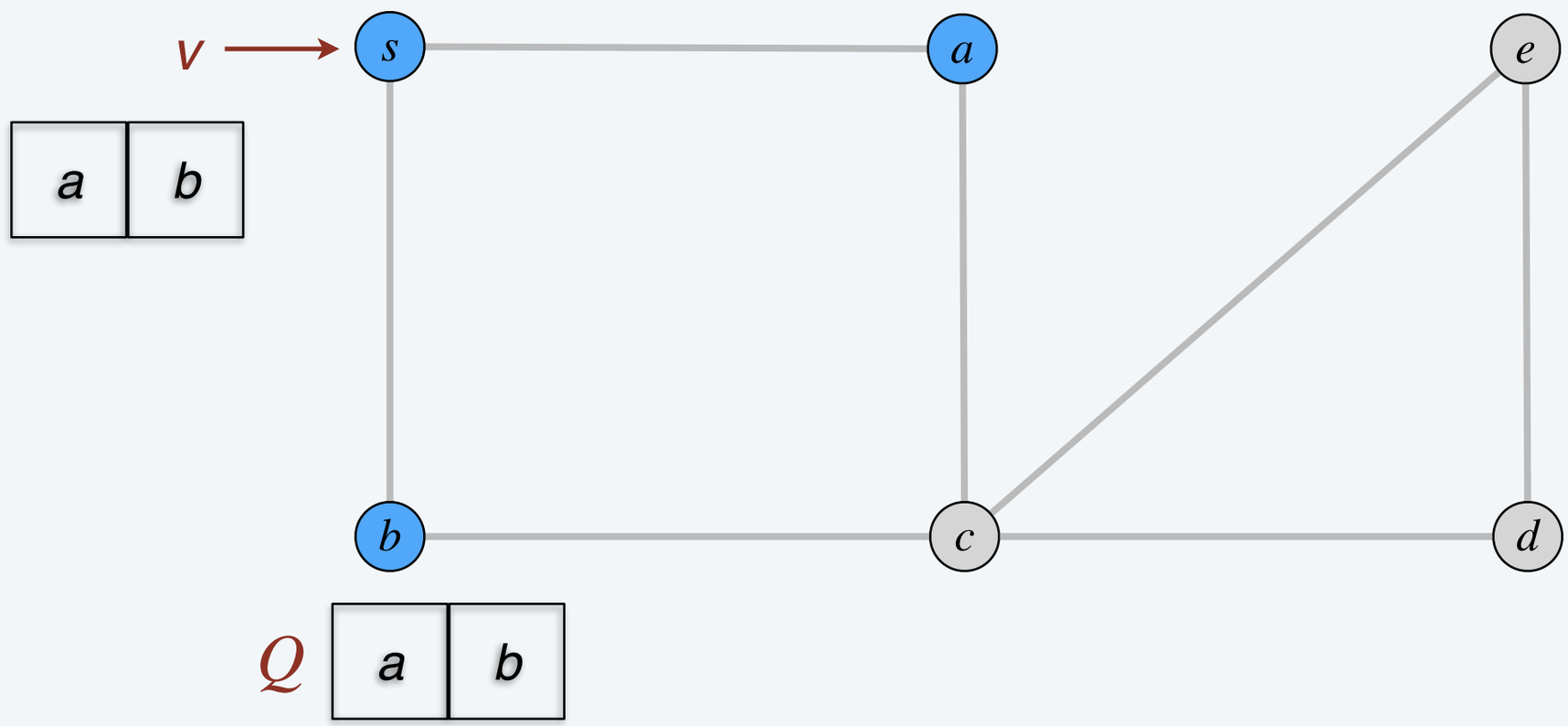
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



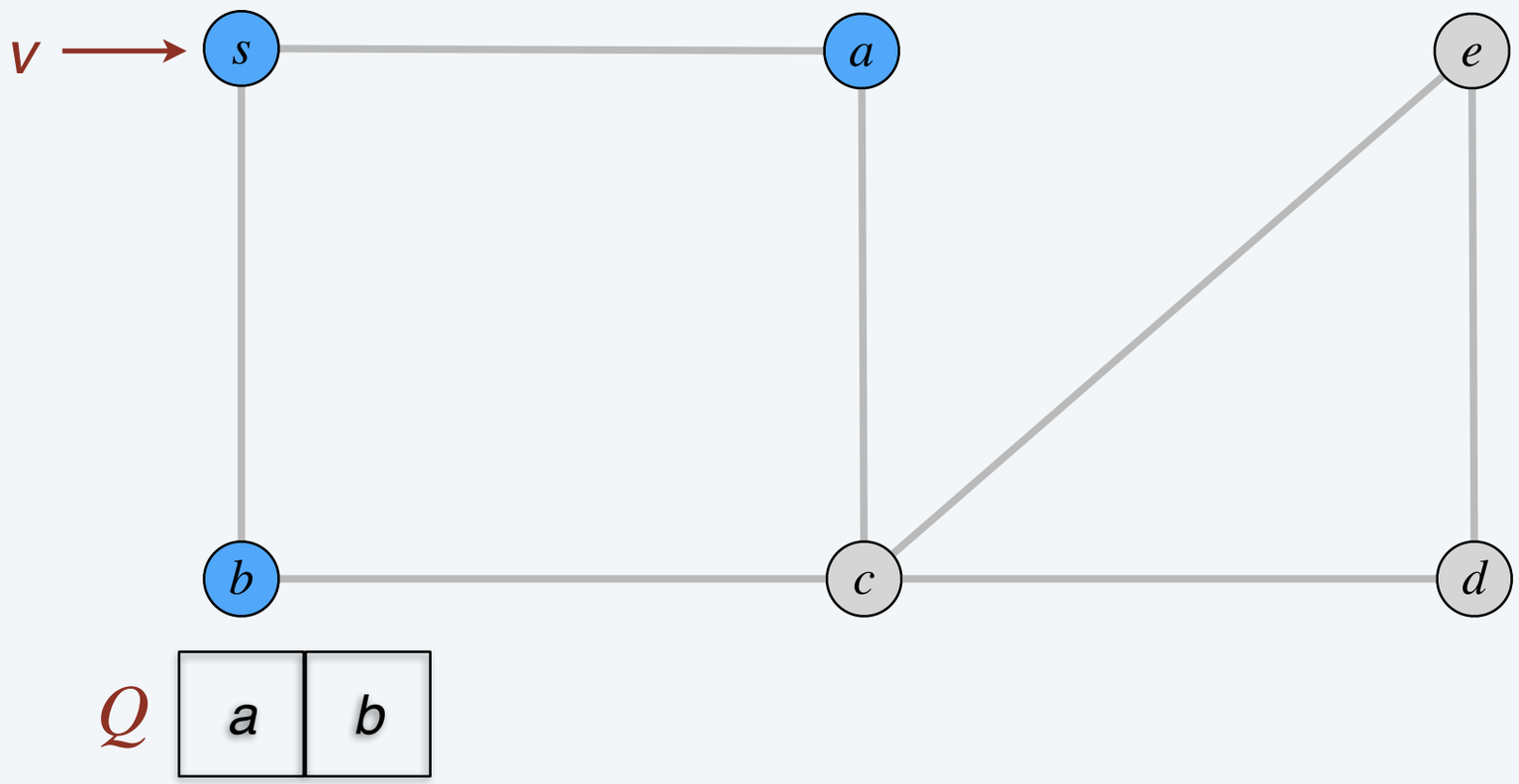
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



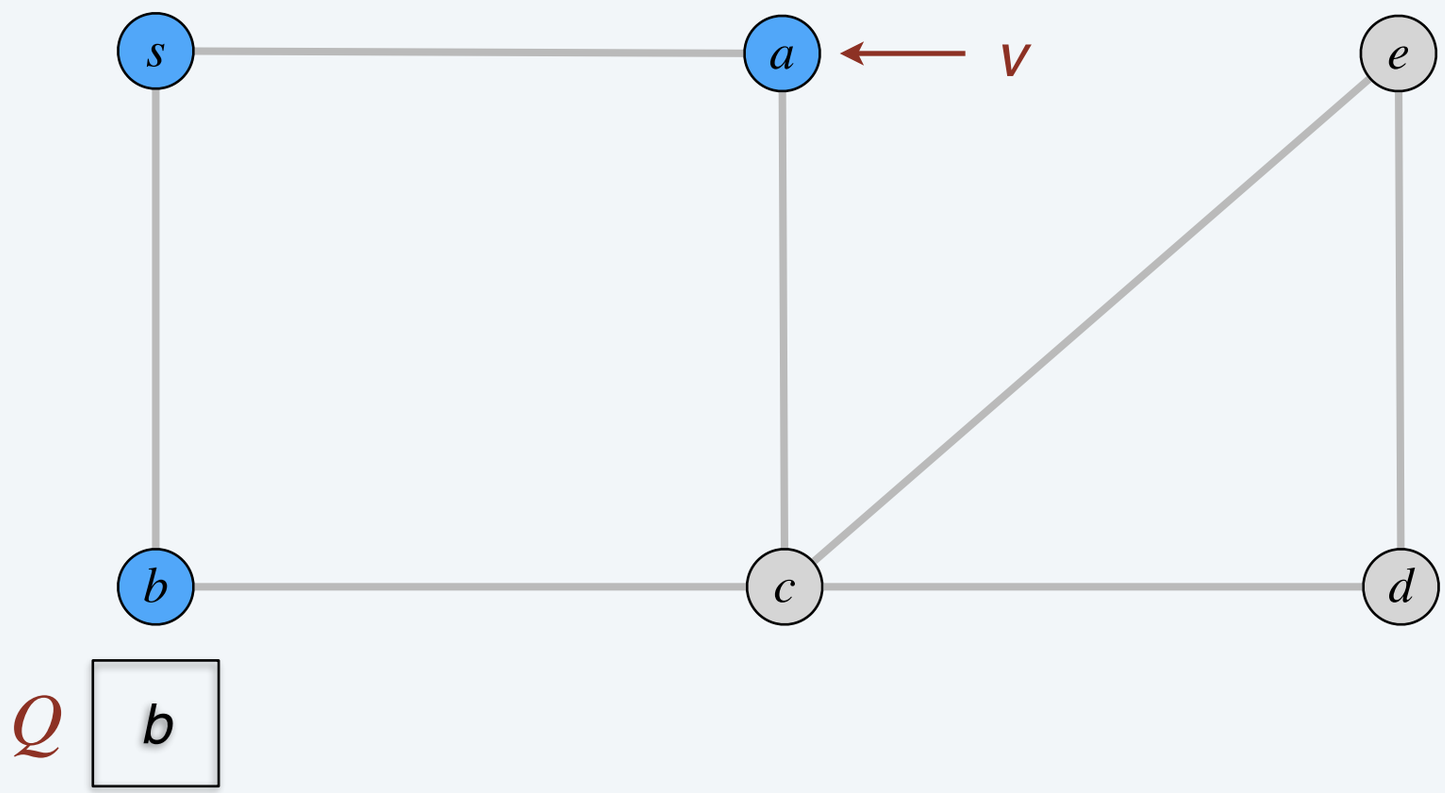
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



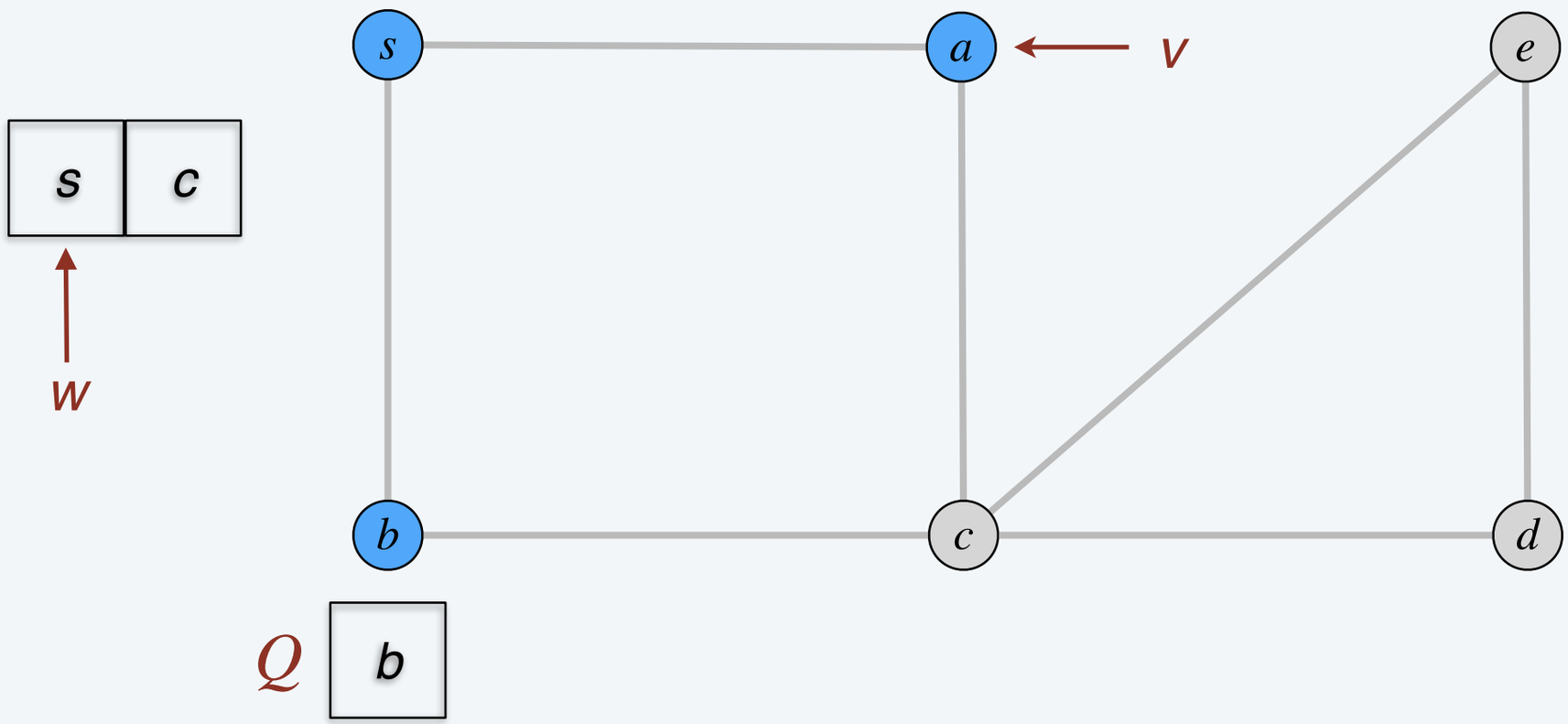
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



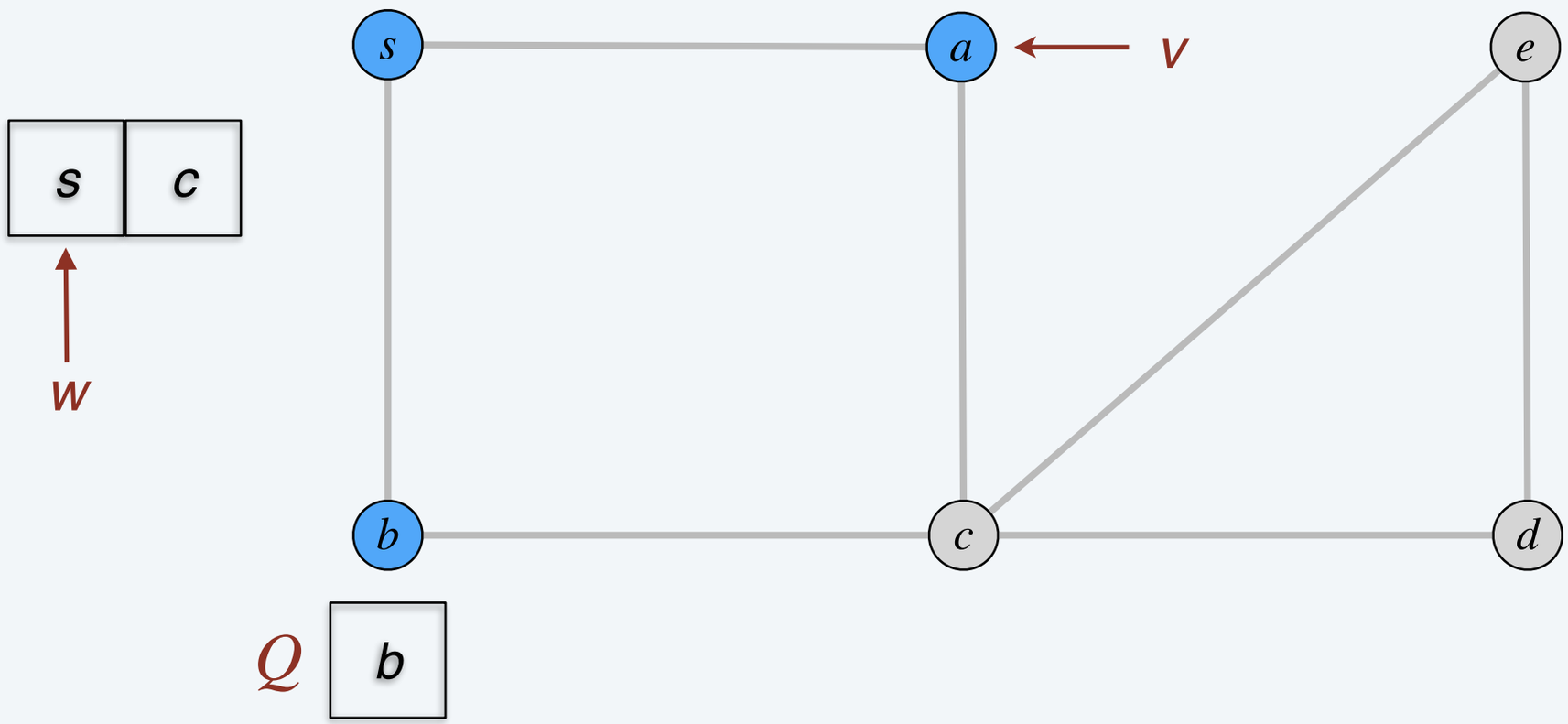
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



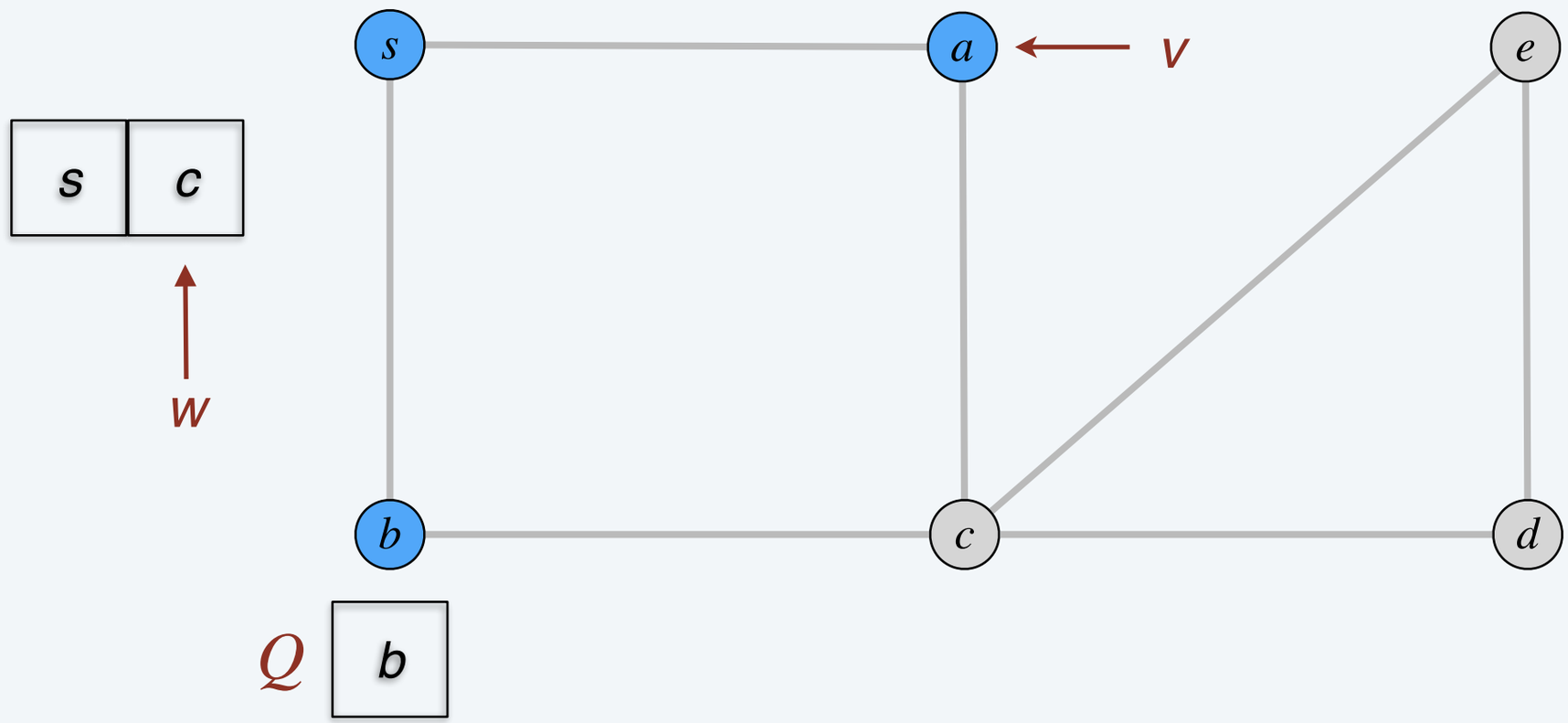
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



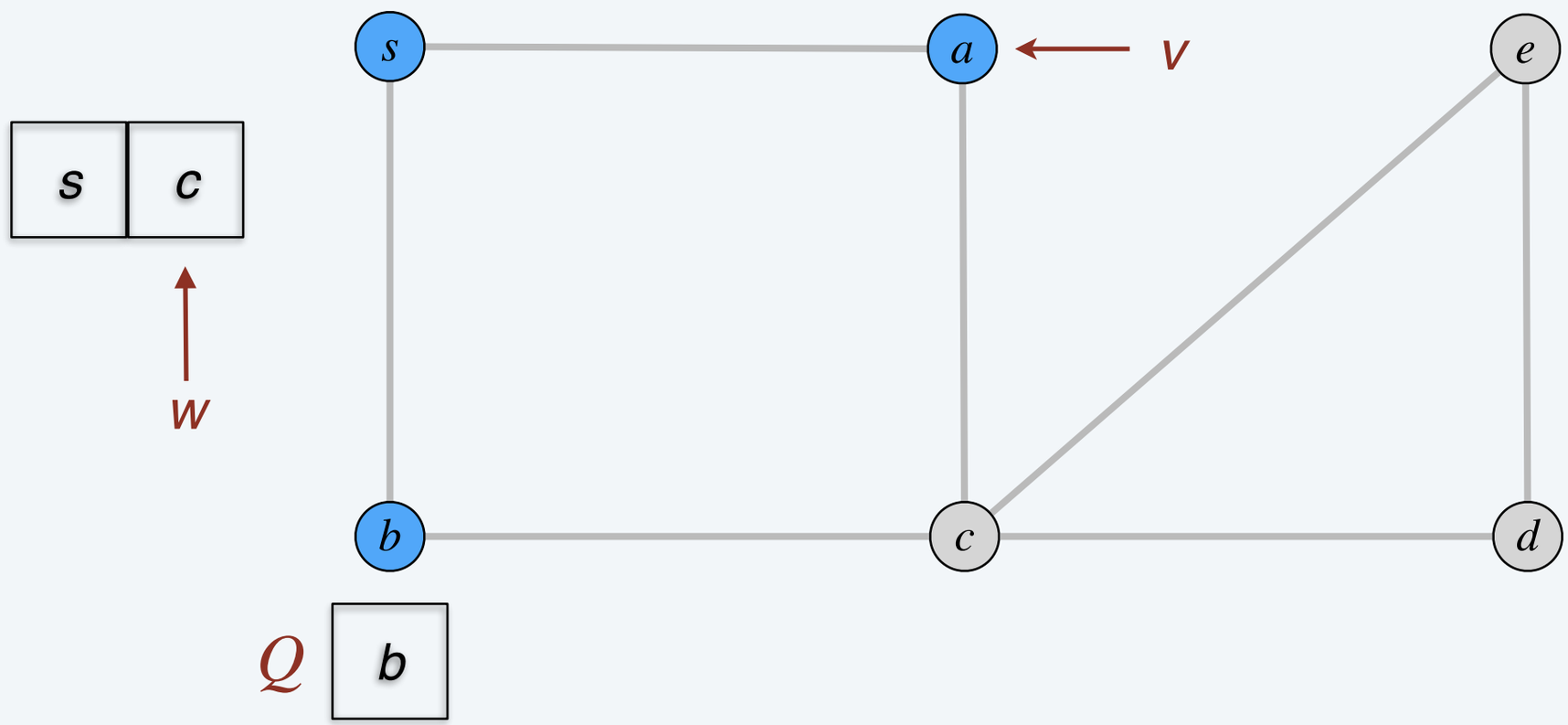
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



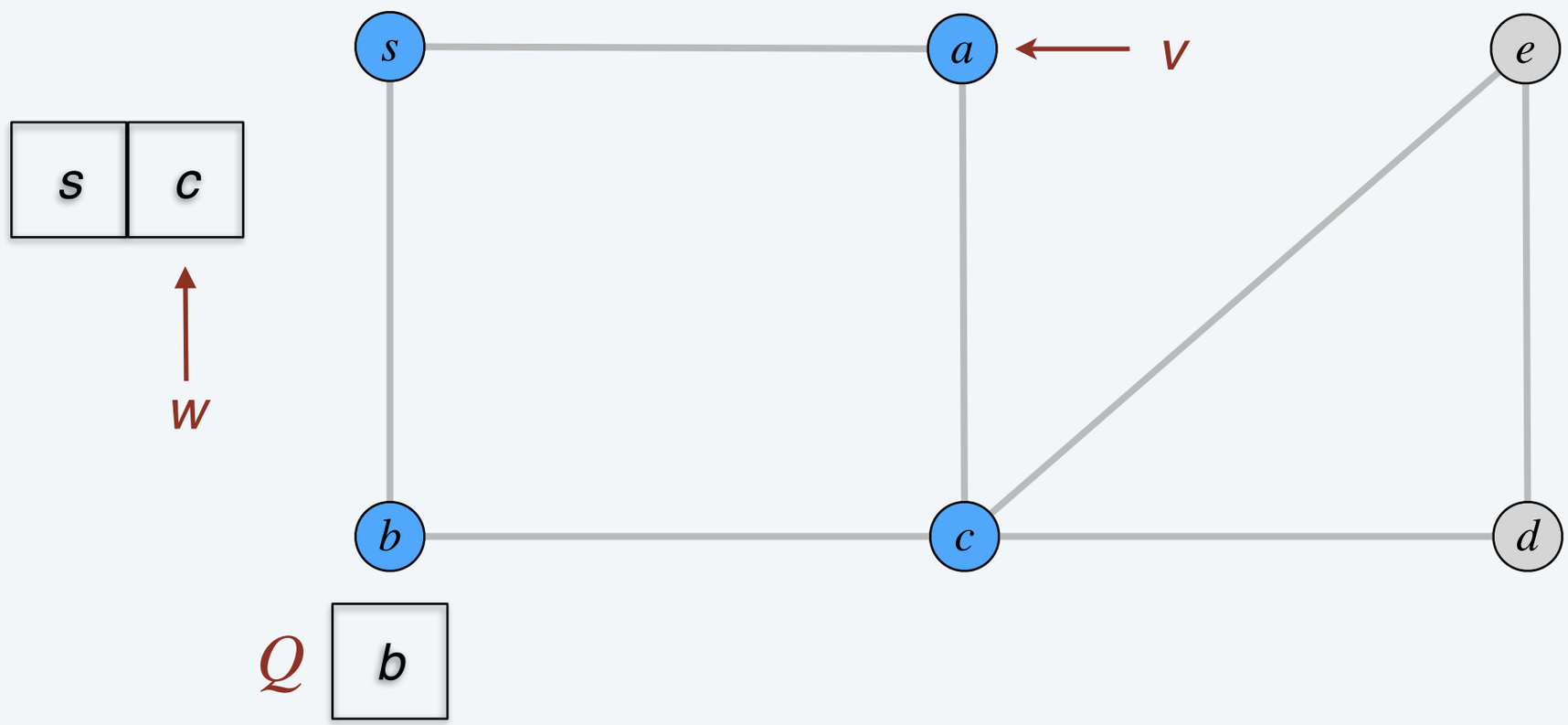
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



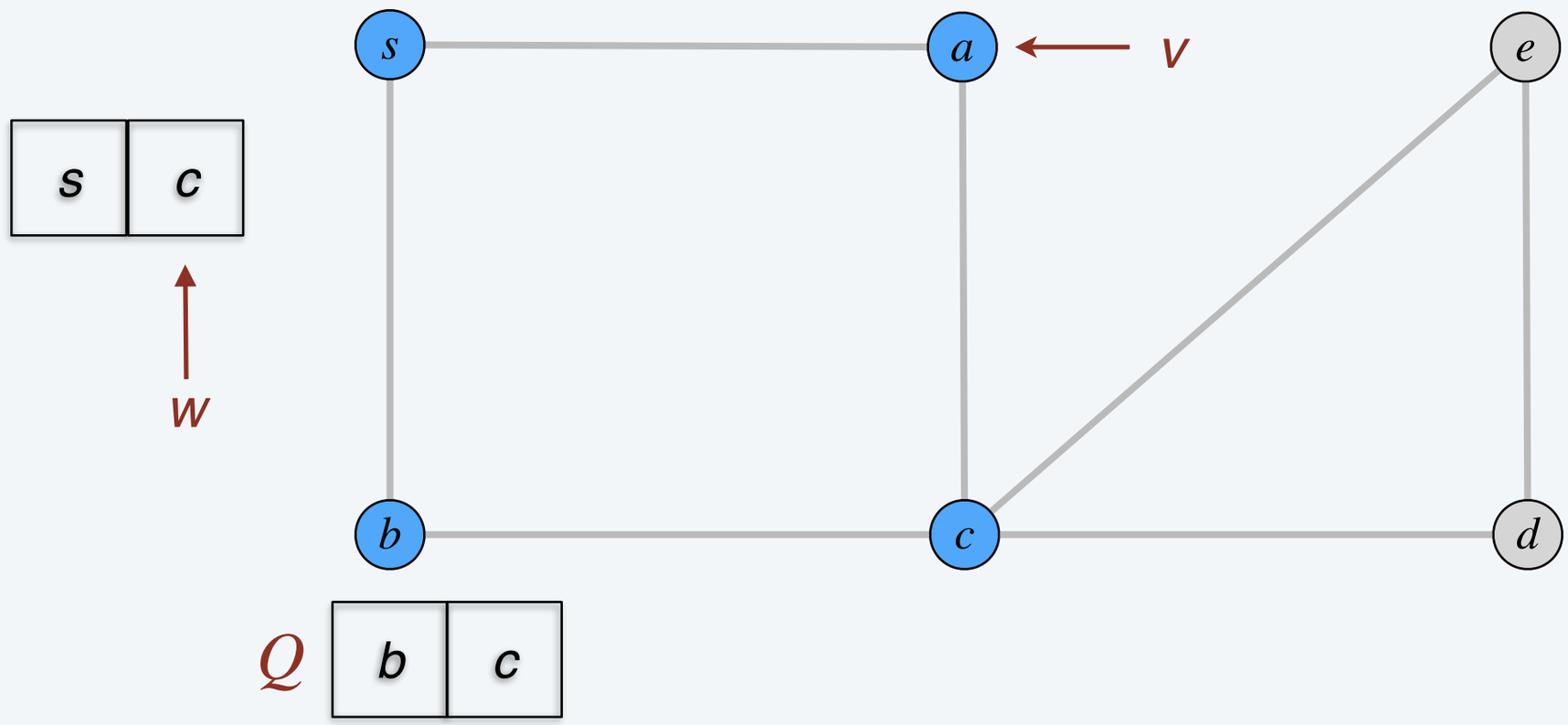
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



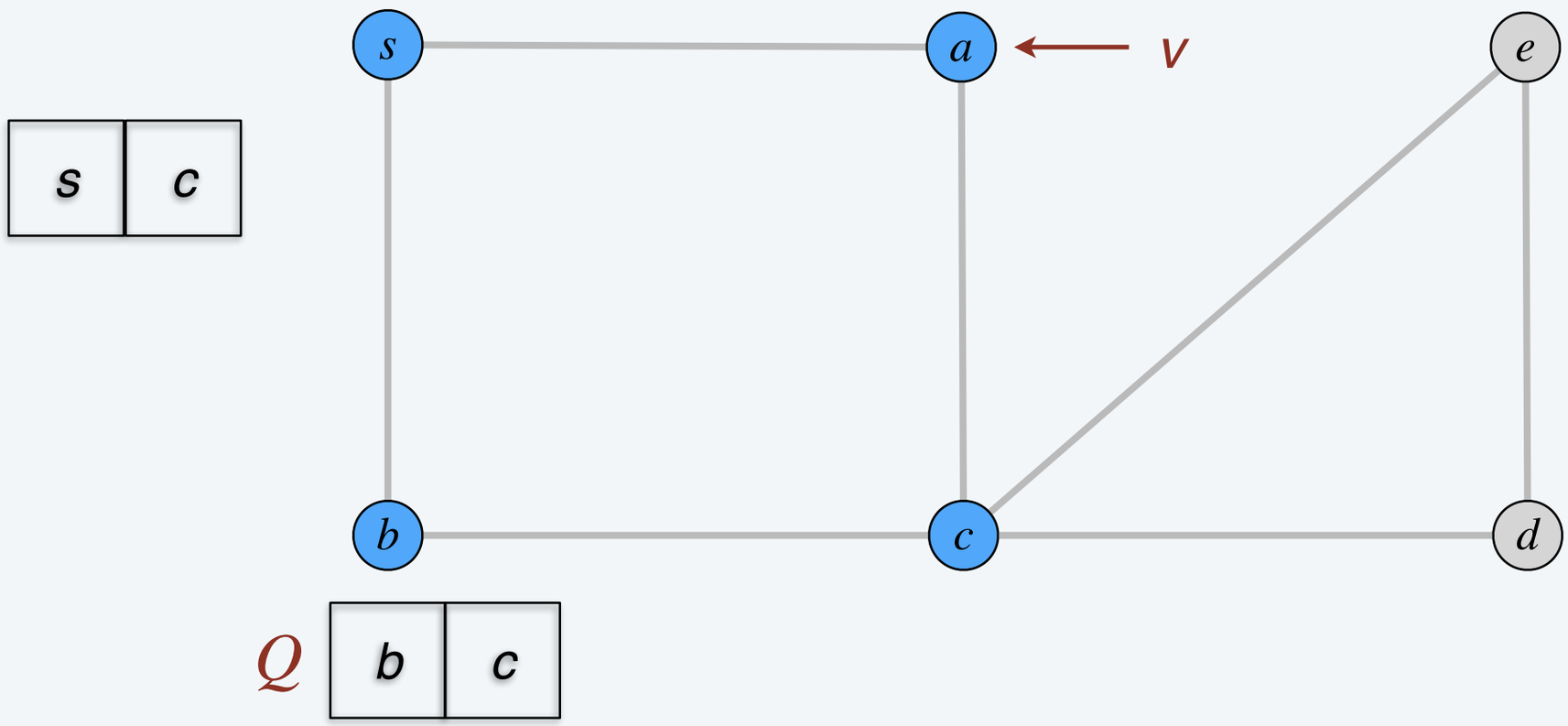
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



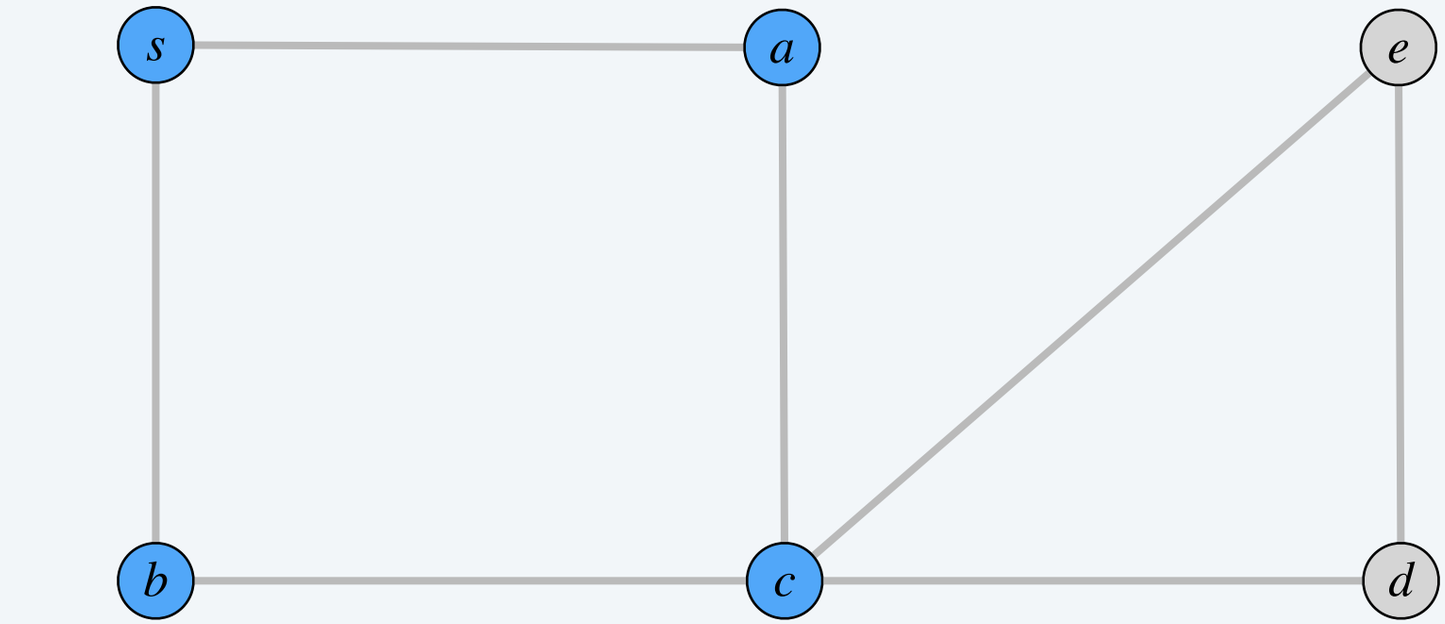
BFS demo

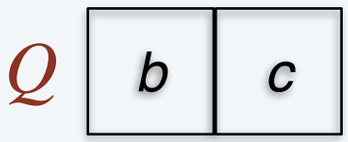
mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



BFS demo

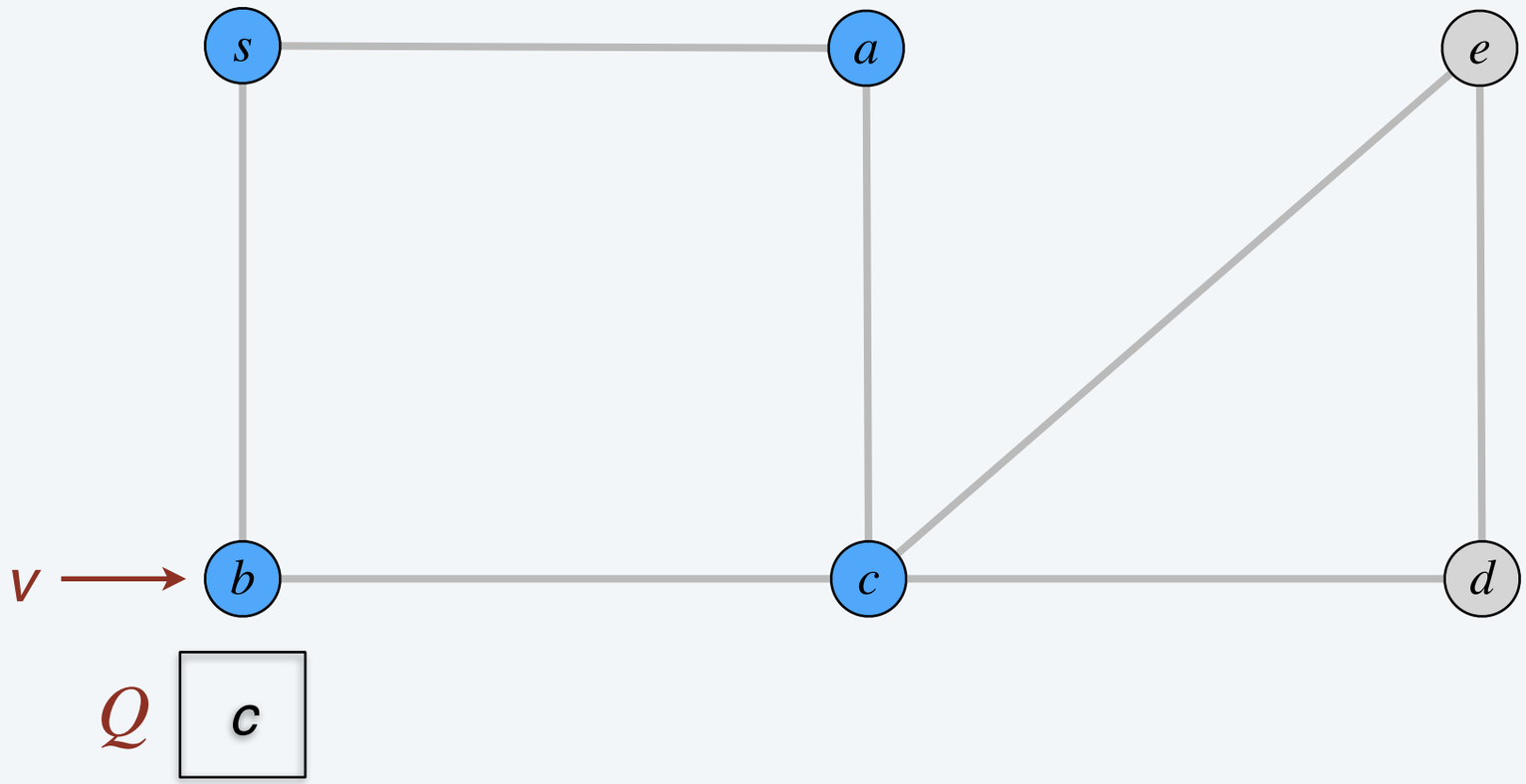
mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q





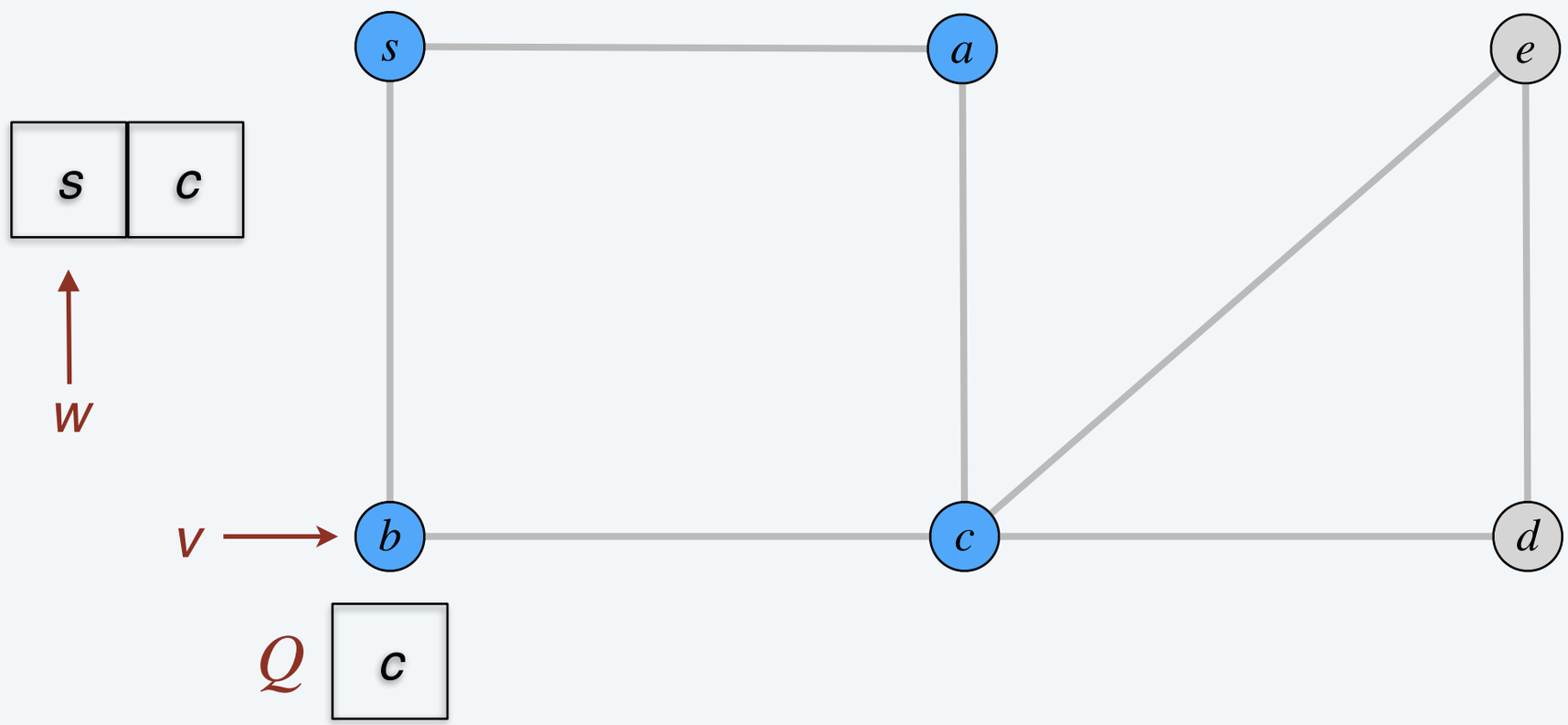
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



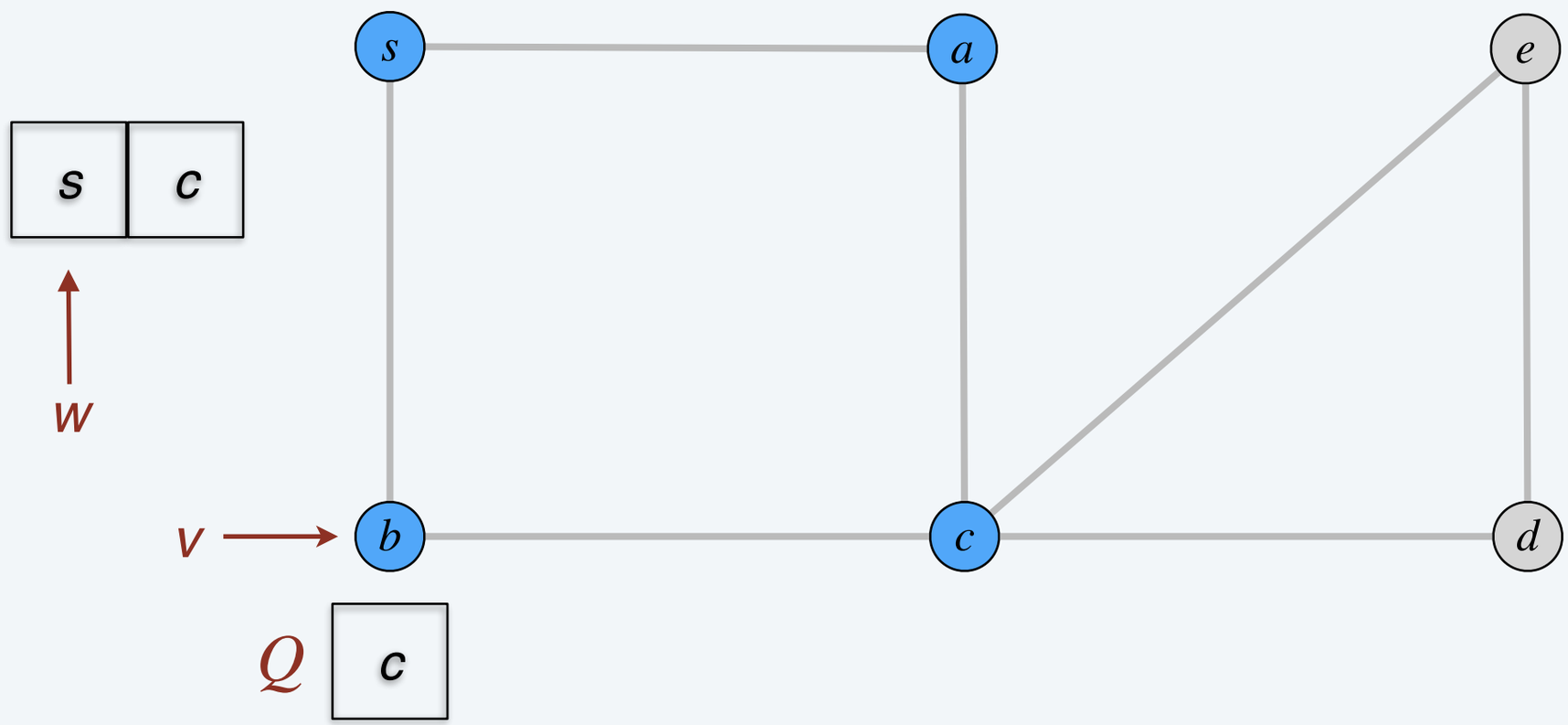
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



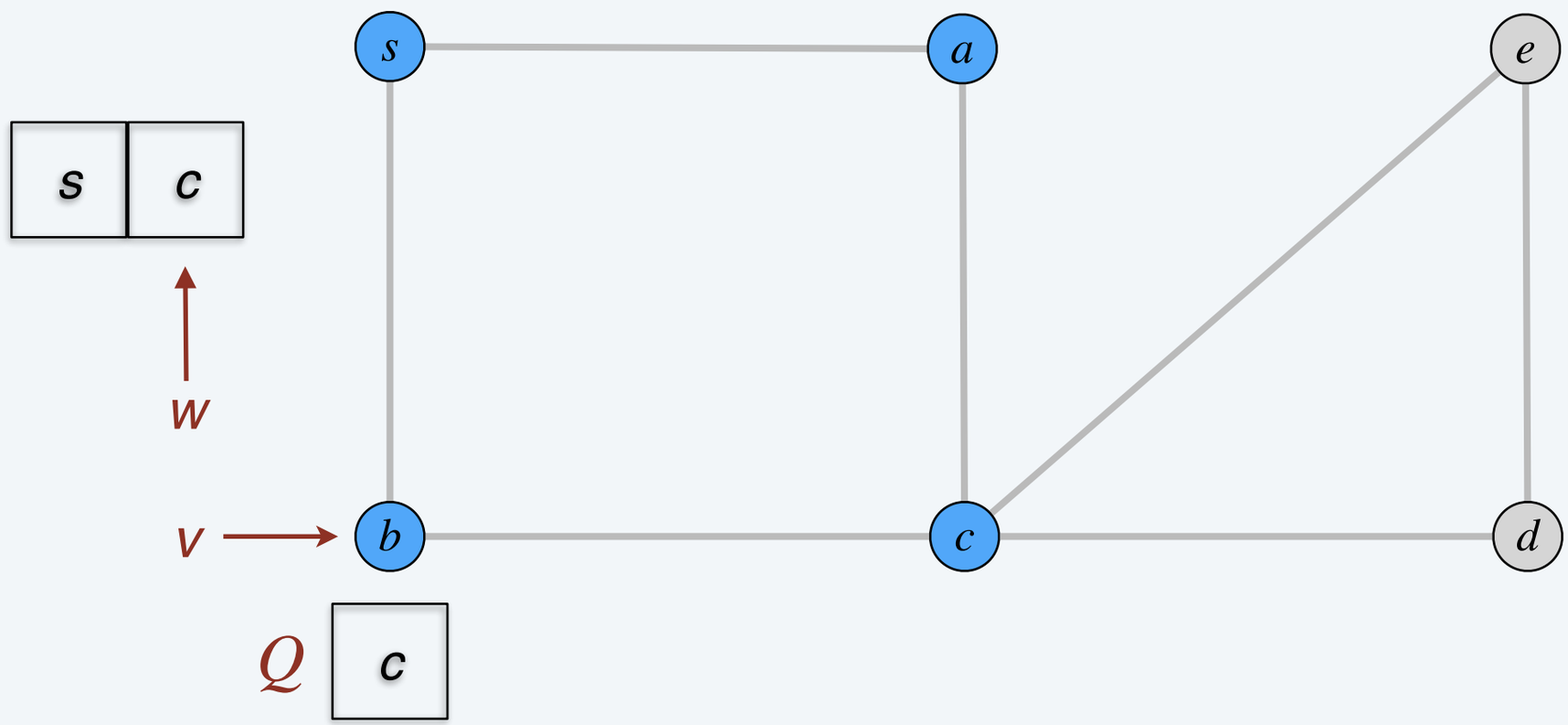
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



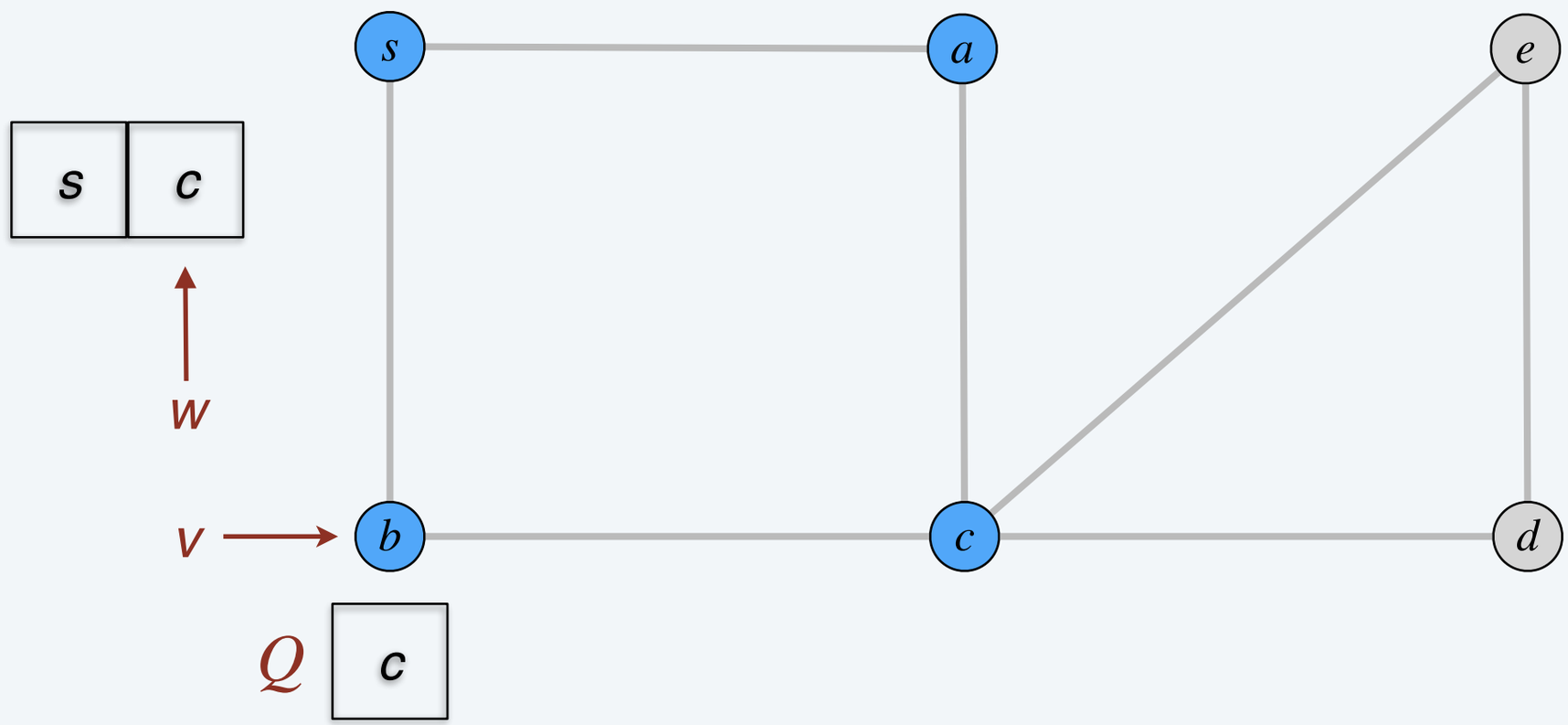
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



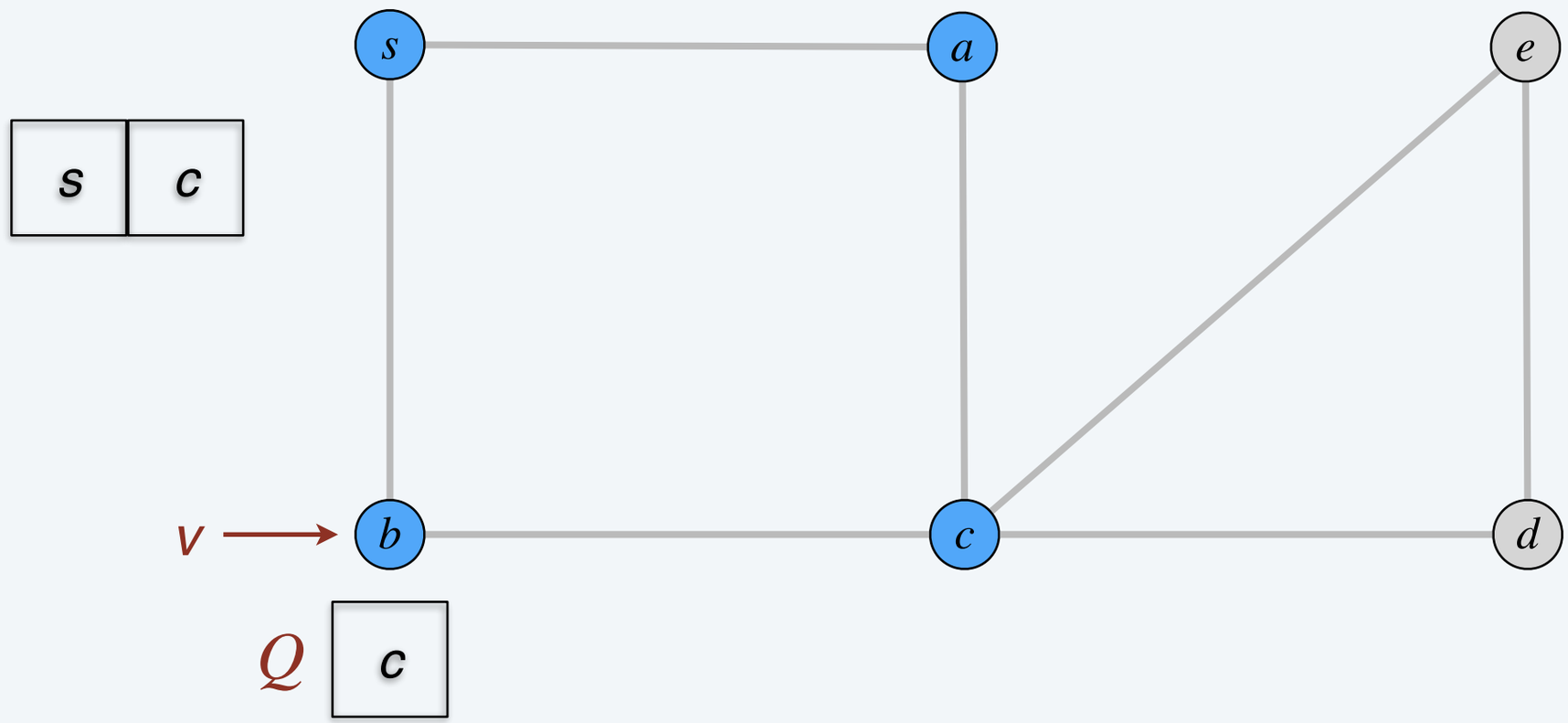
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



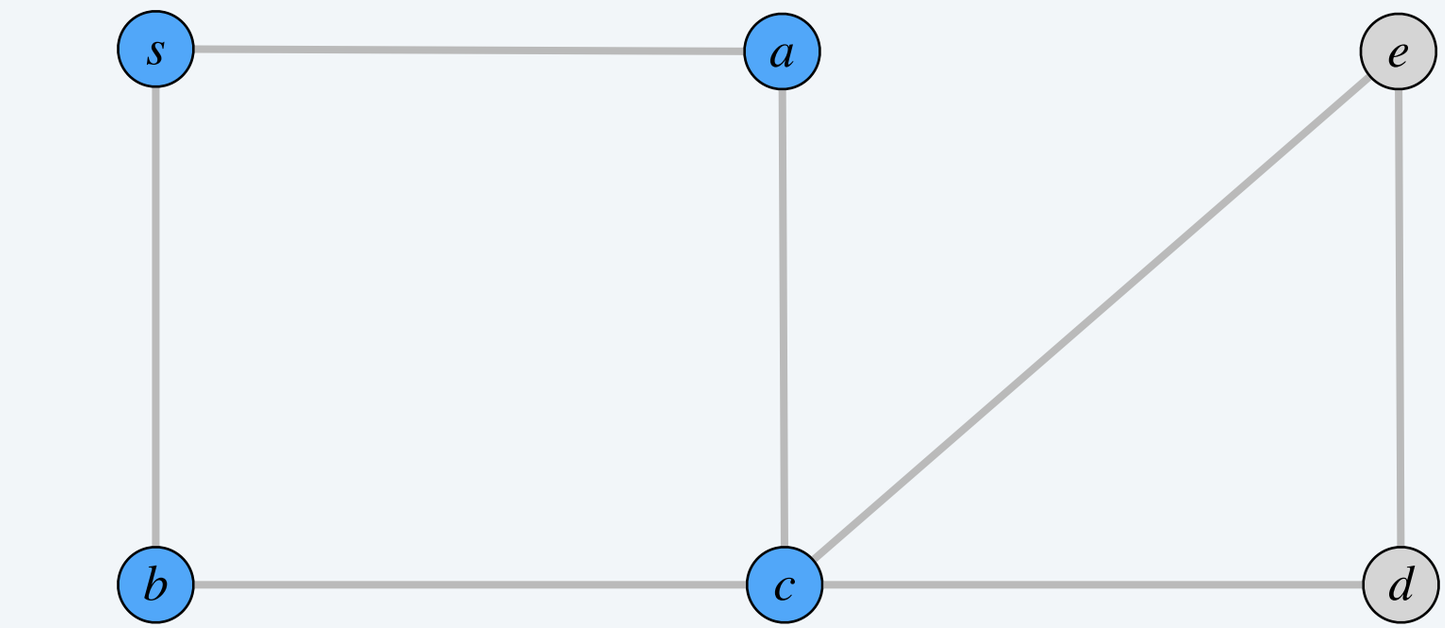
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



BFS demo

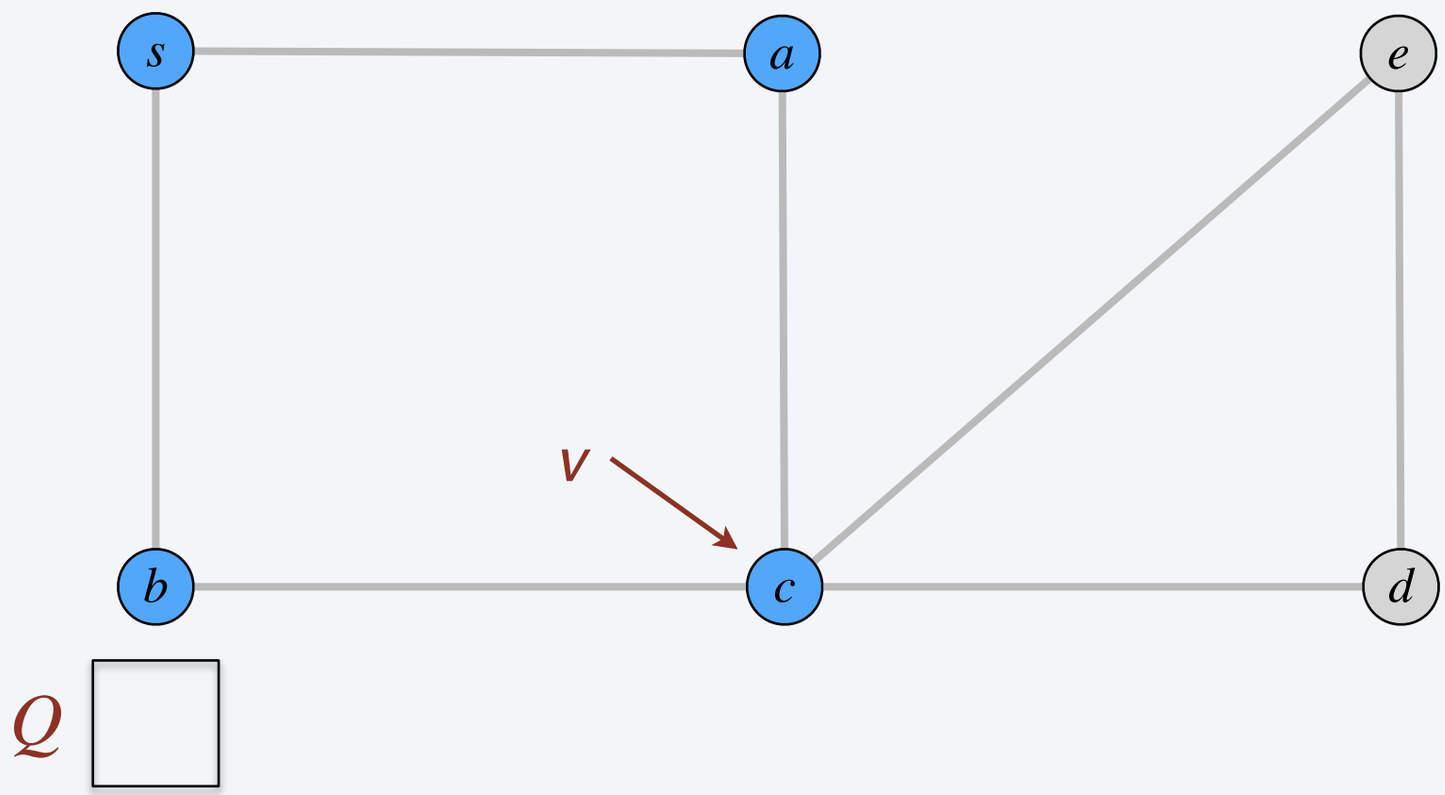
mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



Q | c |

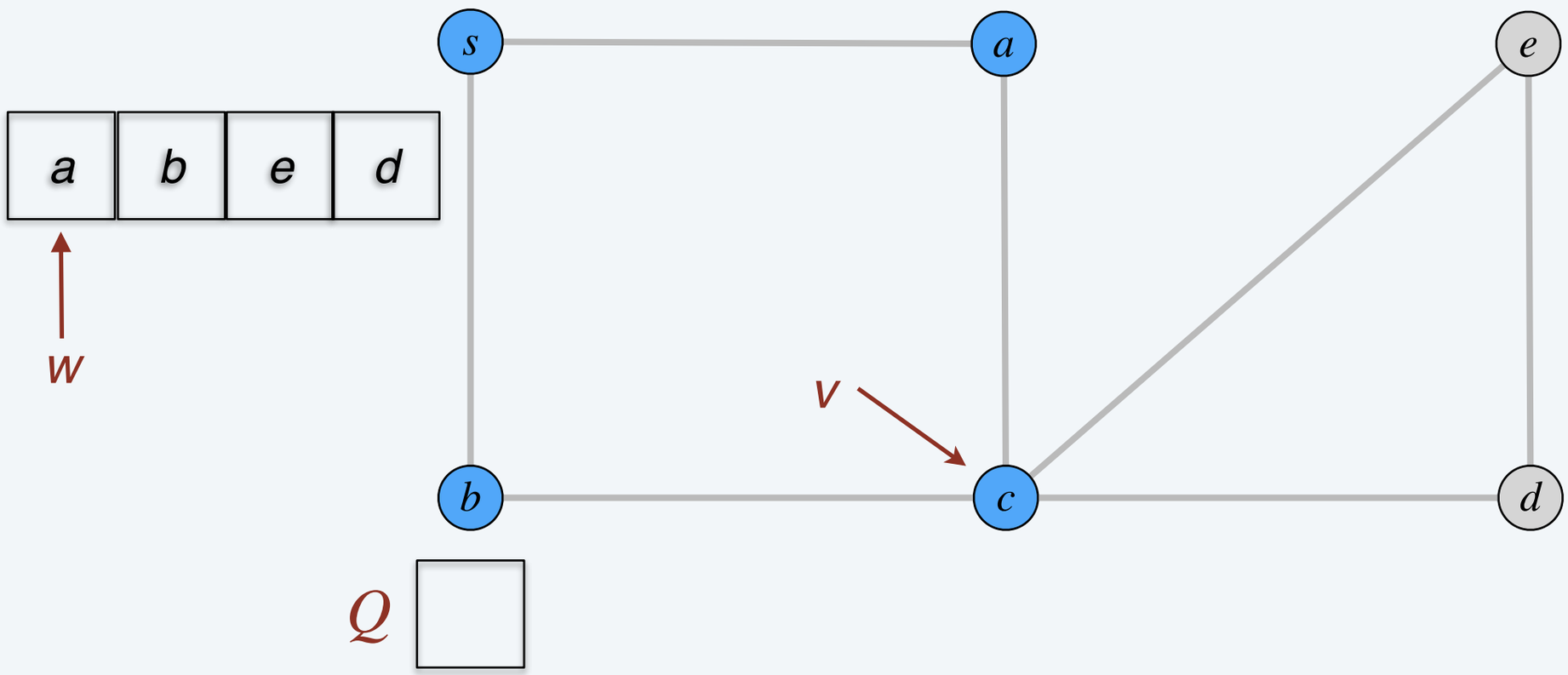
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



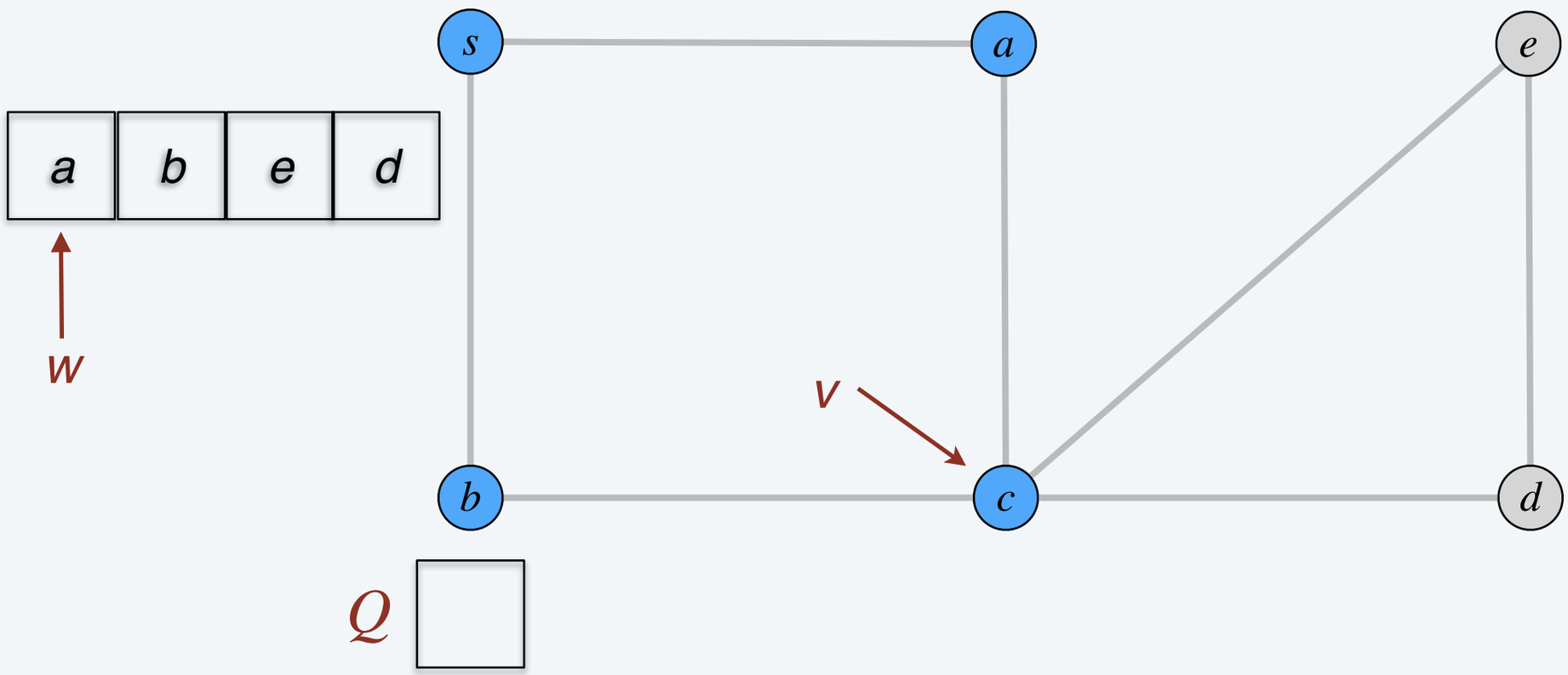
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



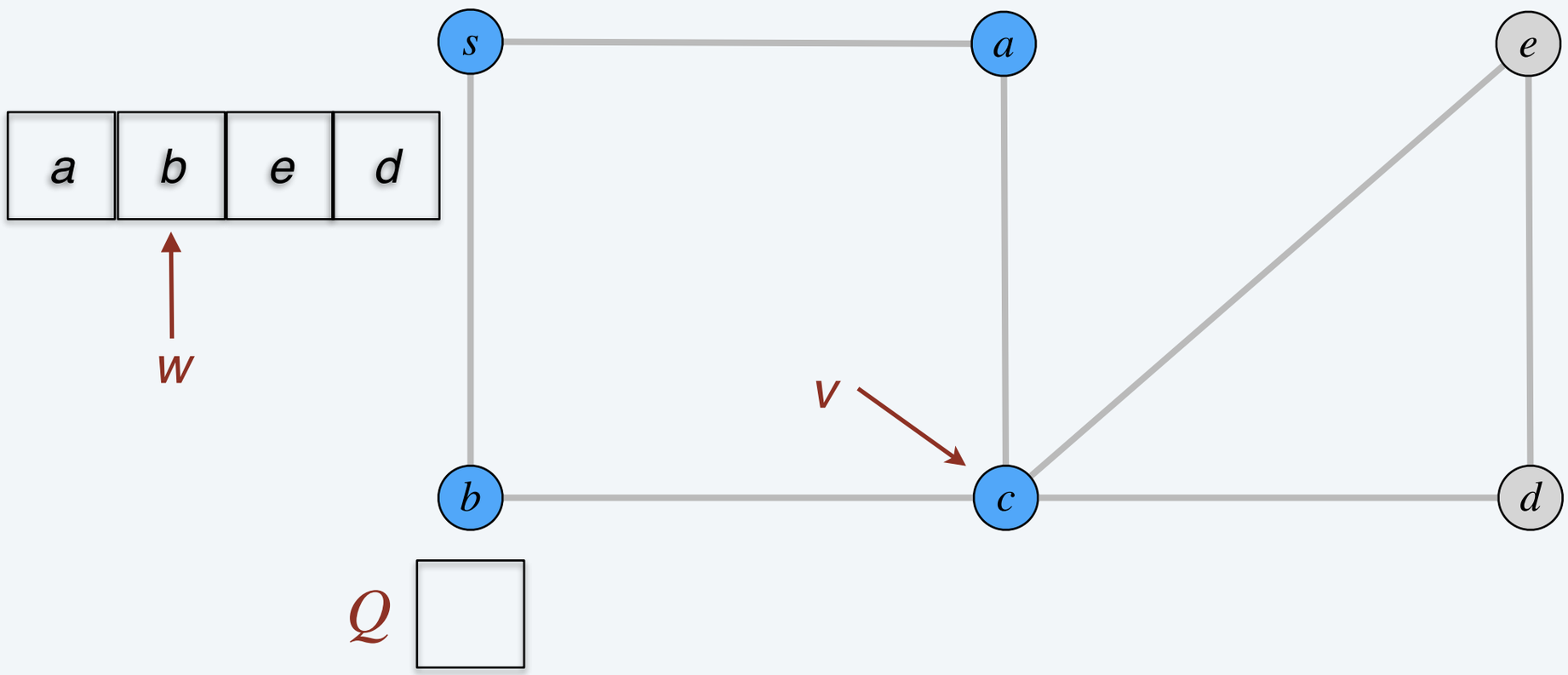
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



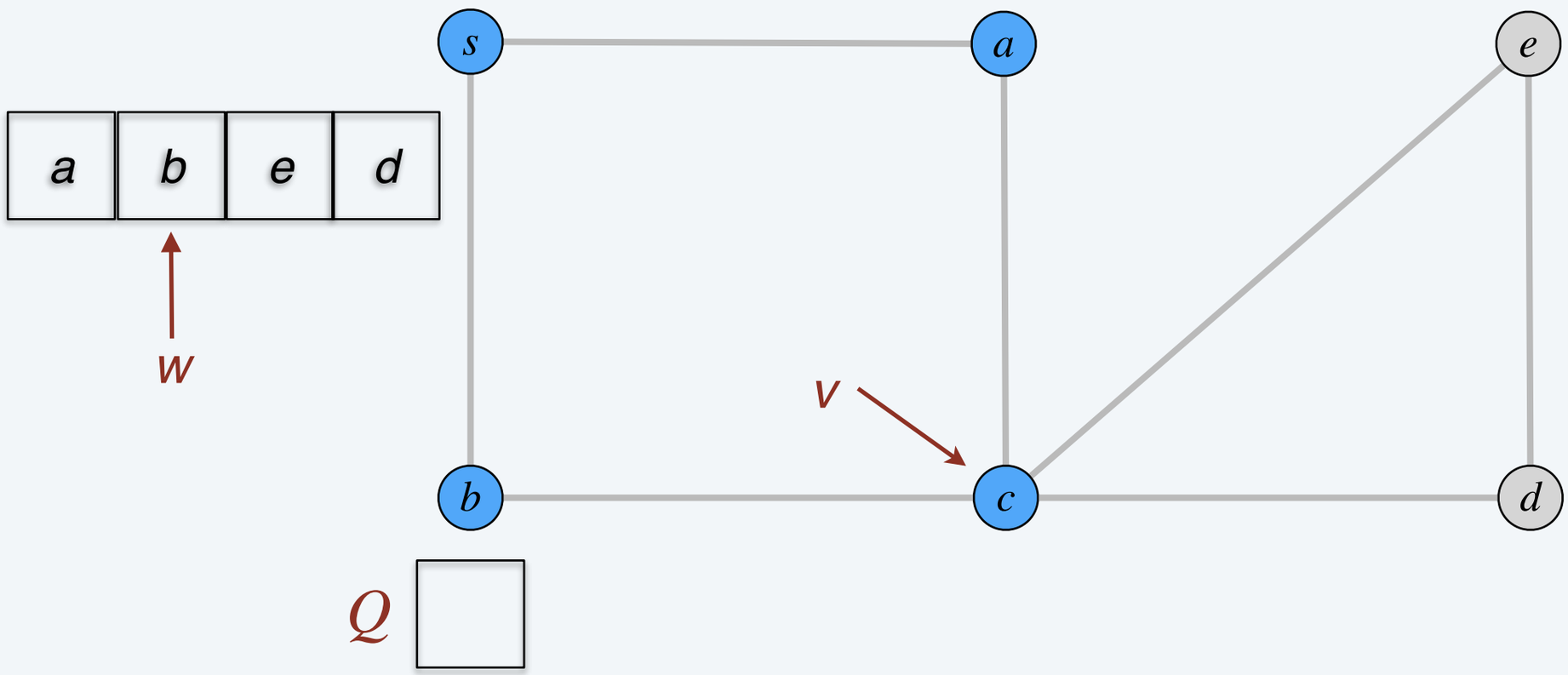
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



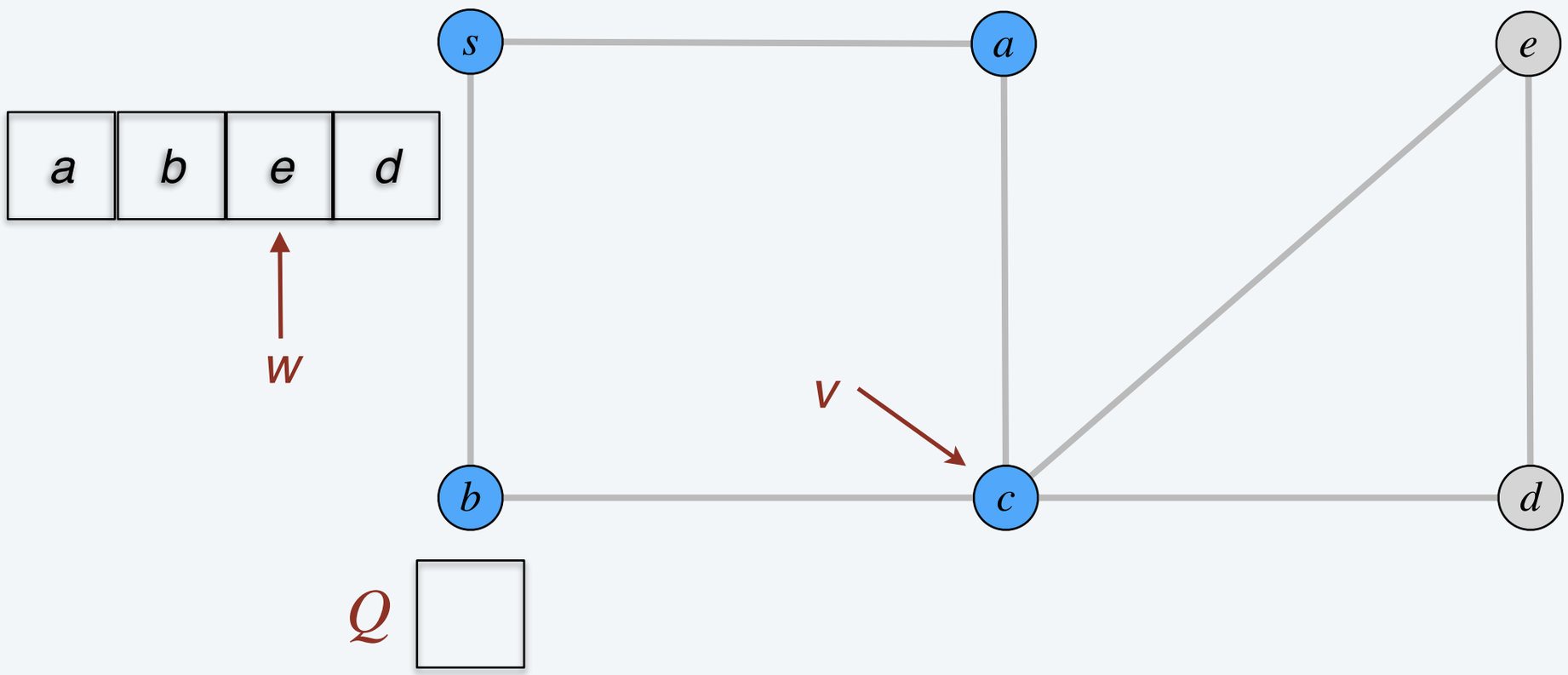
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



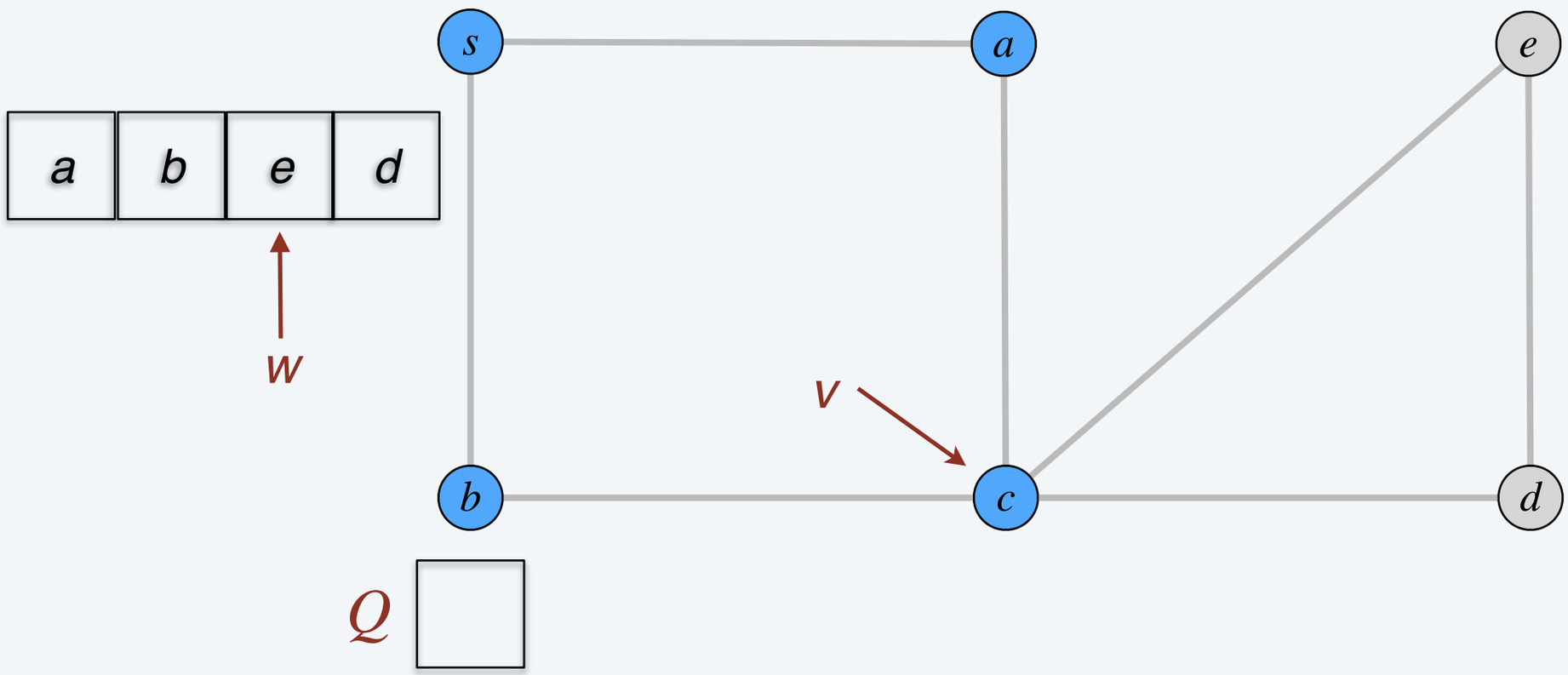
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



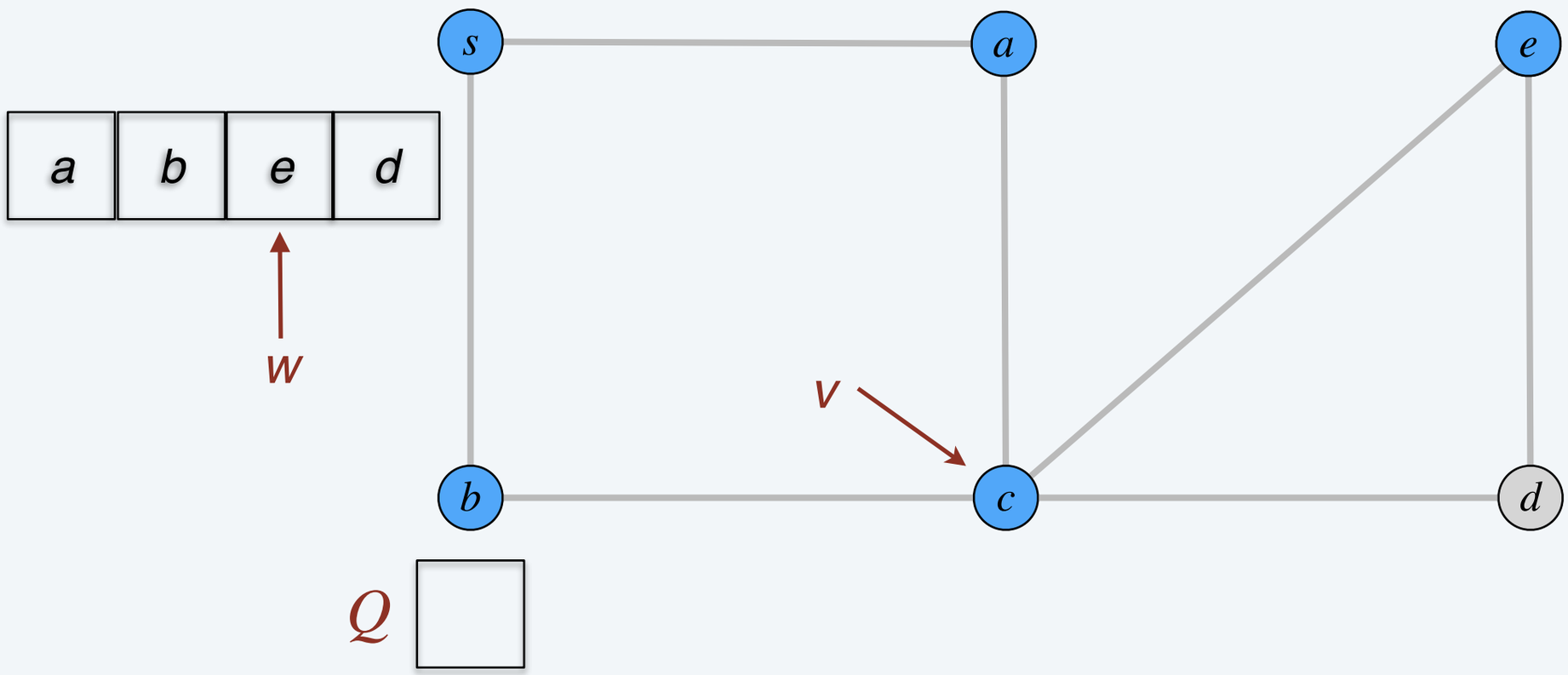
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



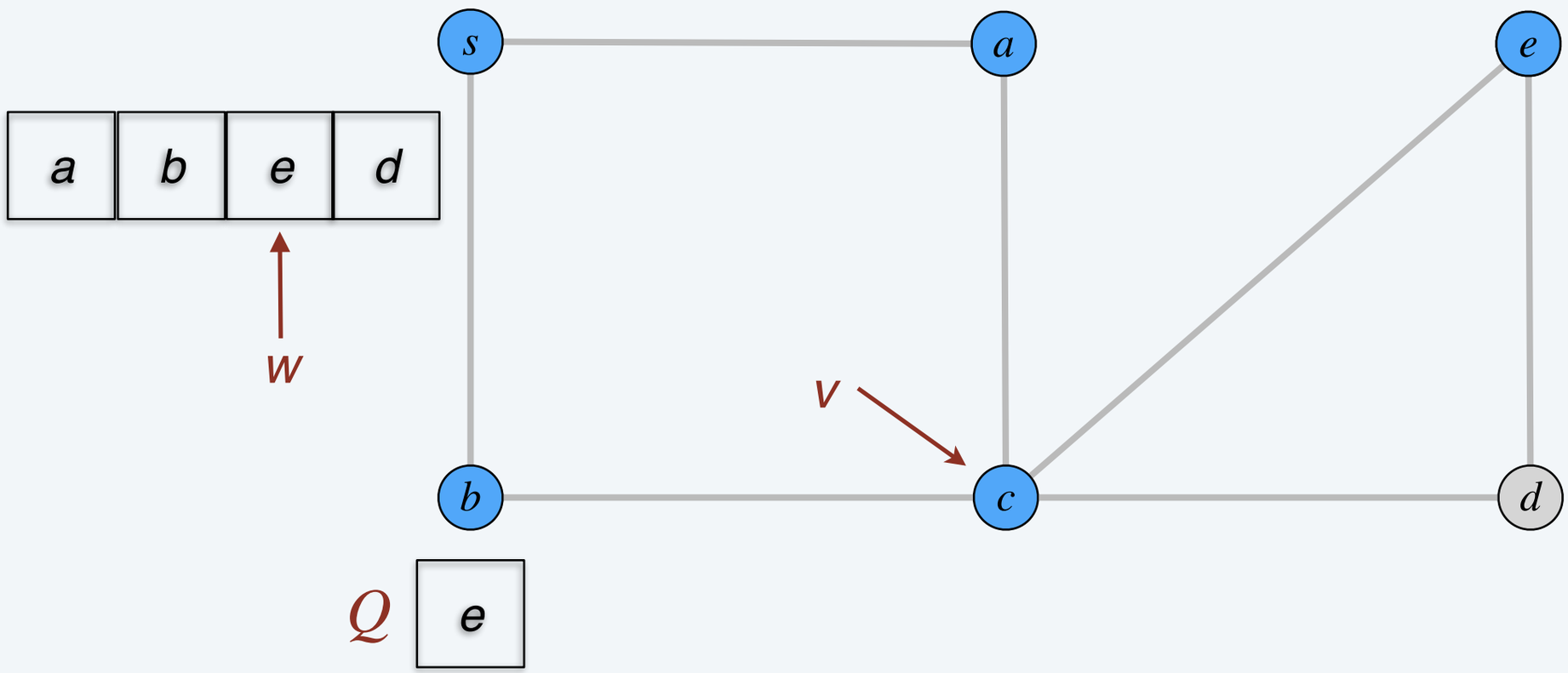
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



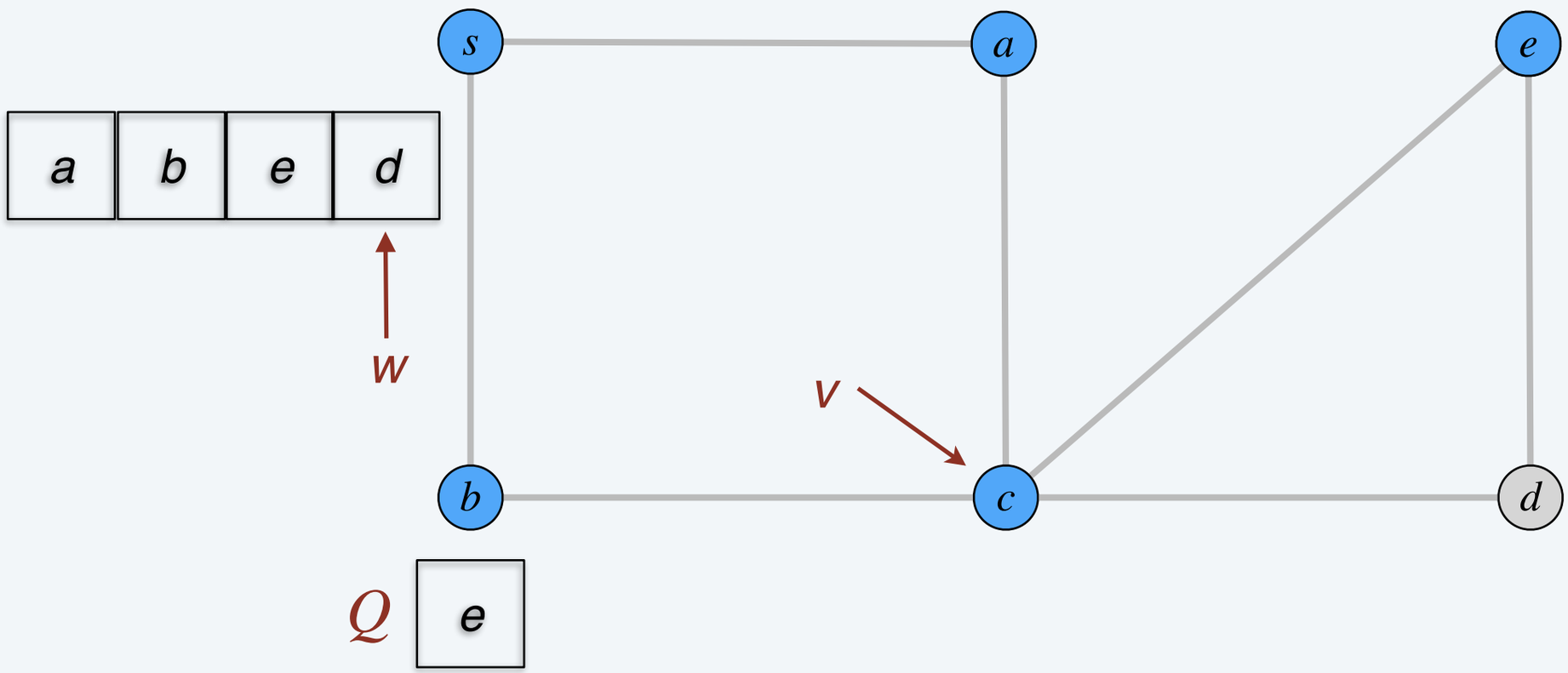
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



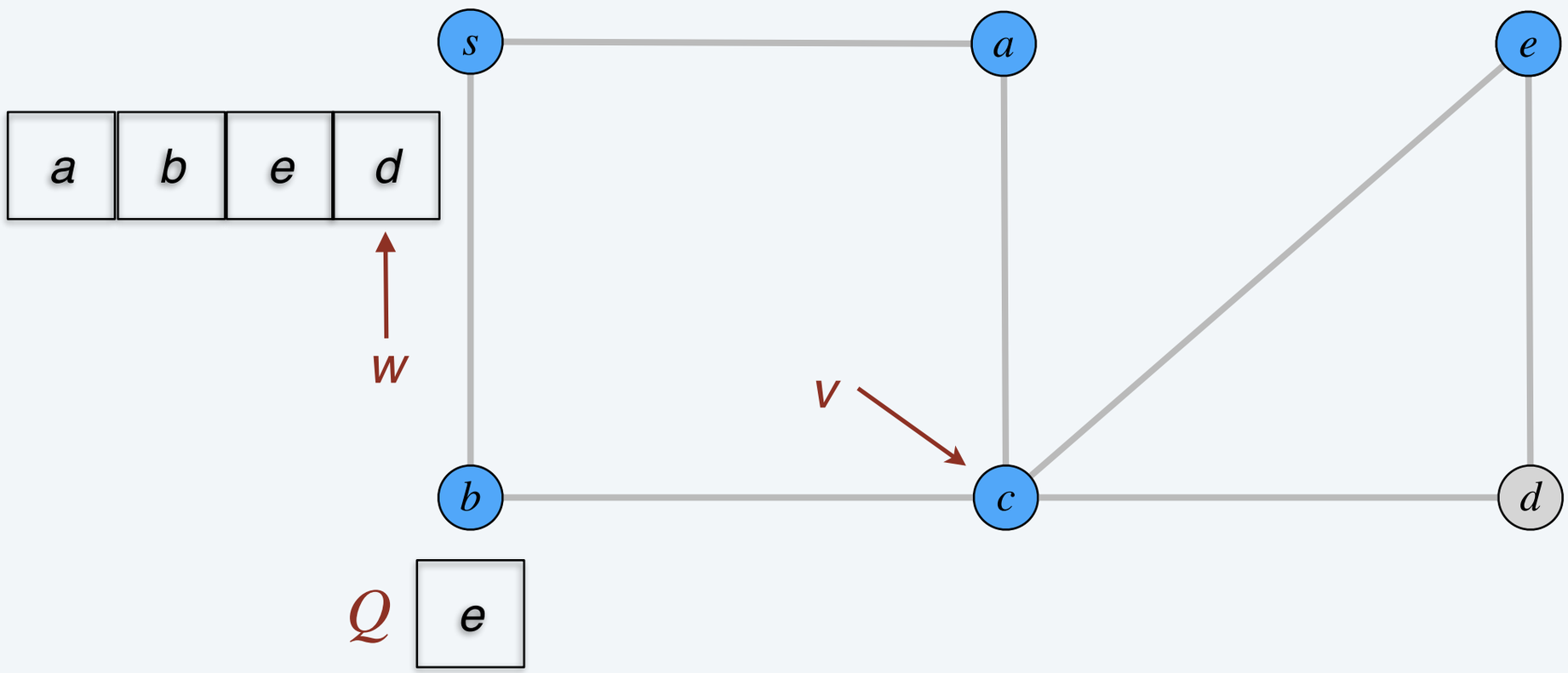
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



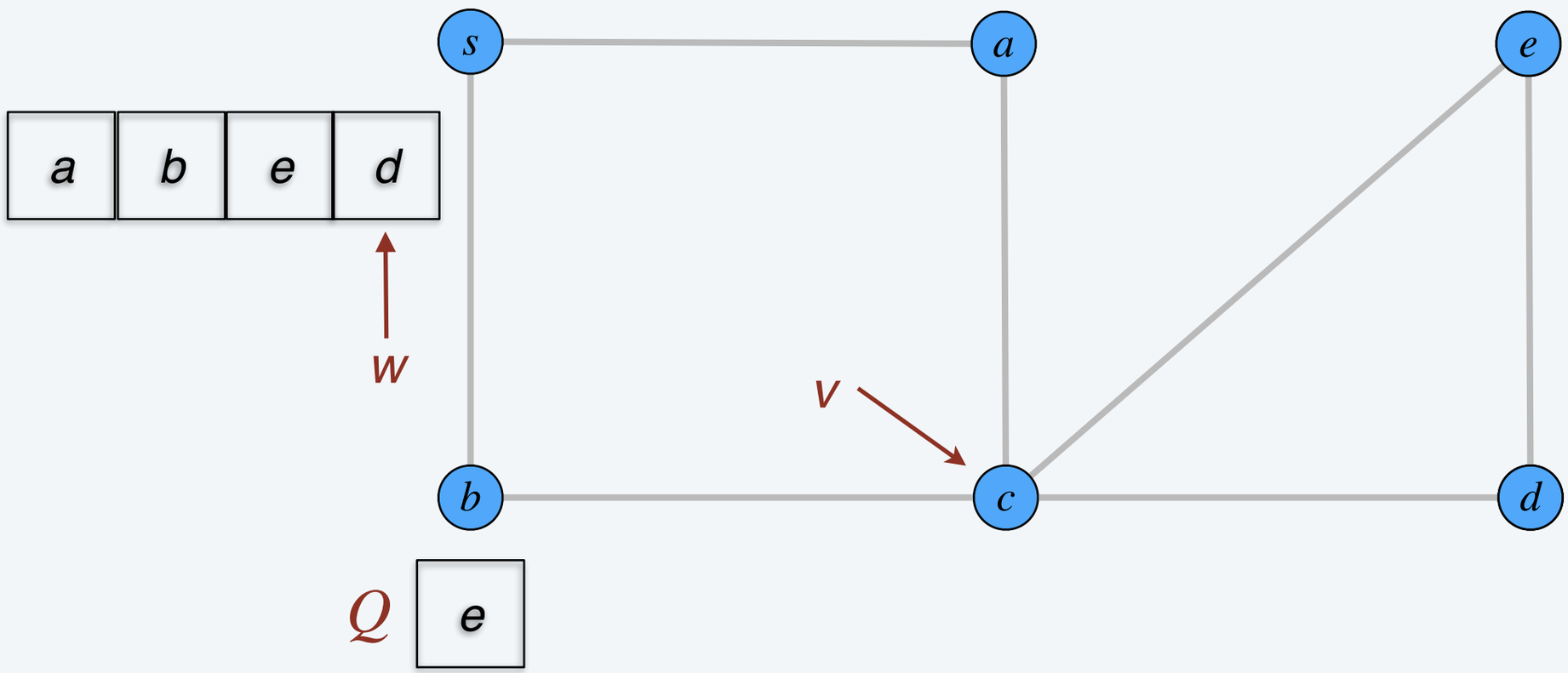
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



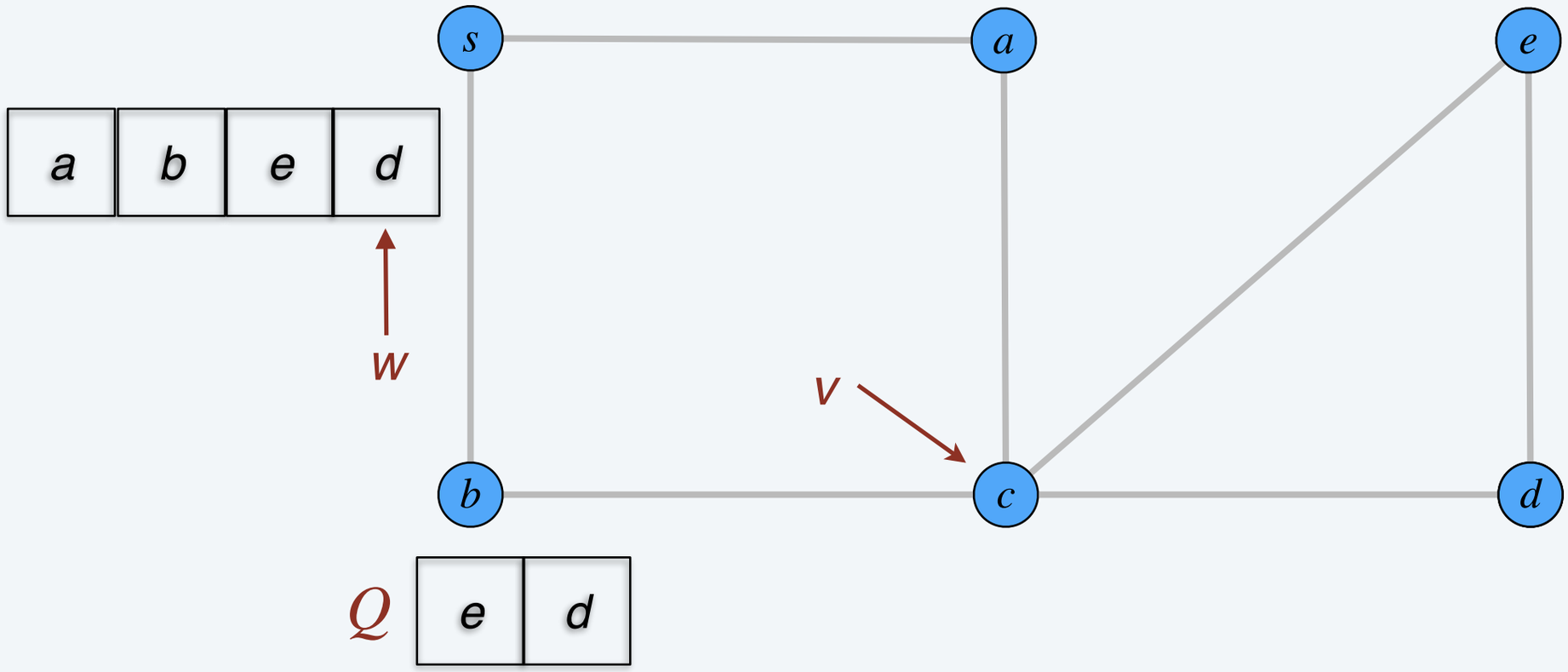
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



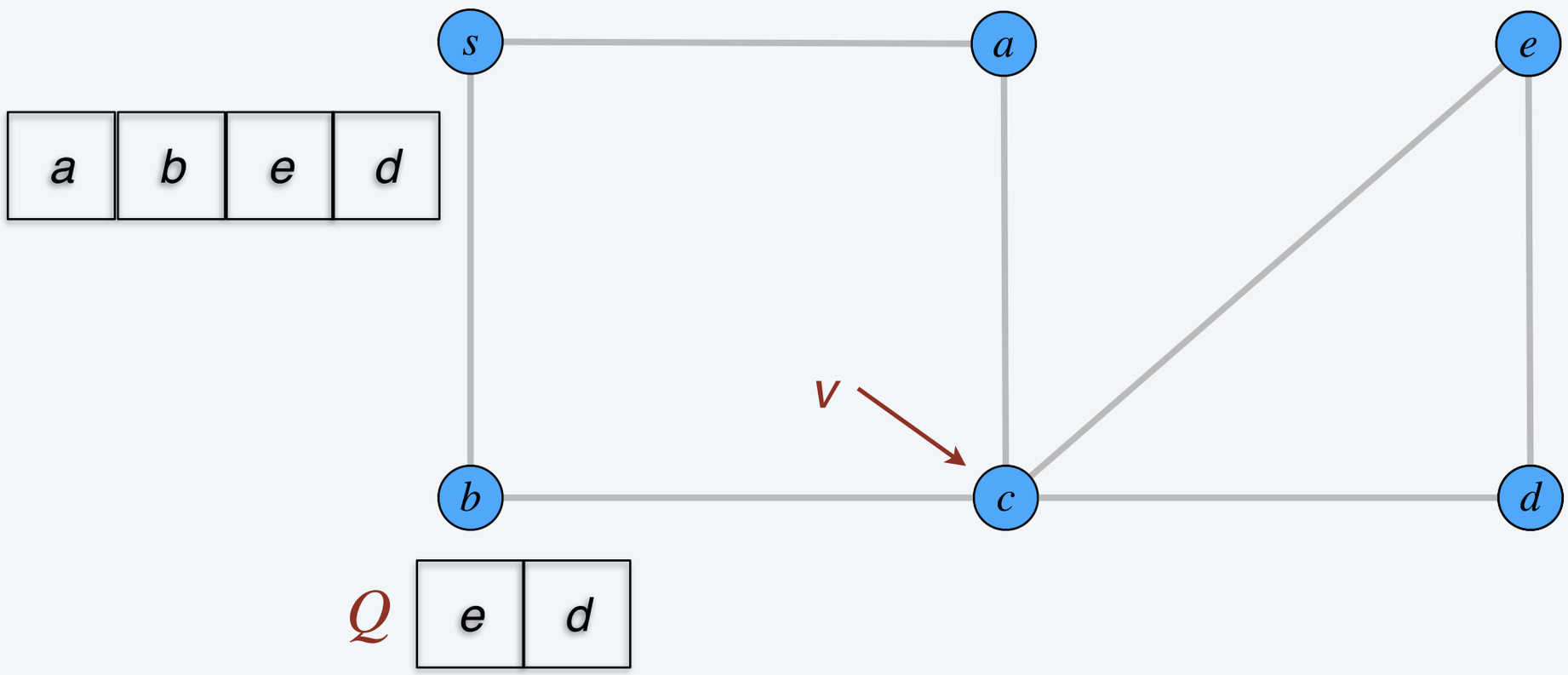
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



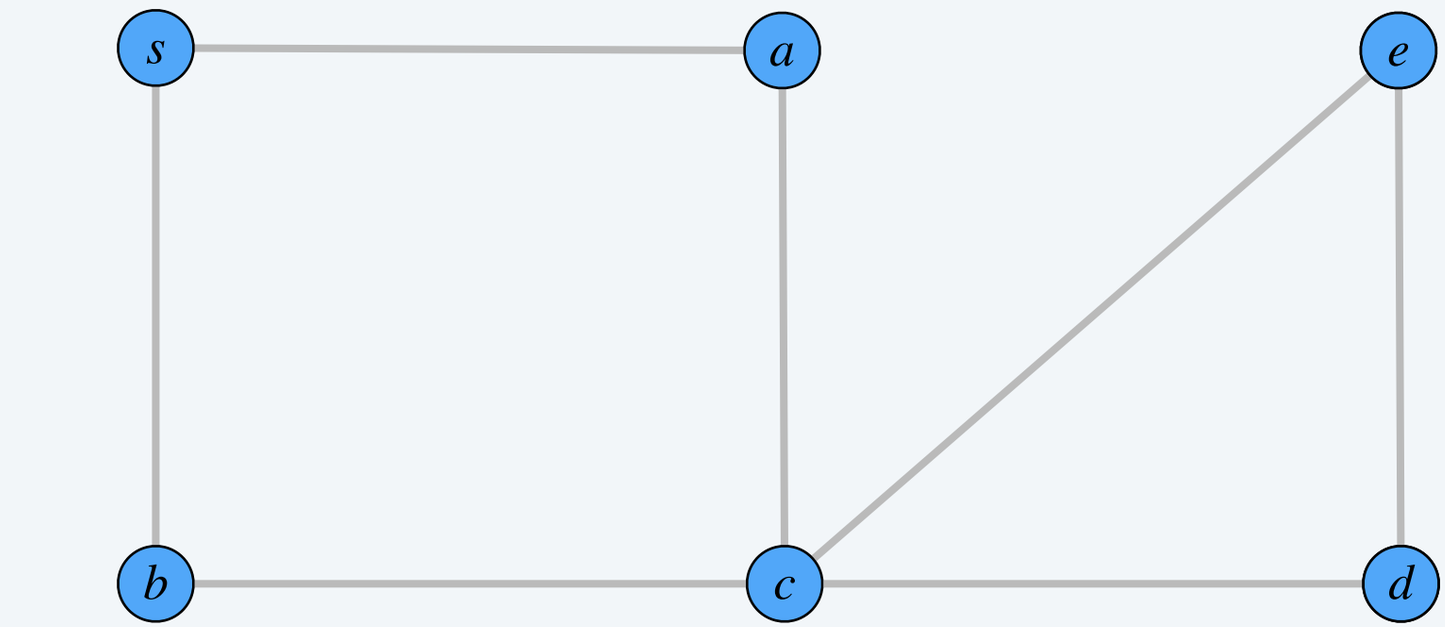
BFS demo

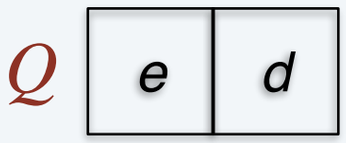
mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



BFS demo

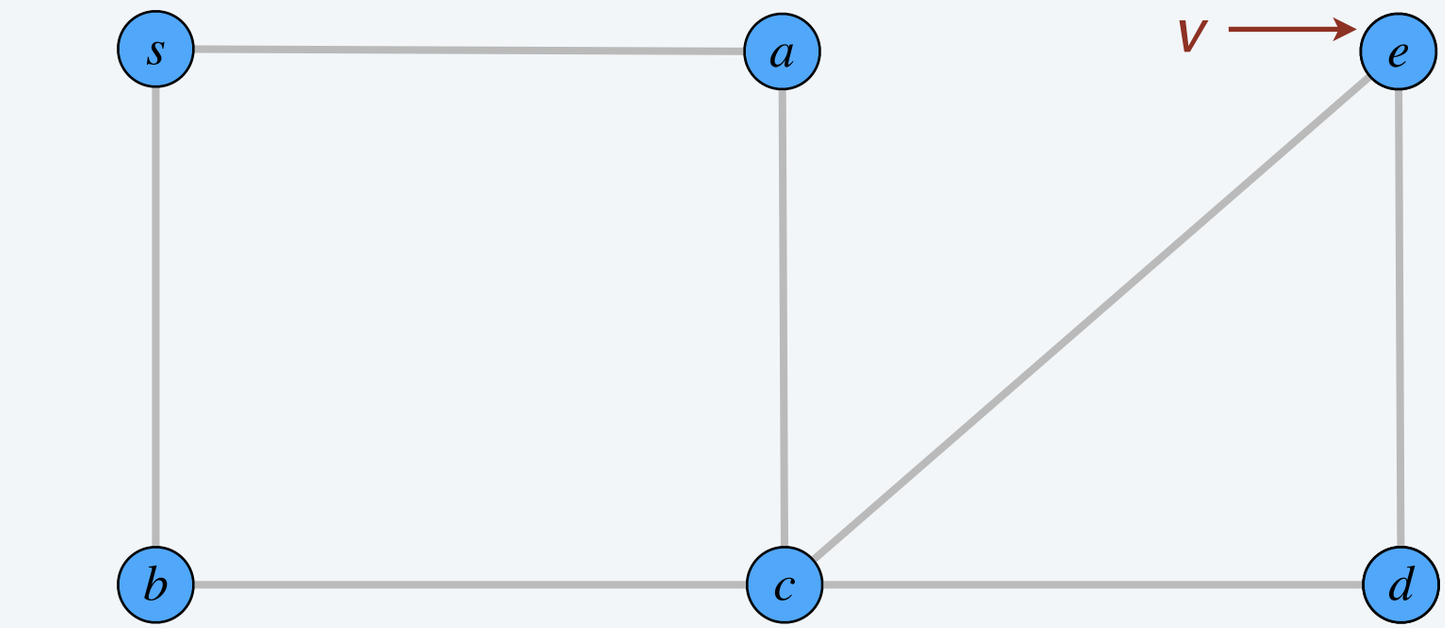
mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q





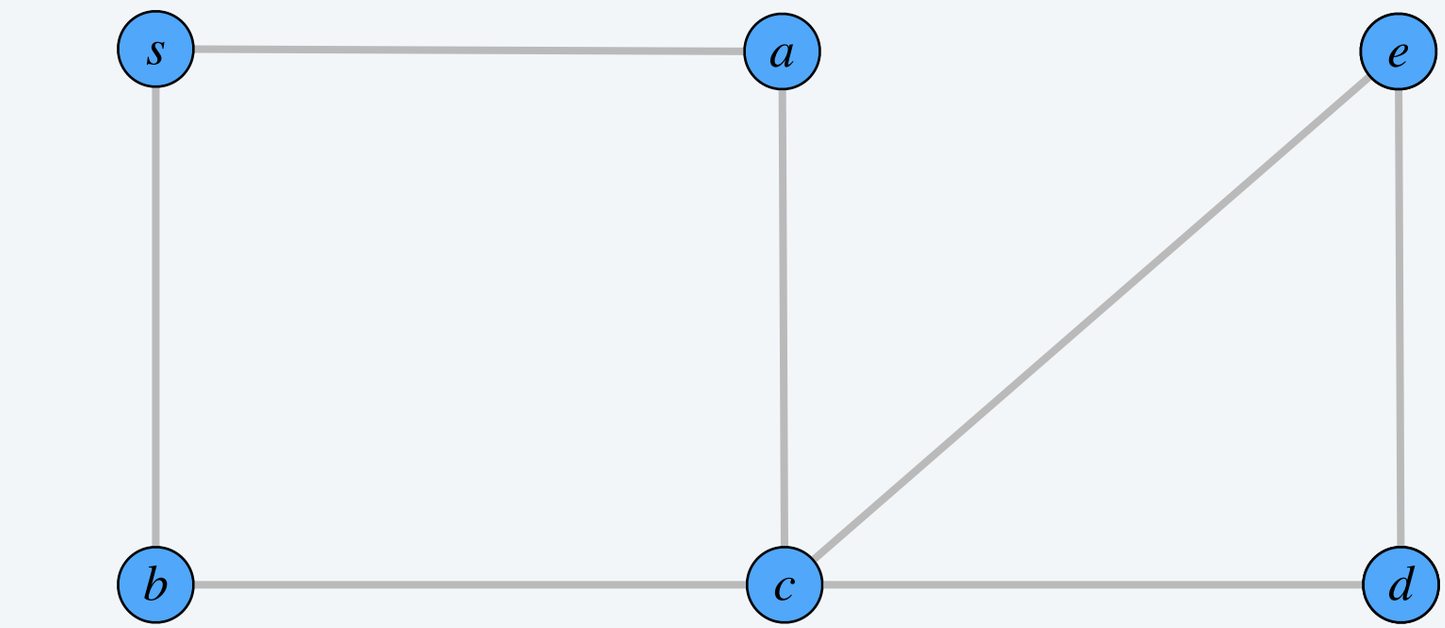
BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q



BFS demo

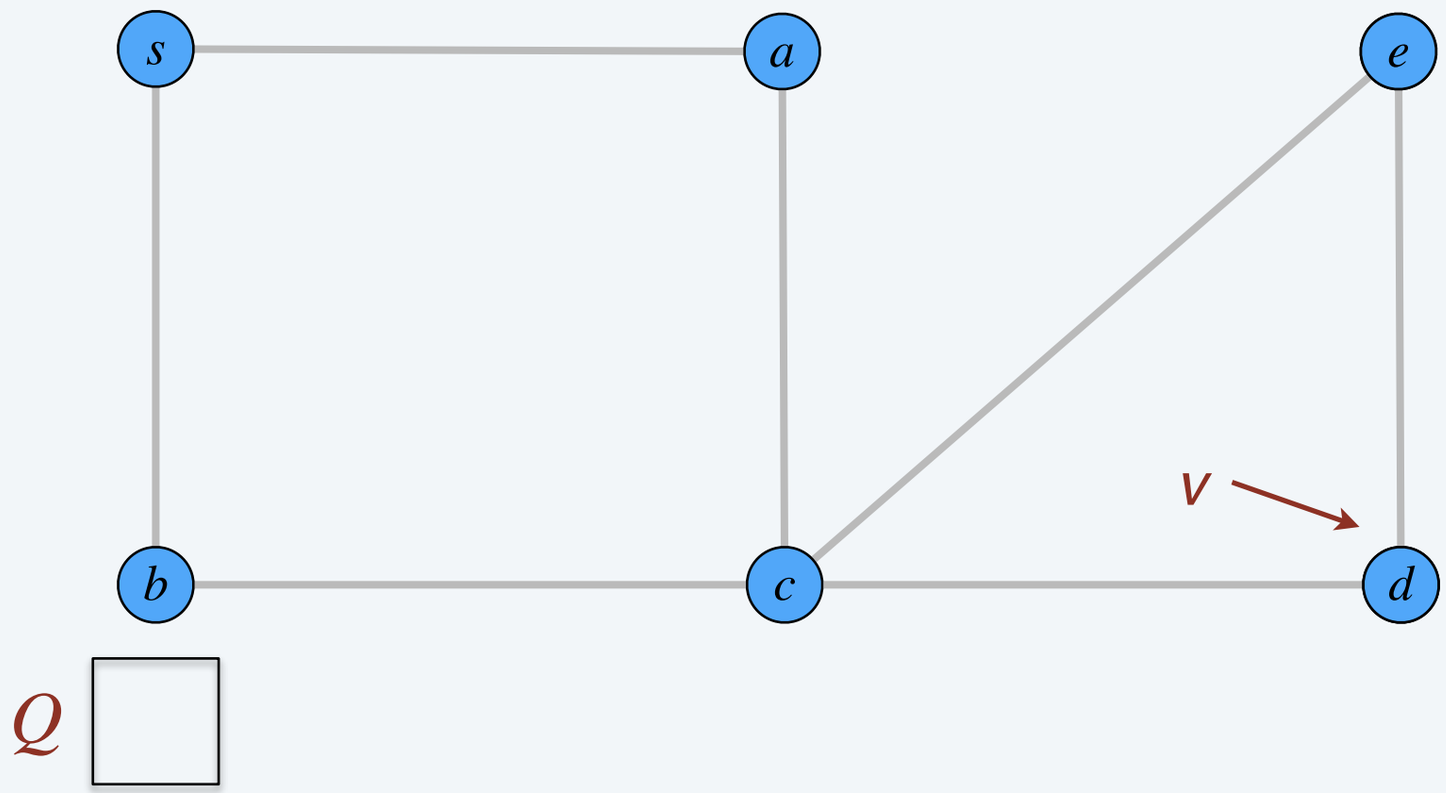
mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q

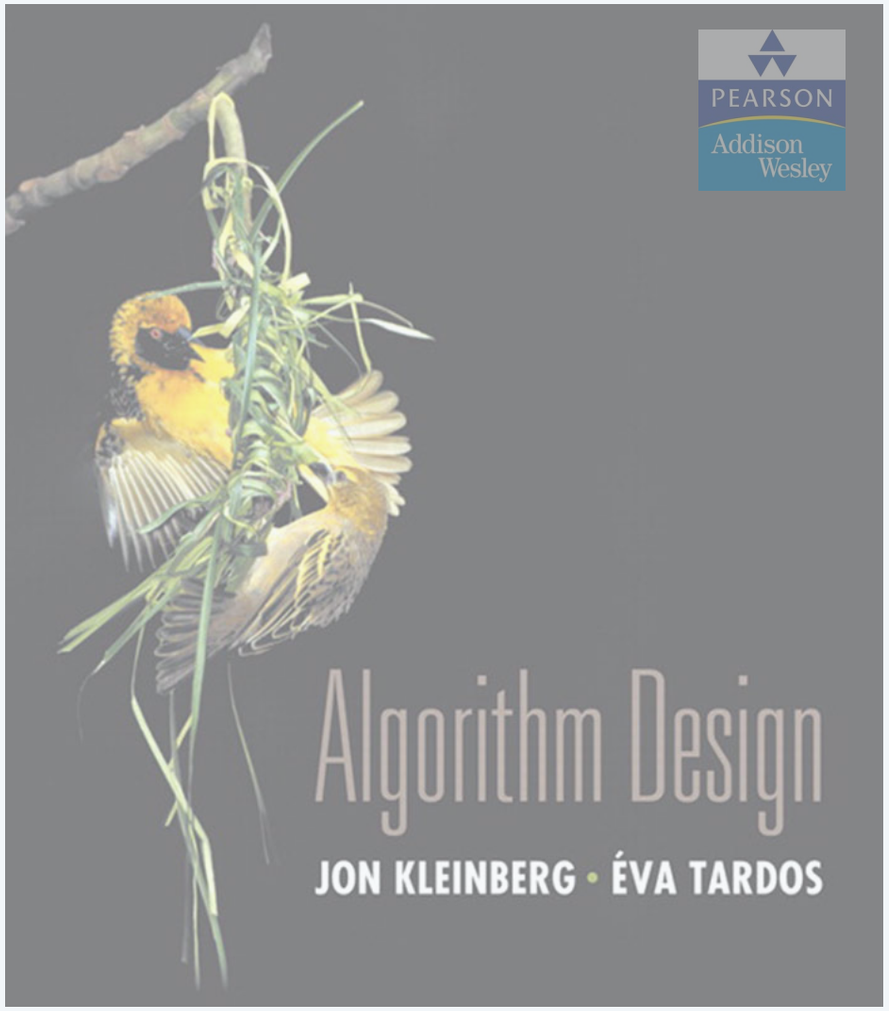


Q | d |

BFS demo

mark s as explored, all other vertices as unexplored
Q := a queue, initialized with s
while Q is not empty do
    remove the vertex from the front of Q, call it v
    for each edge (v, w) in v's adjacency list do
        if w is unexplored then
            mark w as explored
            add w to the end of Q





3. GRAPHS

- *BFS demo*
- *DFS demo*

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

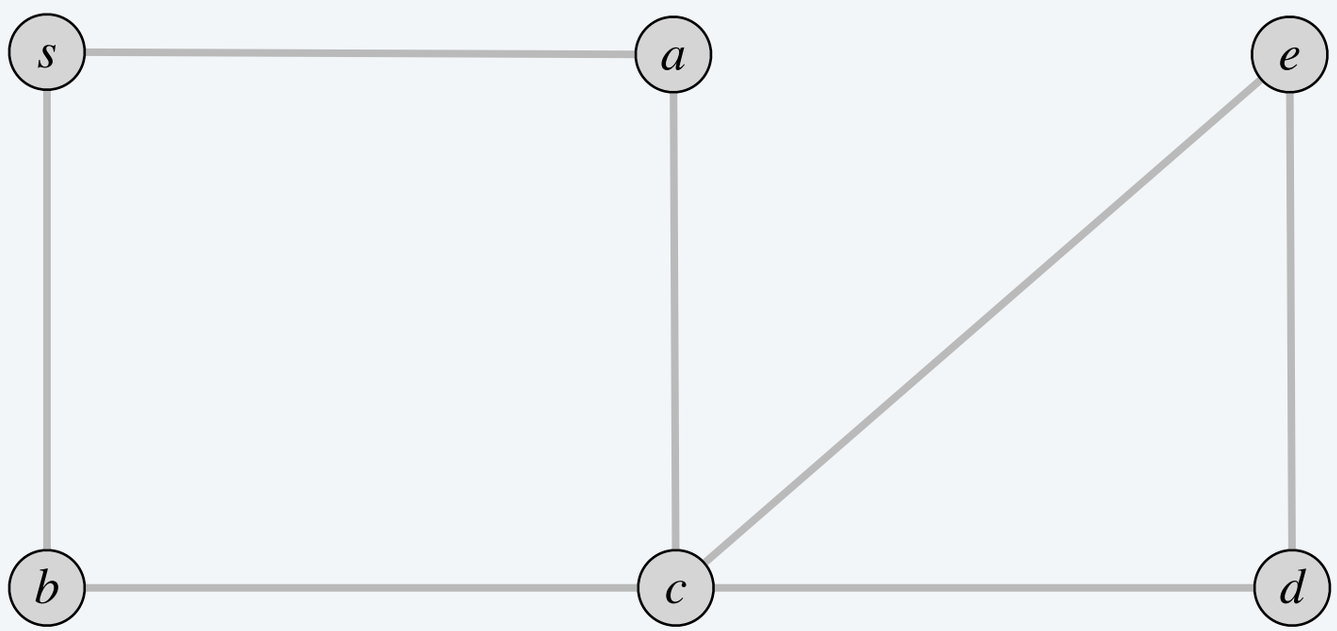
    remove ("pop") the vertex v from the front of S

    IF v is unexplored

        mark v as explored

        FOR each edge (v, w) in v's adjacency-list

            add ("push") w to the front of S



DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

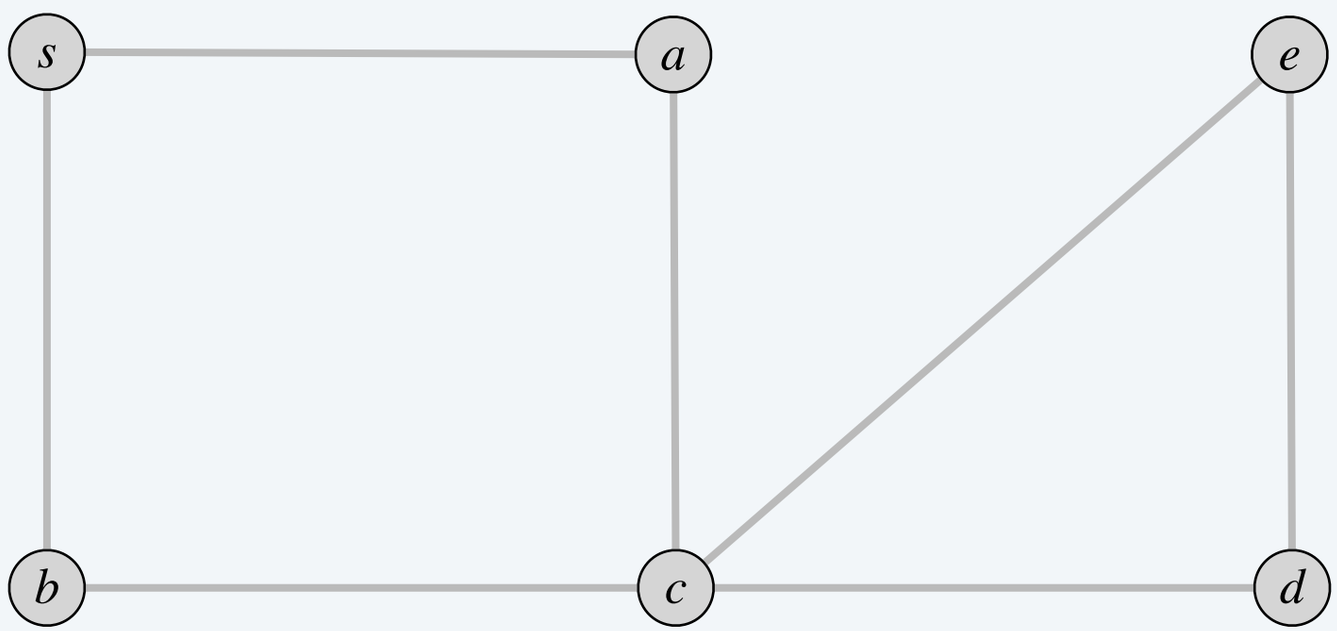
  remove ("pop") the vertex v from the front of S

  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S



DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

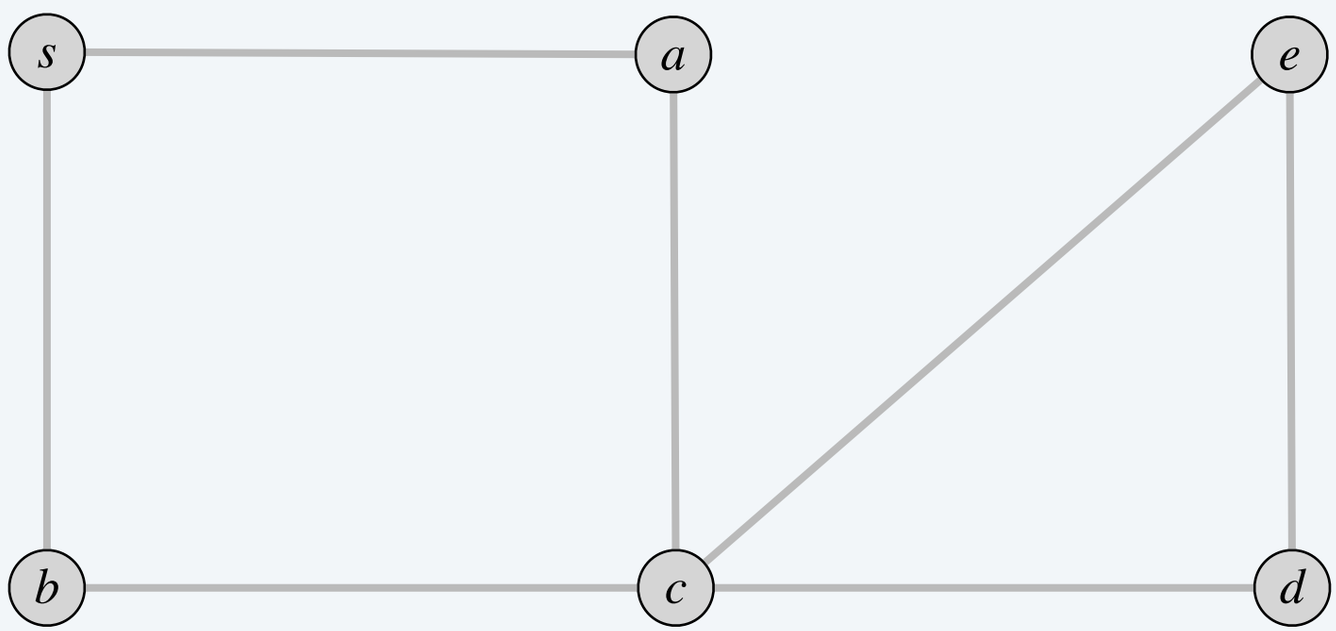
  remove ("pop") the vertex v from the front of S

  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S



S

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

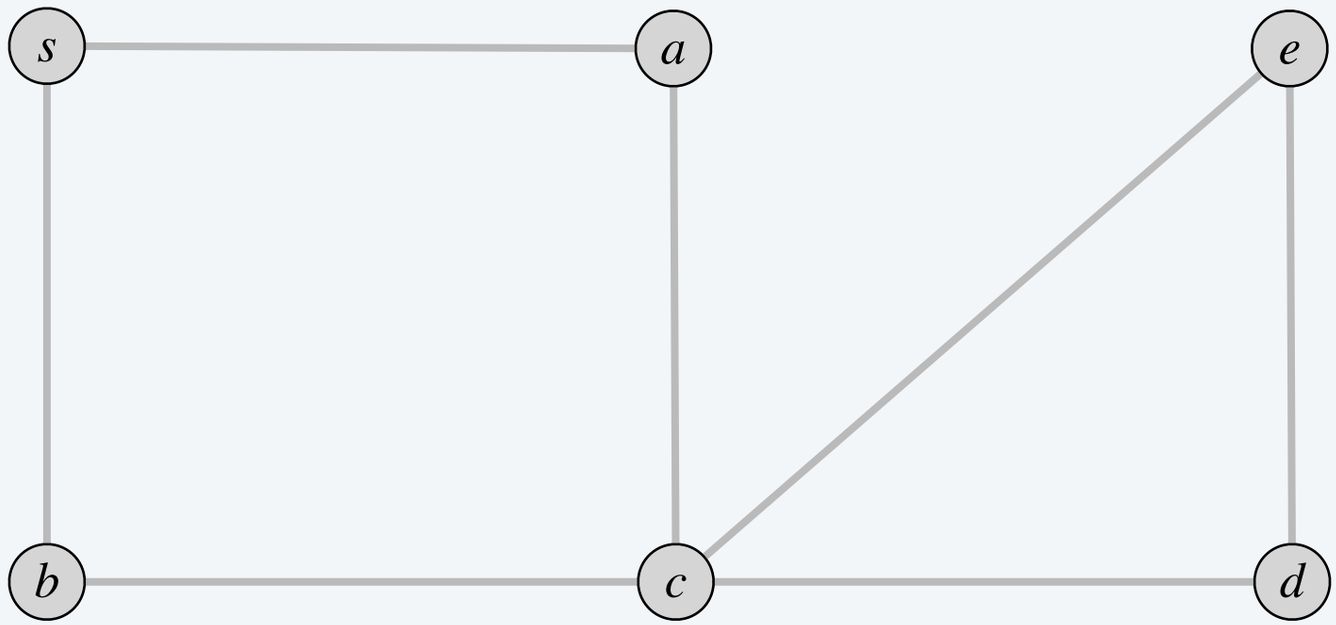
    remove ("pop") the vertex v from the front of S

    IF v is unexplored

        mark v as explored

        FOR each edge (v, w) in v's adjacency-list

            add ("push") w to the front of S



S

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

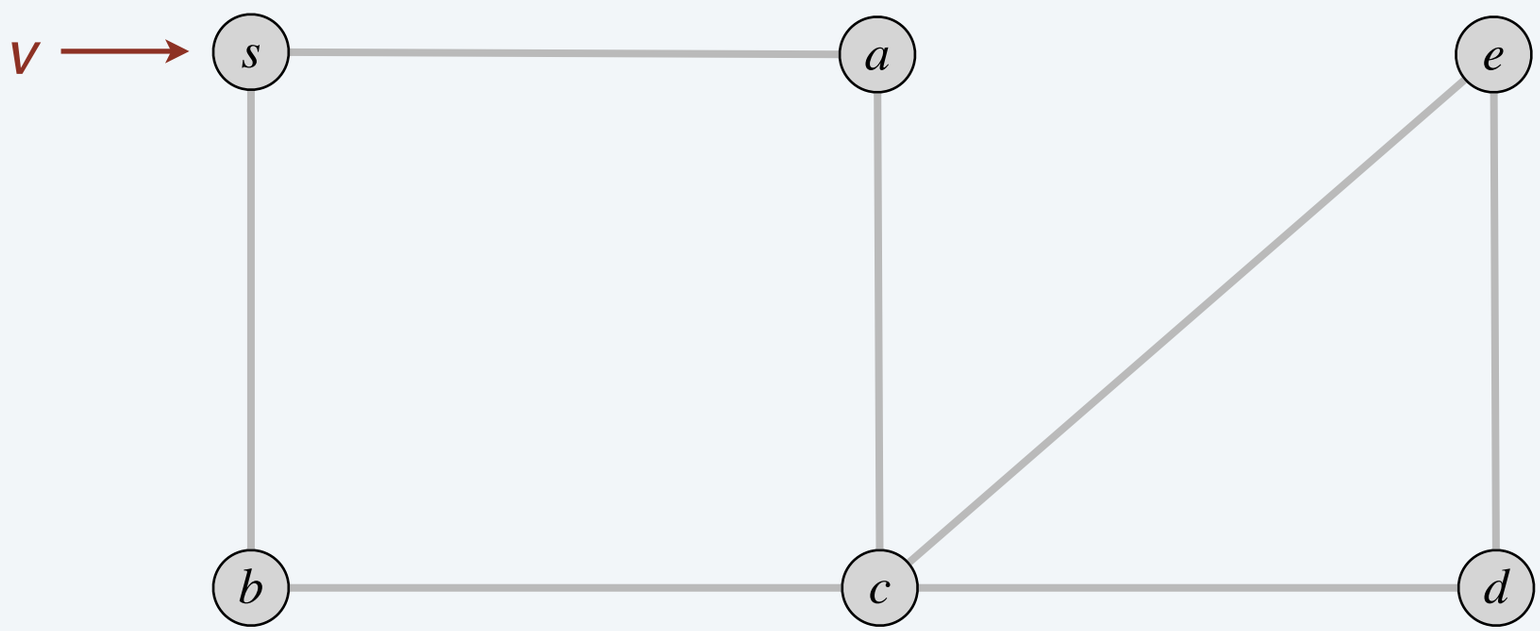
    remove ("pop") the vertex v from the front of S

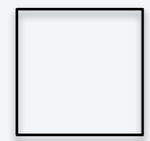
    IF v is unexplored

        mark v as explored

        FOR each edge (v, w) in v's adjacency-list

            add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

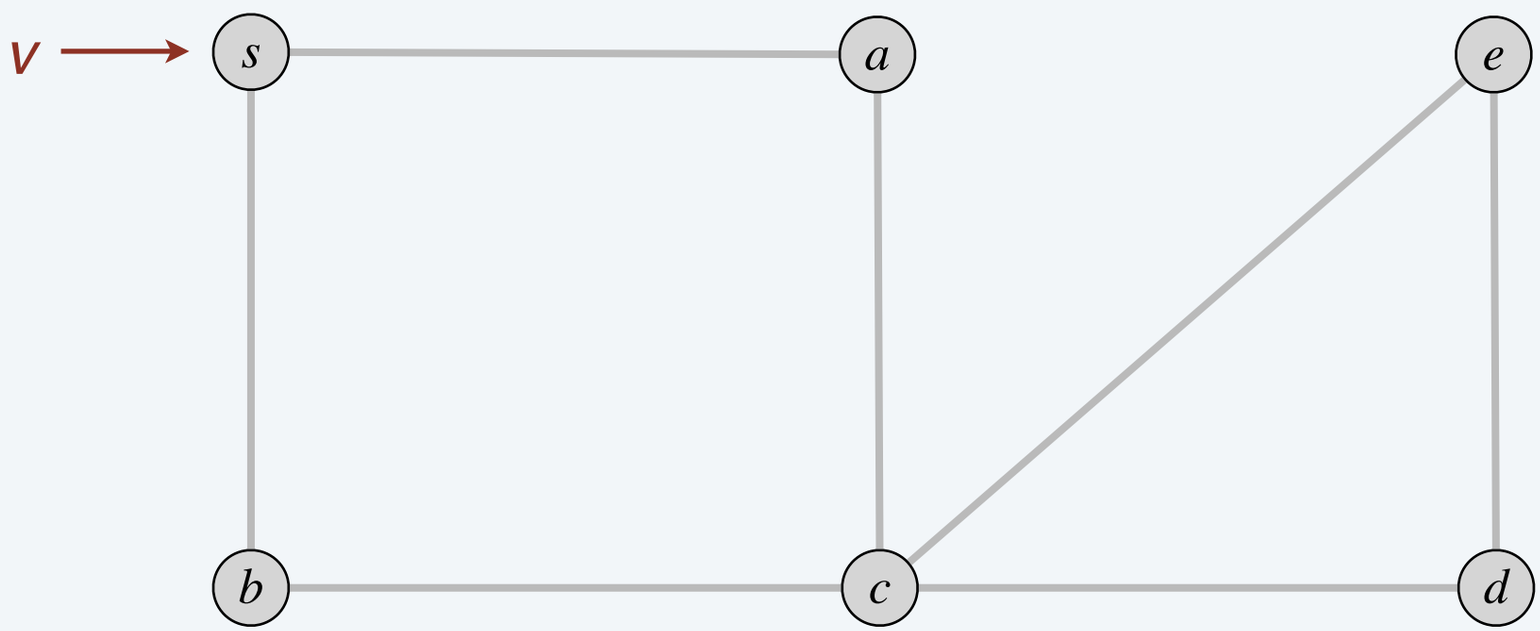
  remove ("pop") the vertex v from the front of S

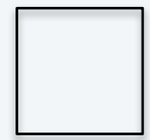
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

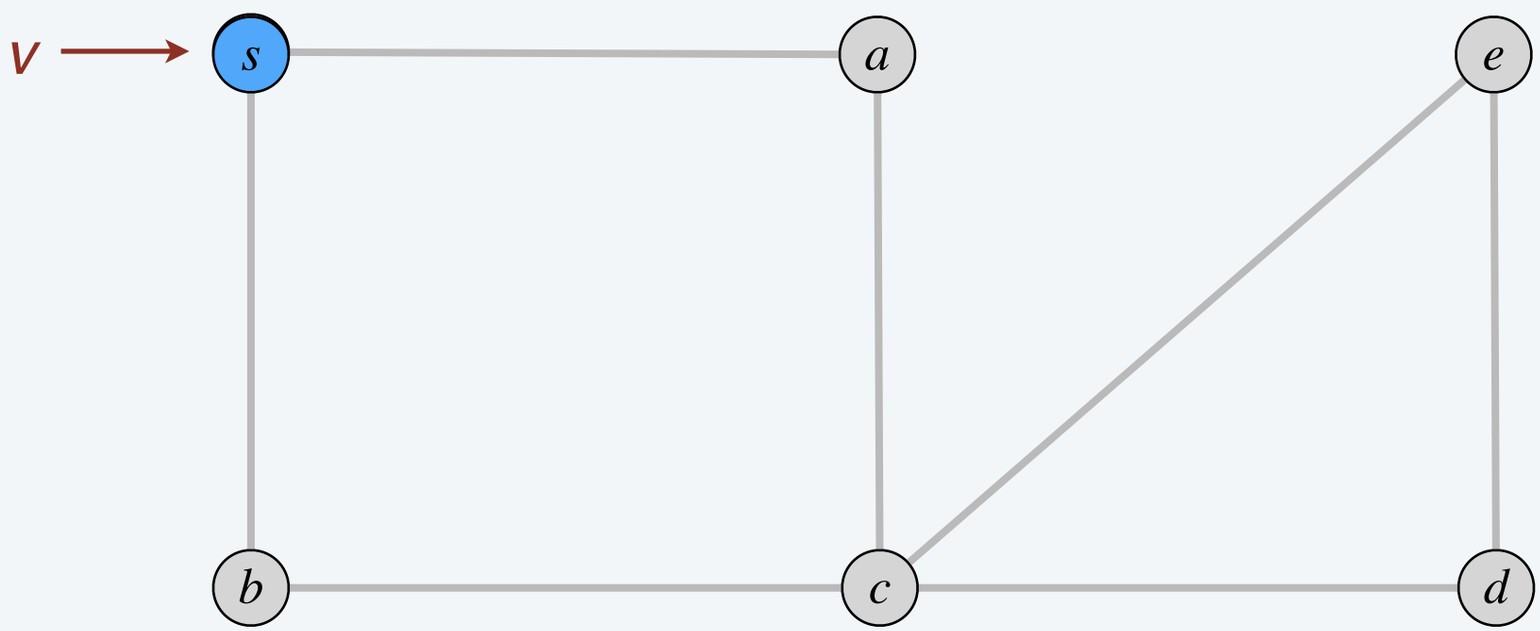
  remove ("pop") the vertex v from the front of S

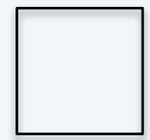
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

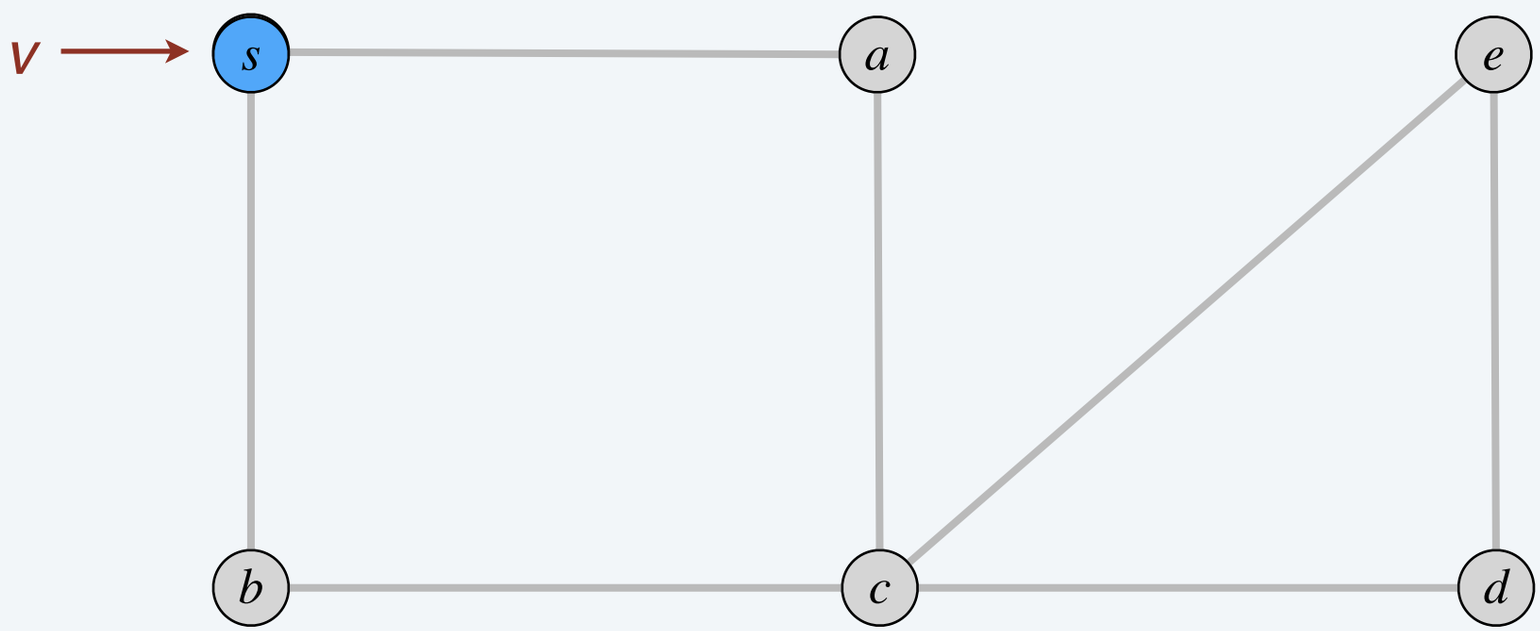
  remove ("pop") the vertex v from the front of S

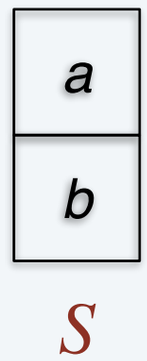
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





s

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

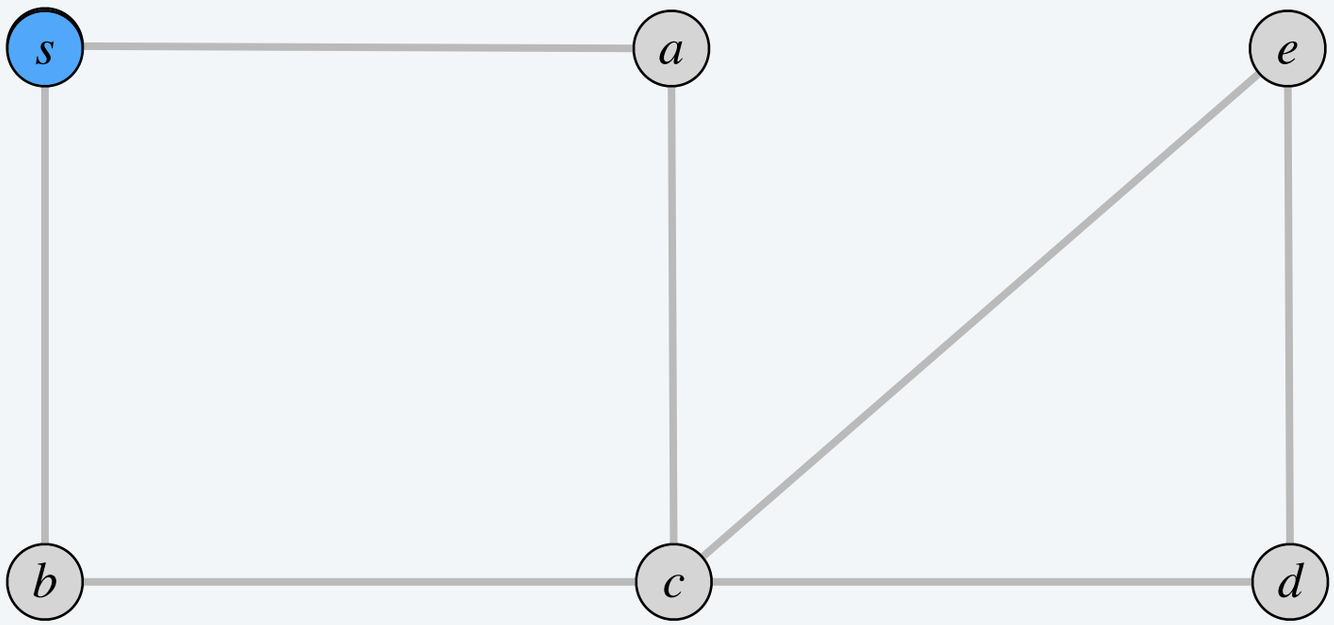
    remove ("pop") the vertex v from the front of S

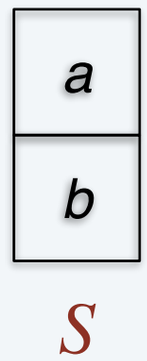
    IF v is unexplored

        mark v as explored

        FOR each edge (v, w) in v's adjacency-list

            add ("push") w to the front of S





s

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

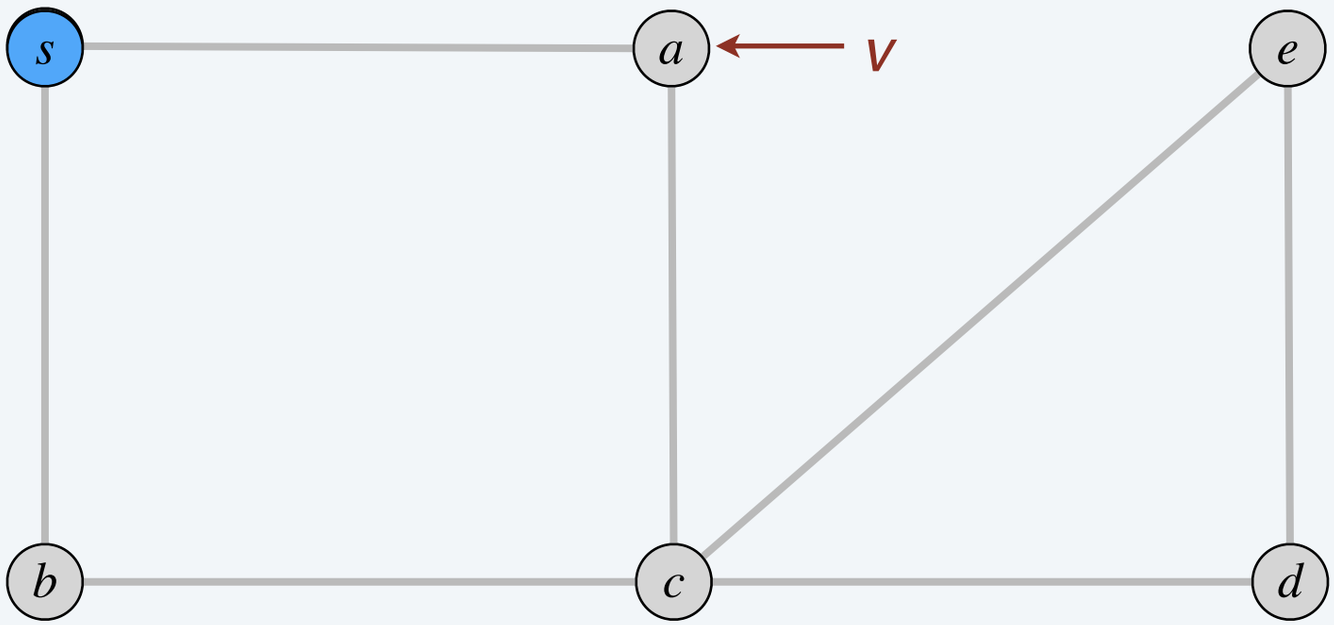
    remove ("pop") the vertex v from the front of S

    IF v is unexplored

        mark v as explored

        FOR each edge (v, w) in v's adjacency-list

            add ("push") w to the front of S



S

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

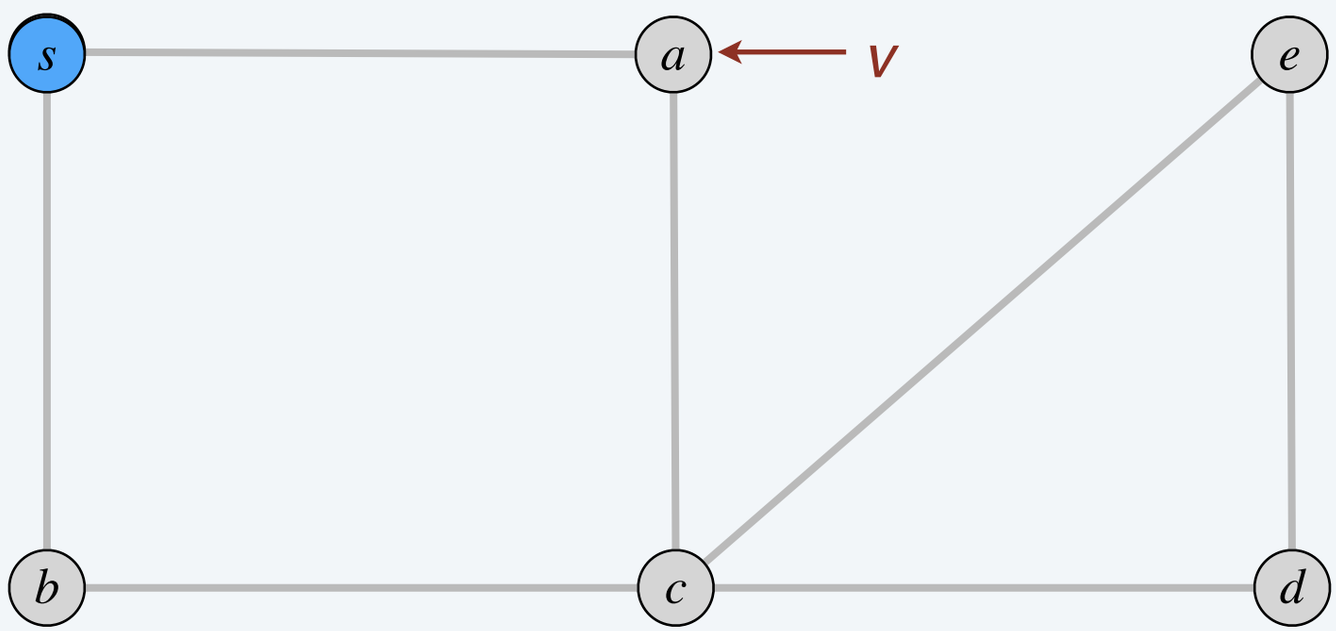
  remove ("pop") the vertex v from the front of S

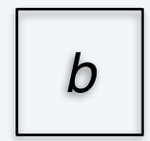
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





S

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

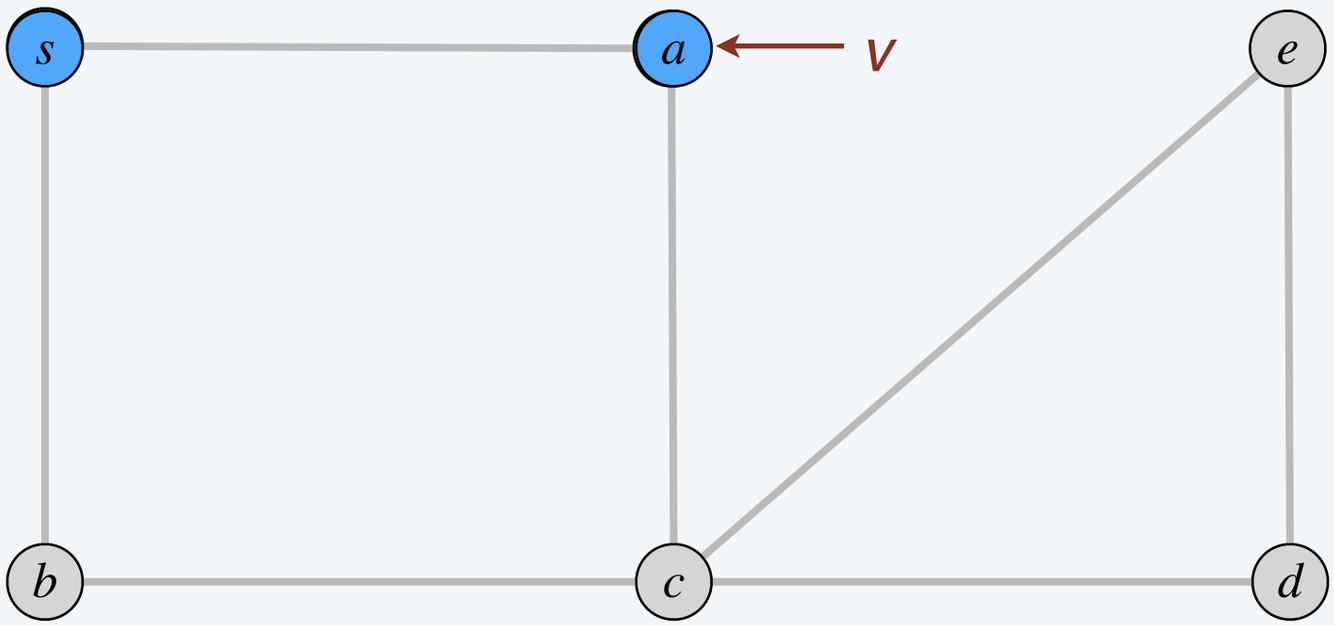
  remove ("pop") the vertex v from the front of S

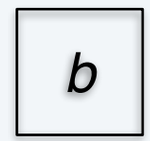
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





S

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

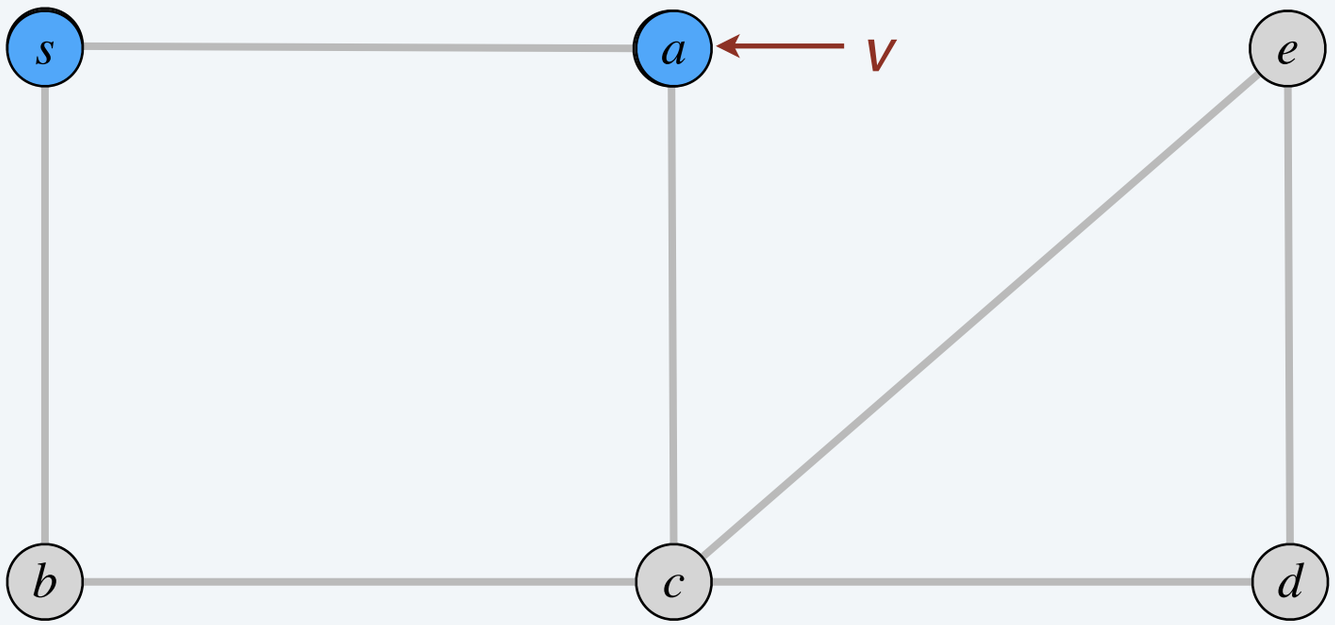
  remove ("pop") the vertex v from the front of S

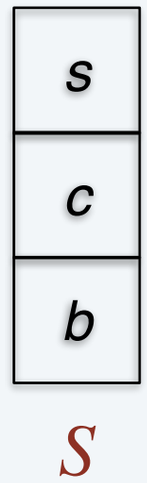
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





a

a

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

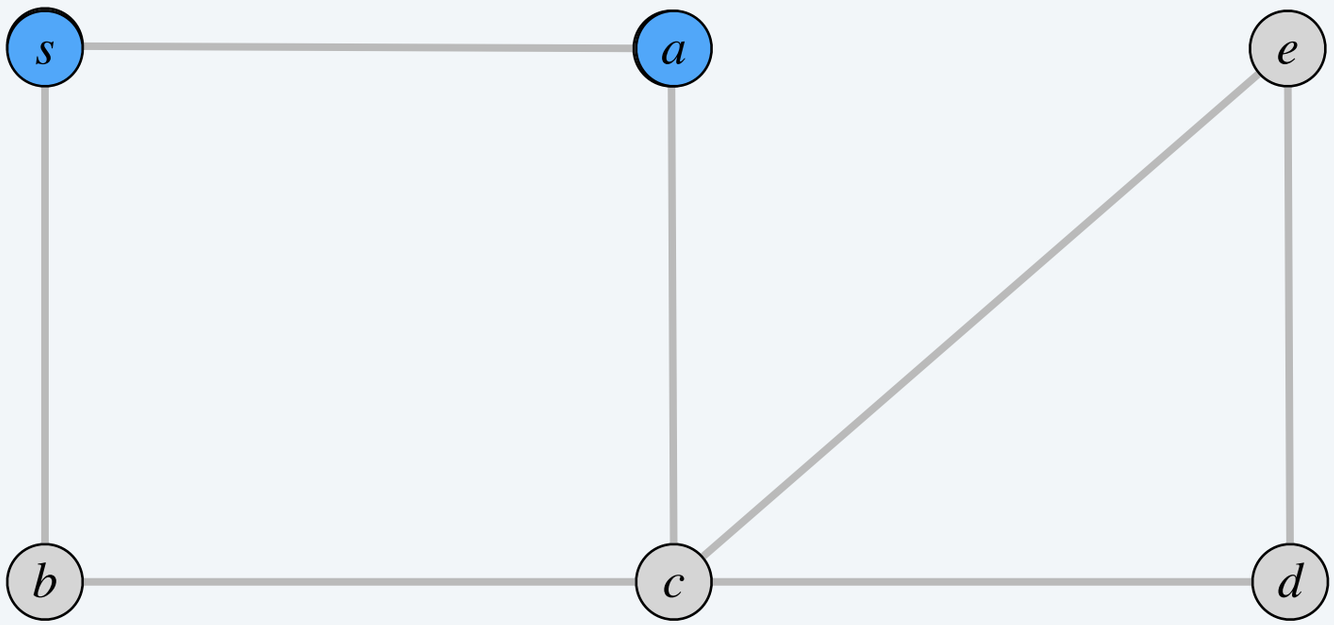
    remove ("pop") the vertex v from the front of S

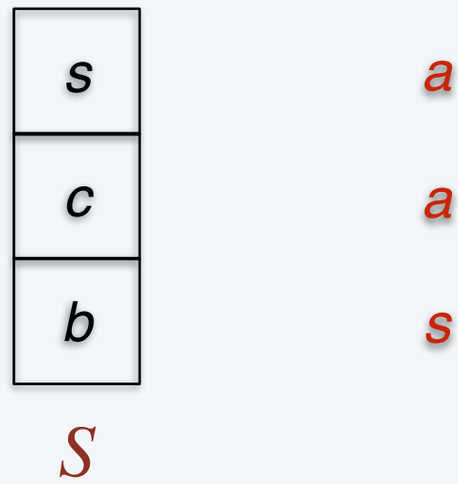
    IF v is unexplored

        mark v as explored

        FOR each edge (v, w) in v's adjacency-list

            add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

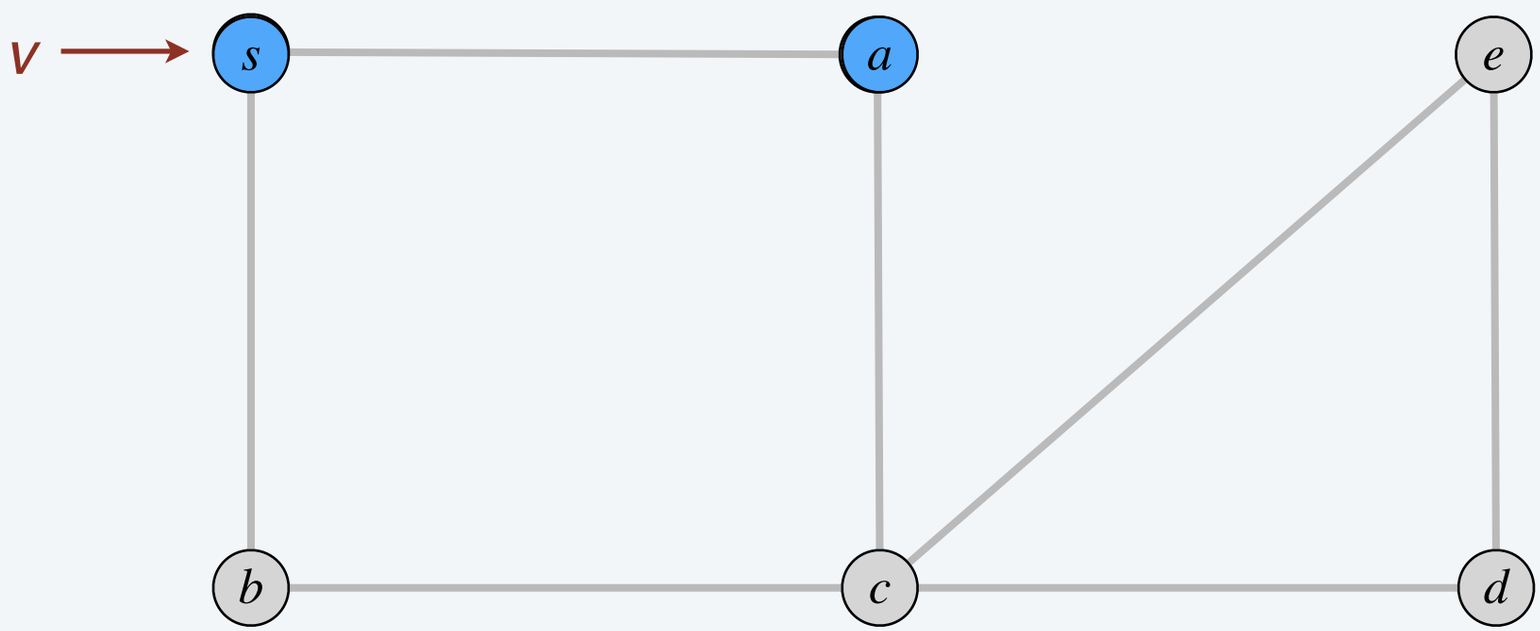
    remove ("pop") the vertex v from the front of S

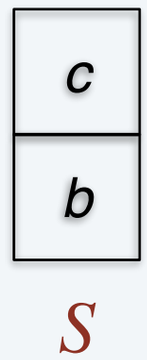
    IF v is unexplored

        mark v as explored

        FOR each edge (v, w) in v's adjacency-list

            add ("push") w to the front of S





a

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

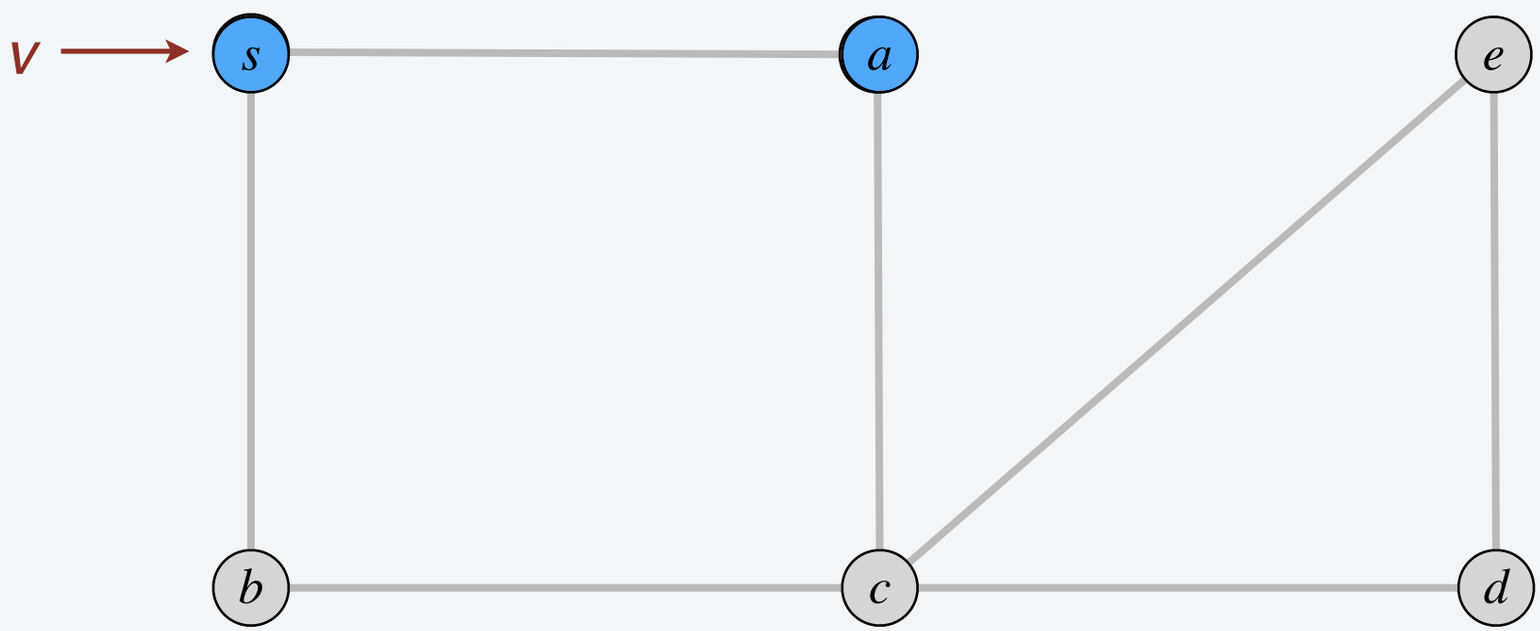
  remove ("pop") the vertex v from the front of S

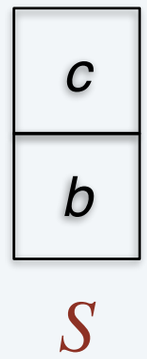
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





a

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

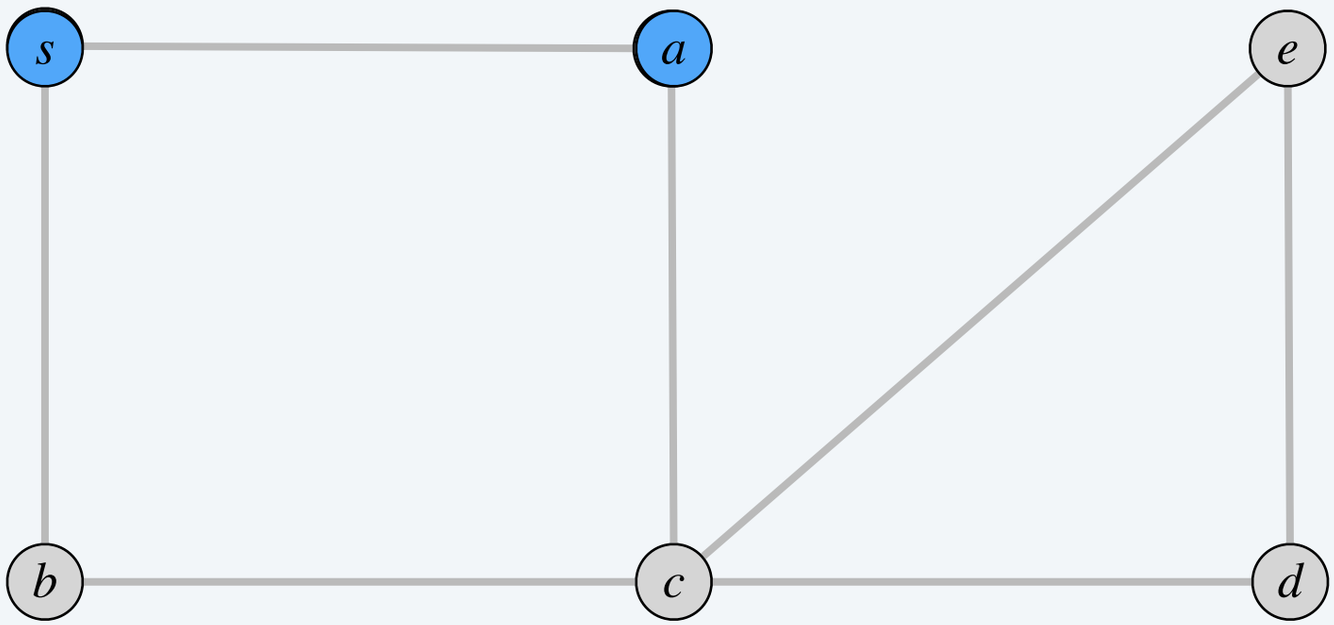
  remove ("pop") the vertex v from the front of S

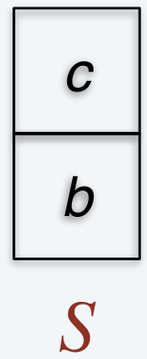
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





a

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

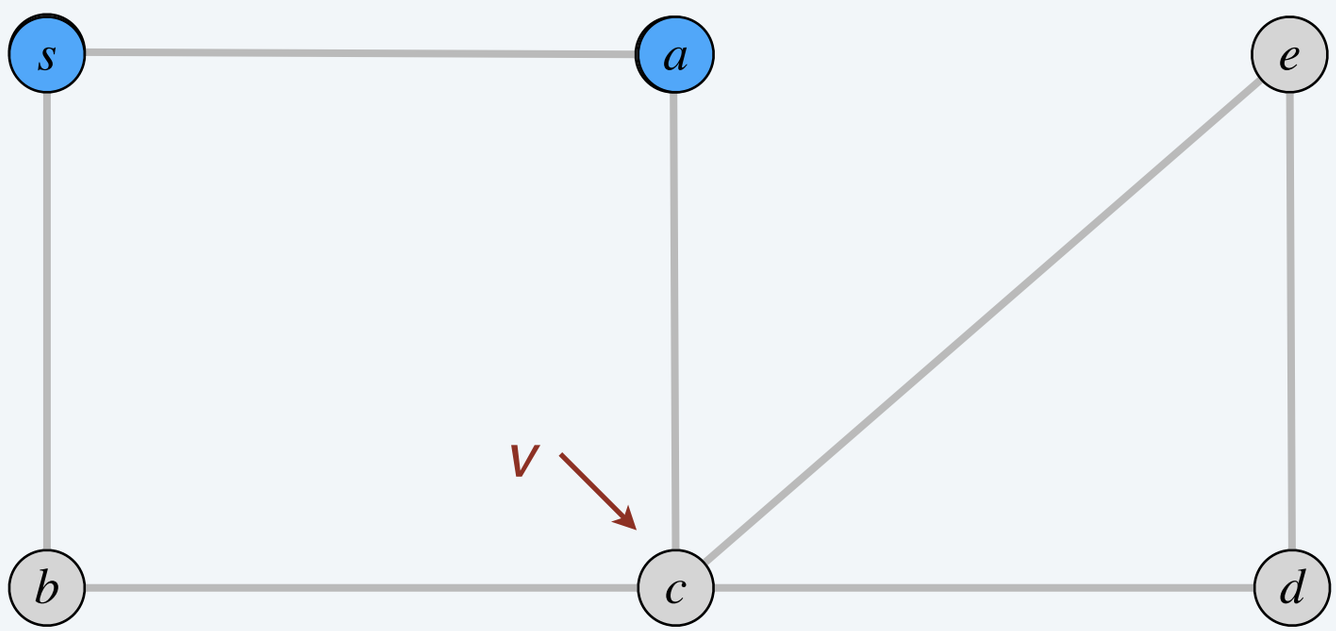
    remove ("pop") the vertex v from the front of S

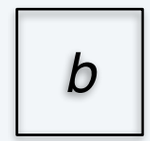
    IF v is unexplored

        mark v as explored

        FOR each edge (v, w) in v's adjacency-list

            add ("push") w to the front of S





S

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

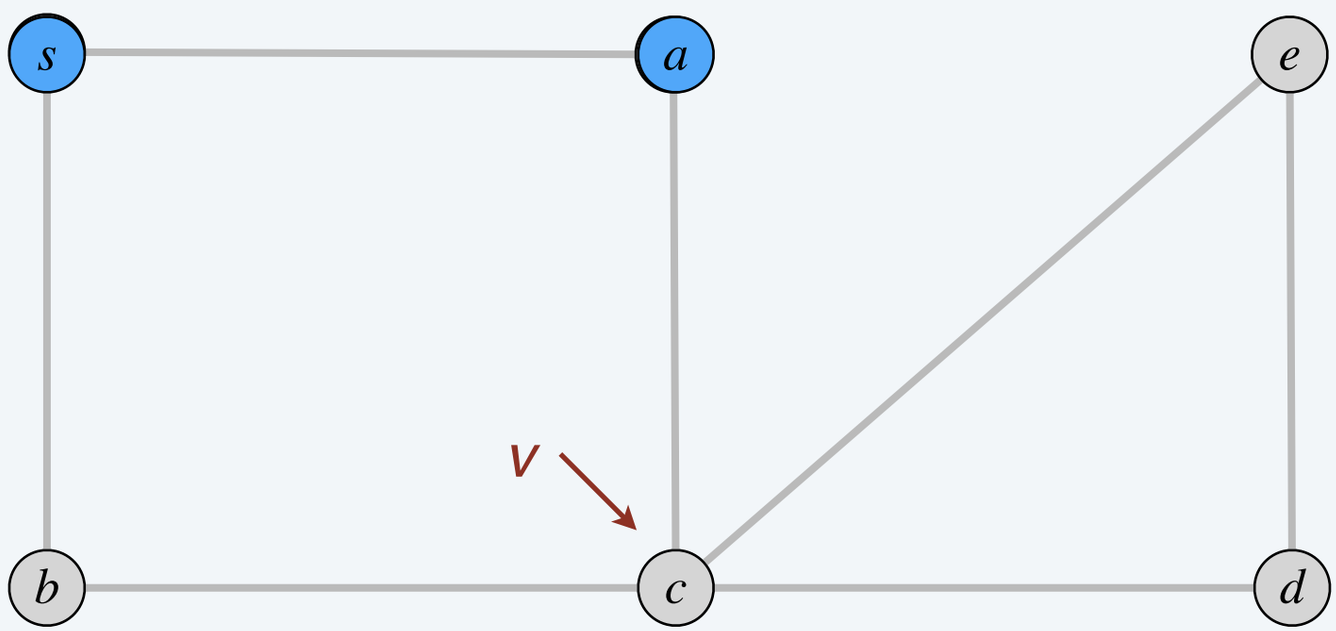
  remove ("pop") the vertex v from the front of S

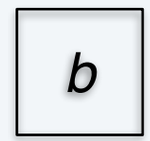
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





S

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

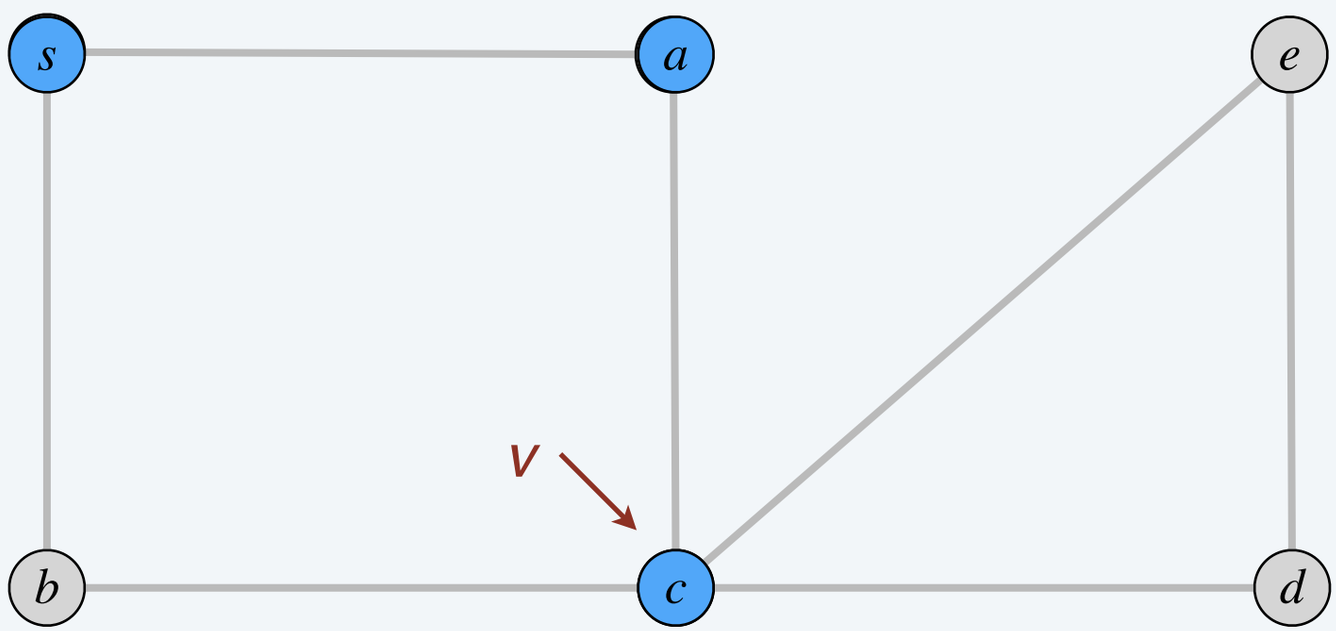
  remove ("pop") the vertex v from the front of S

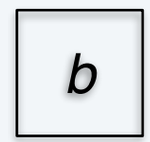
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





S

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

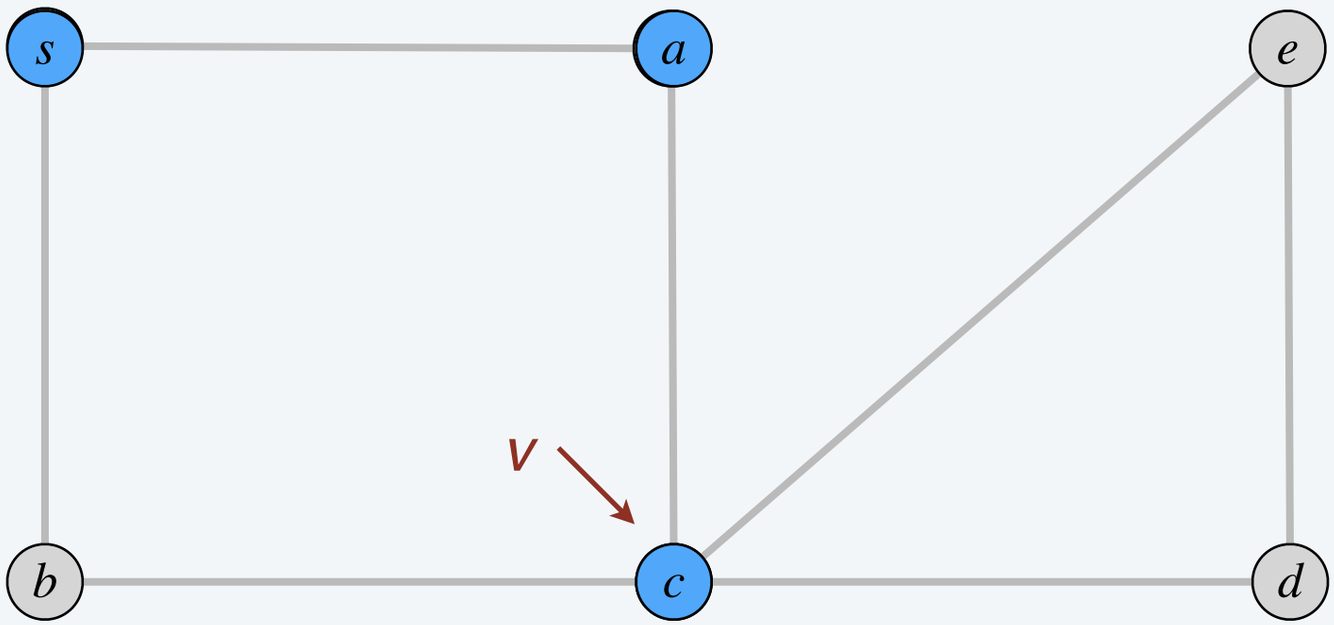
  remove ("pop") the vertex v from the front of S

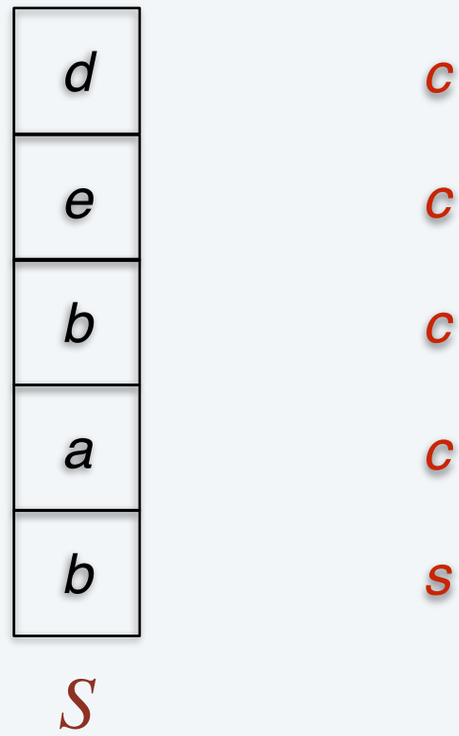
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

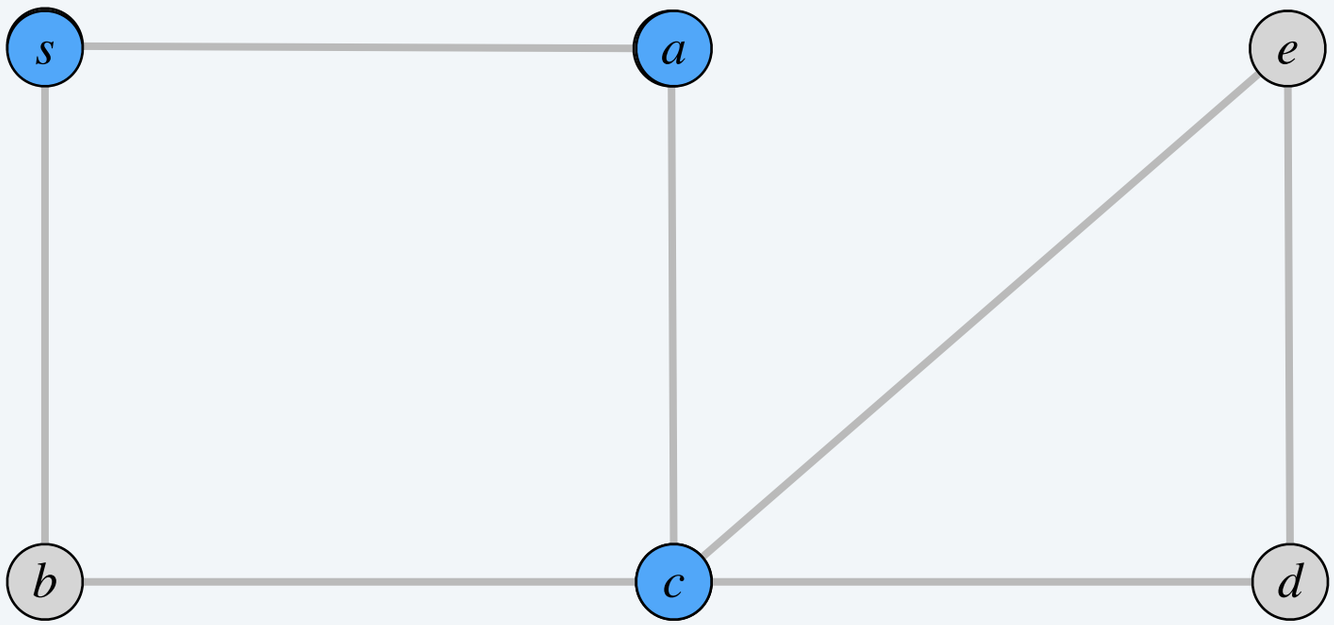
    remove ("pop") the vertex v from the front of S

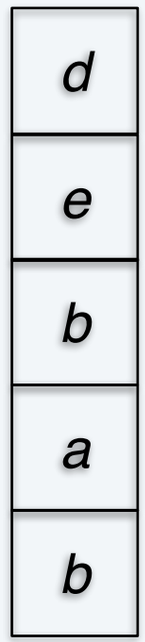
    IF v is unexplored

        mark v as explored

        FOR each edge (v, w) in v's adjacency-list

            add ("push") w to the front of S





c
c
c
c
s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

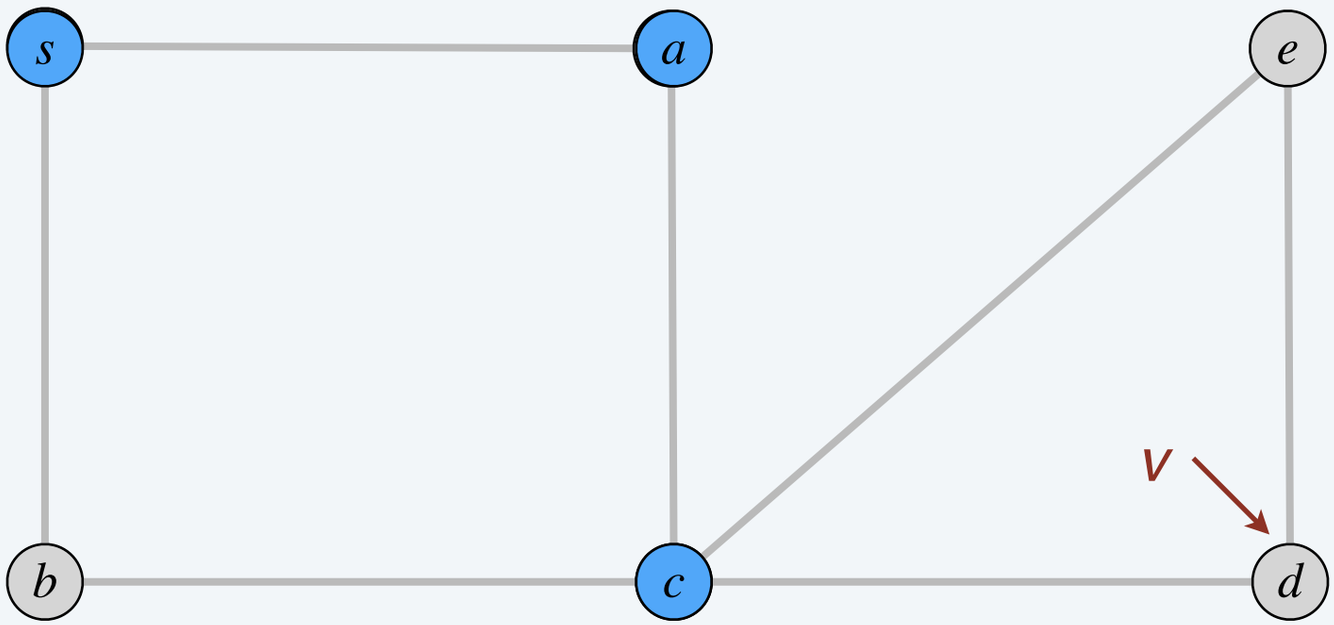
    remove ("pop") the vertex v from the front of S

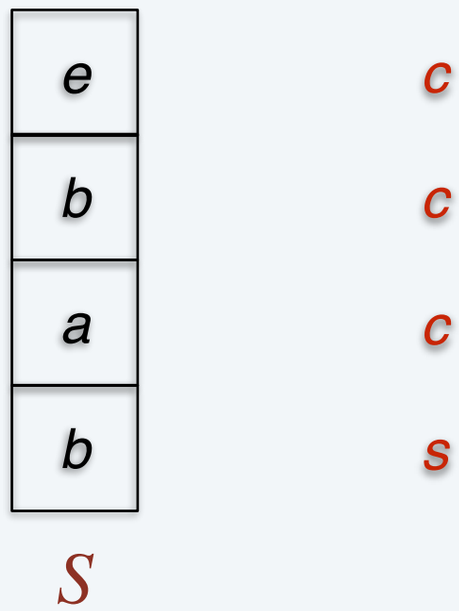
    IF v is unexplored

        mark v as explored

        FOR each edge (v, w) in v's adjacency-list

            add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

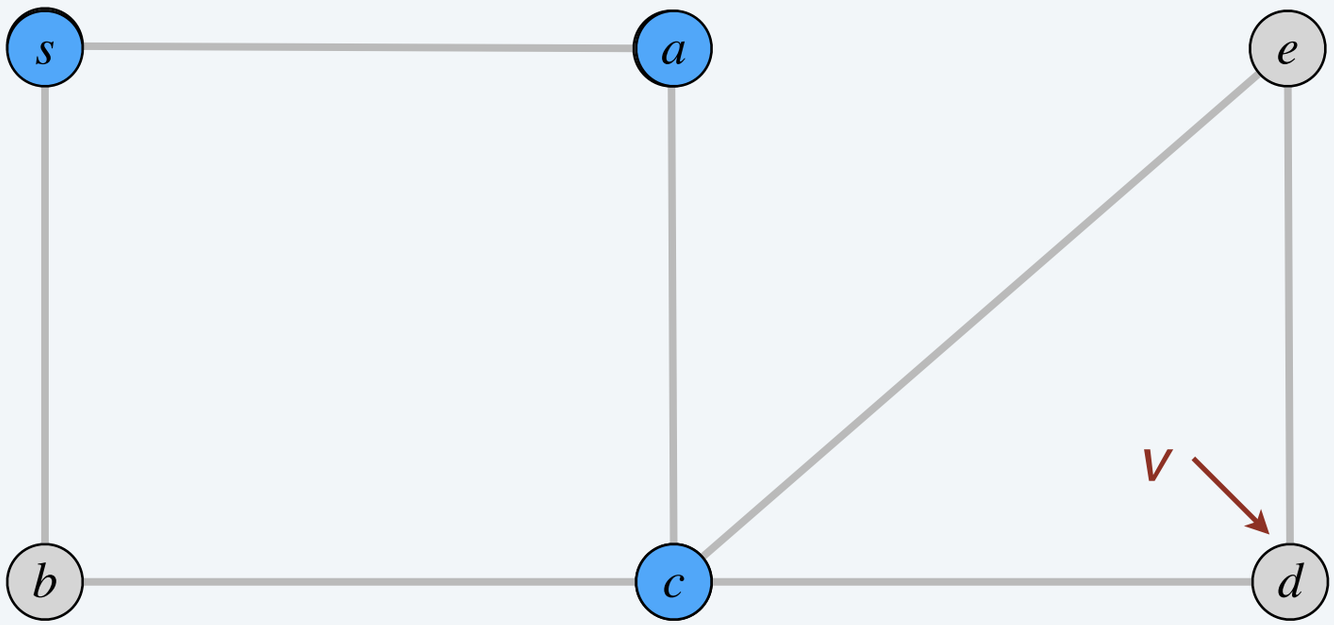
  remove ("pop") the vertex v from the front of S

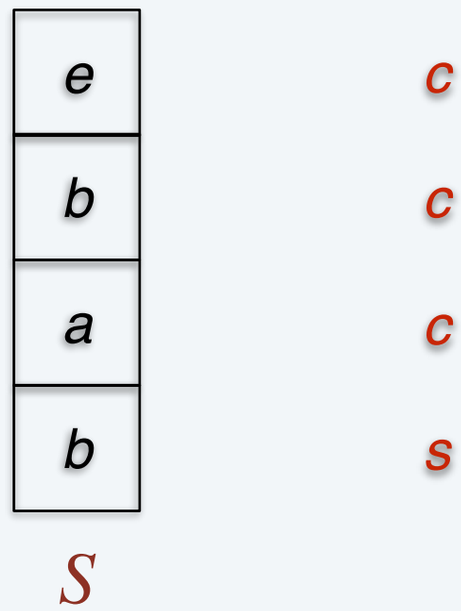
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

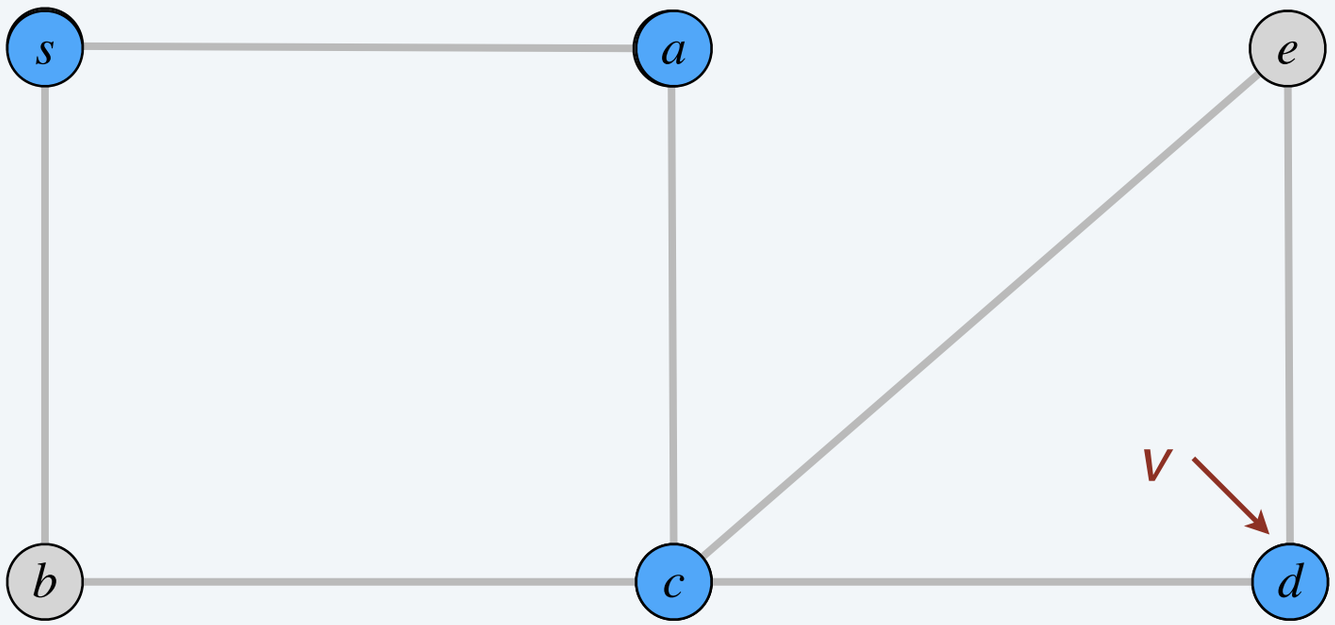
  remove ("pop") the vertex v from the front of S

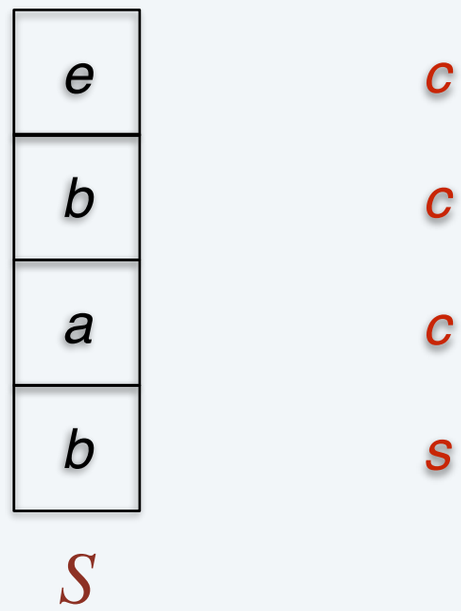
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

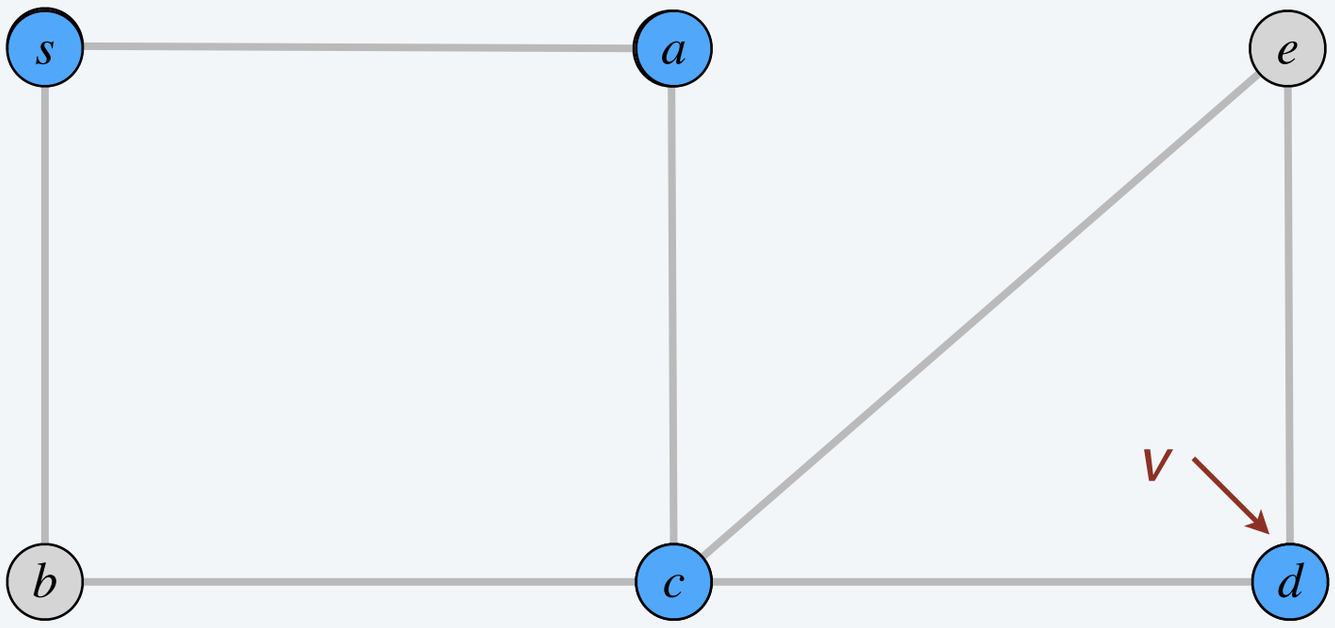
  remove ("pop") the vertex v from the front of S

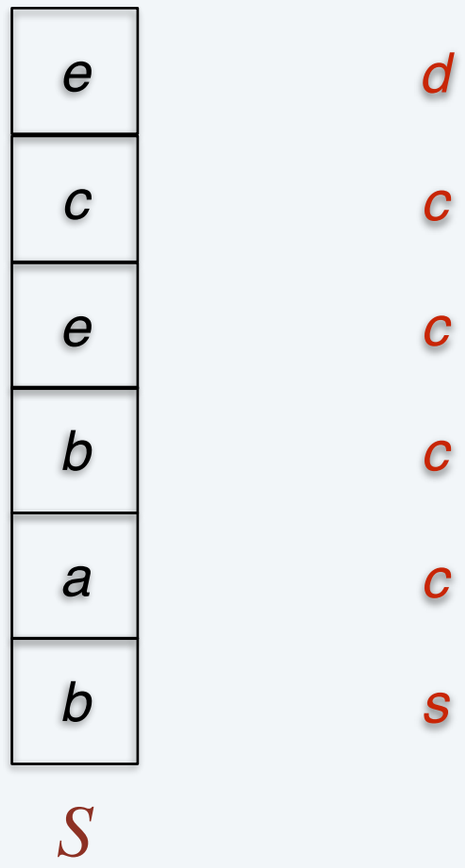
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

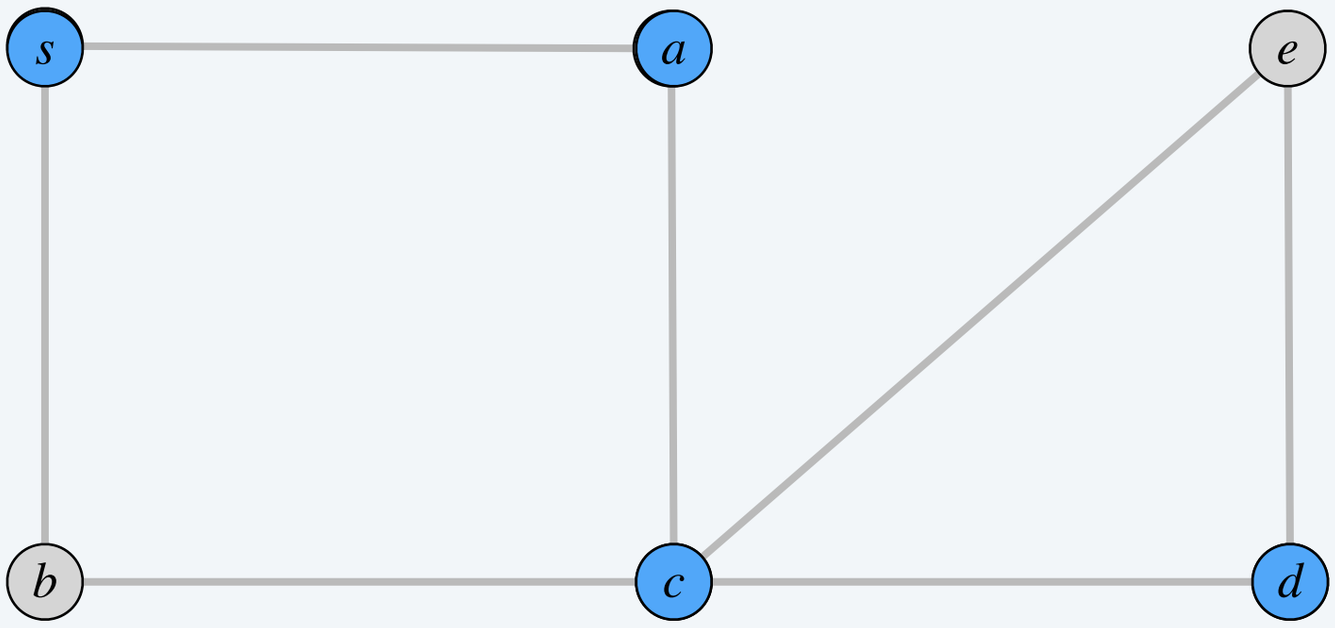
  remove ("pop") the vertex v from the front of S

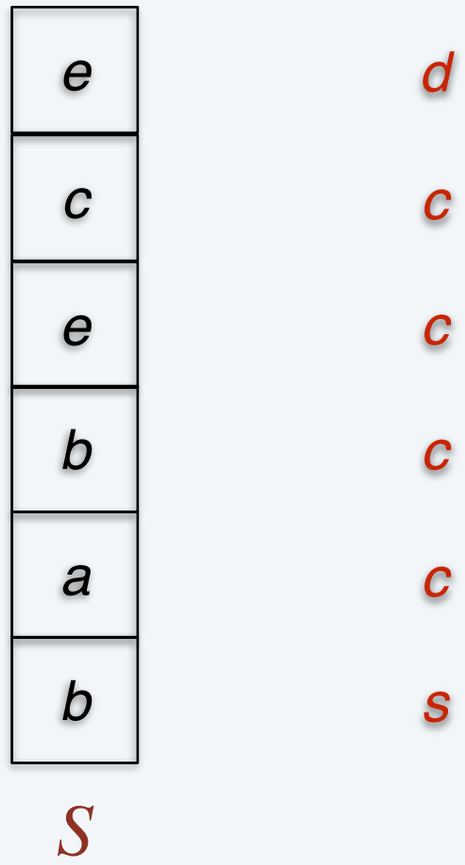
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

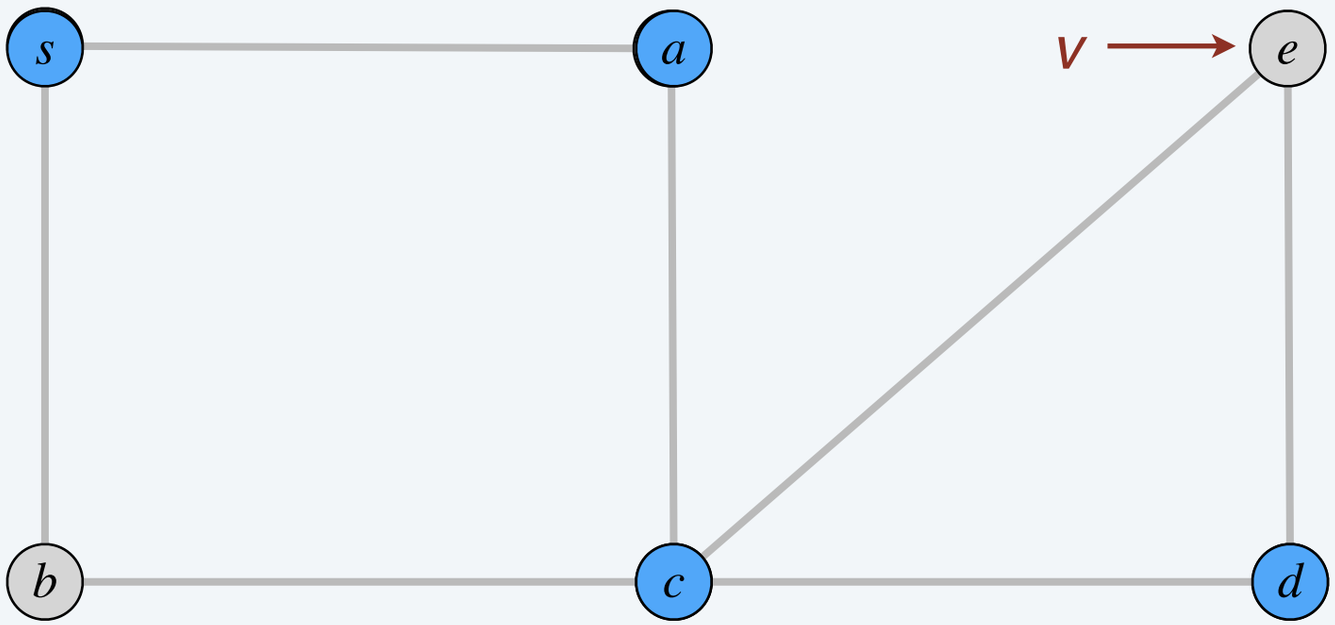
    remove ("pop") the vertex v from the front of S

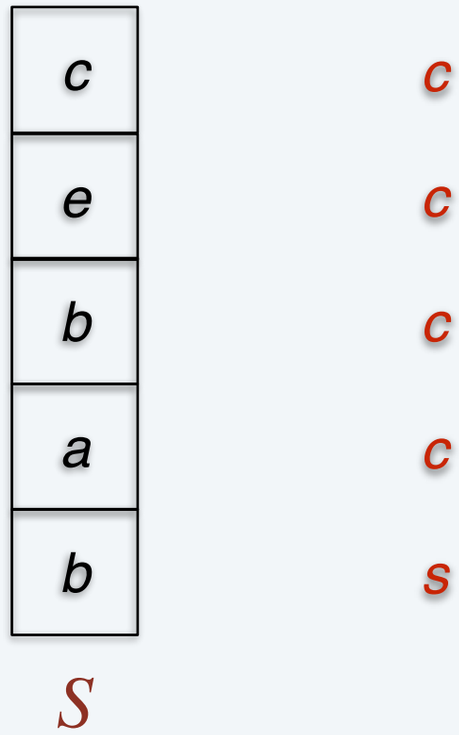
    IF v is unexplored

        mark v as explored

        FOR each edge (v, w) in v's adjacency-list

            add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

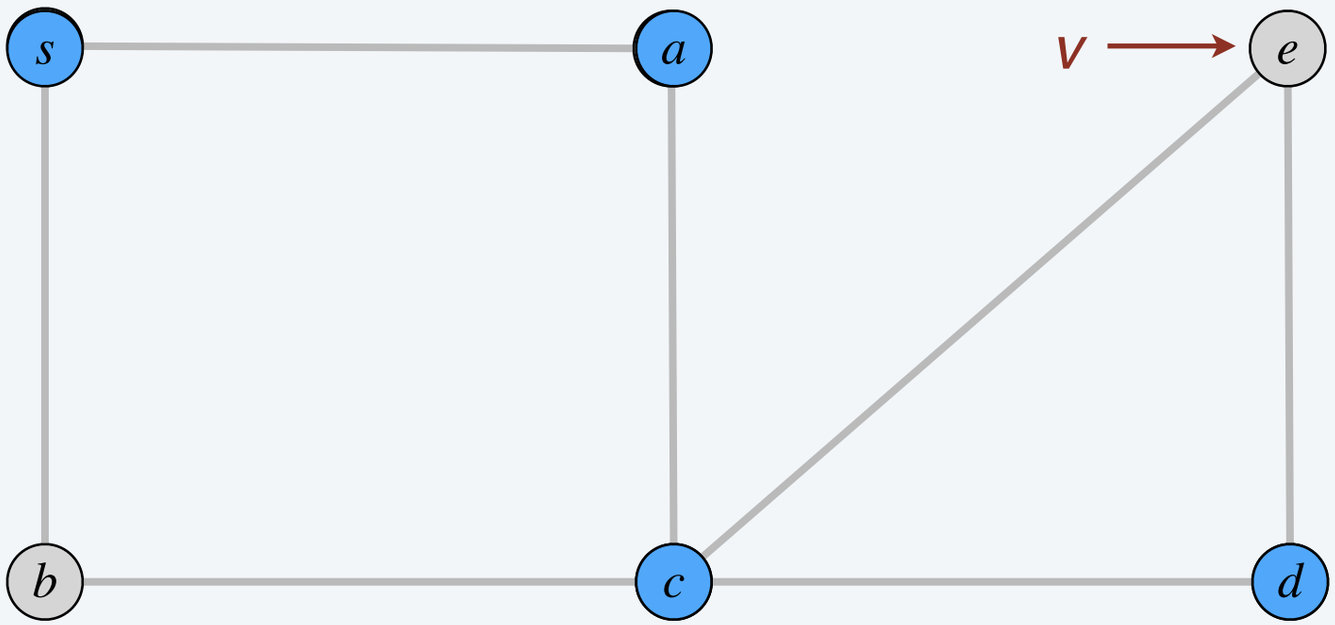
  remove ("pop") the vertex v from the front of S

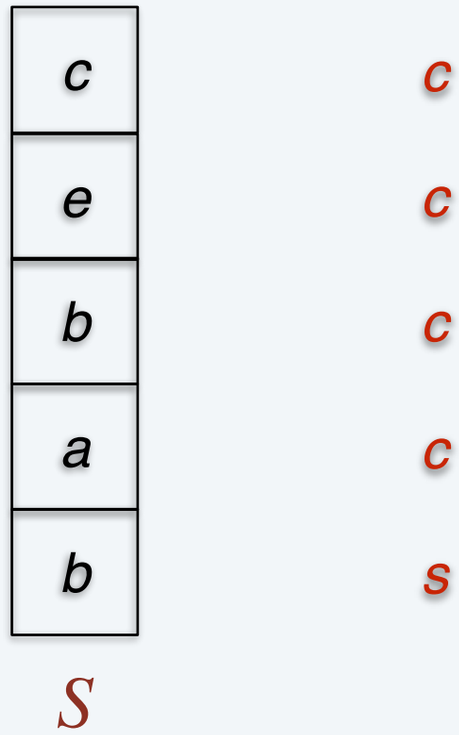
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

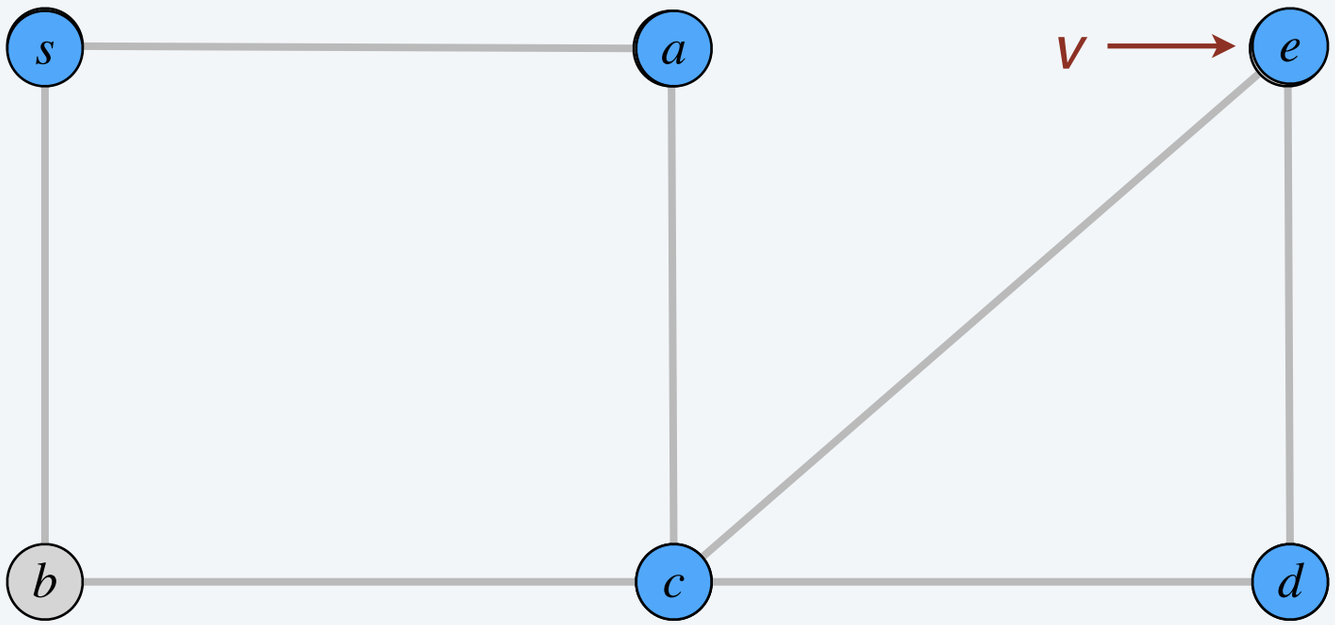
  remove ("pop") the vertex v from the front of S

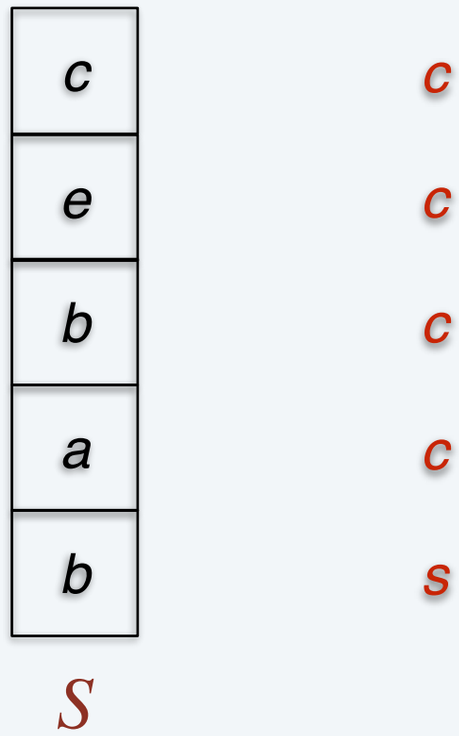
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

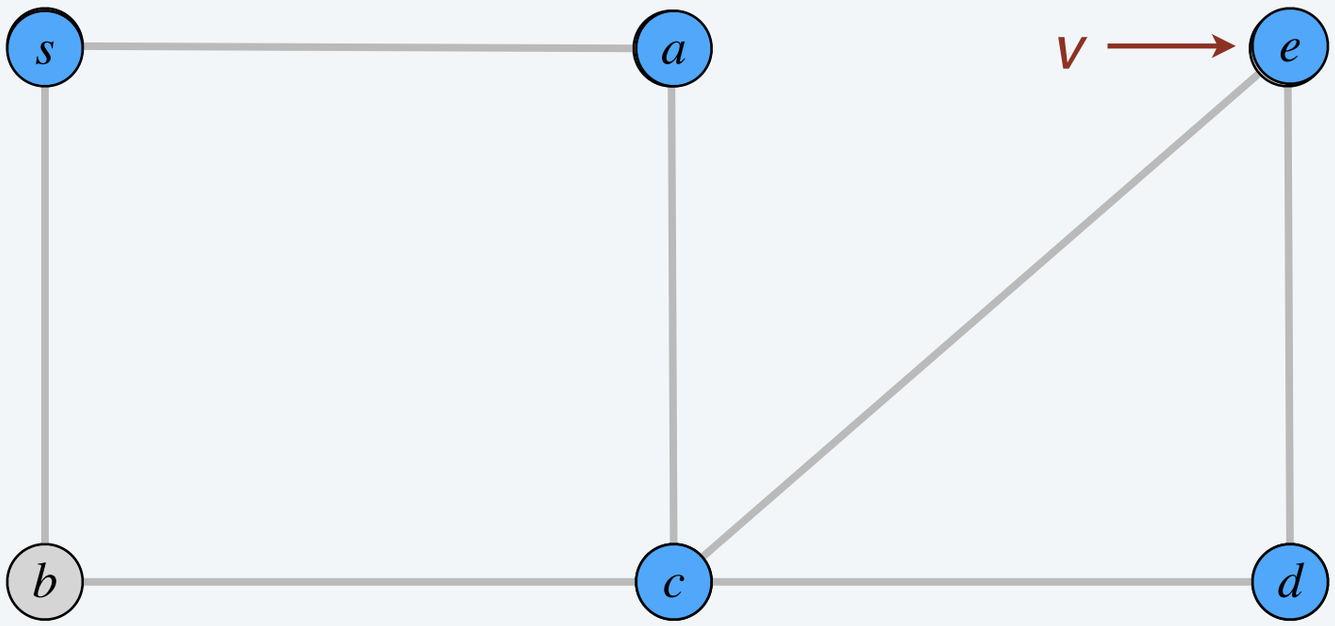
  remove ("pop") the vertex v from the front of S

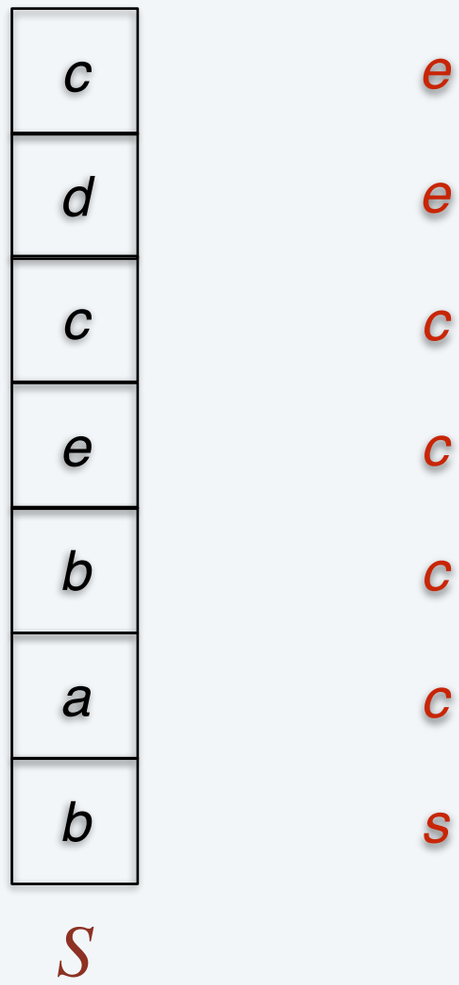
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

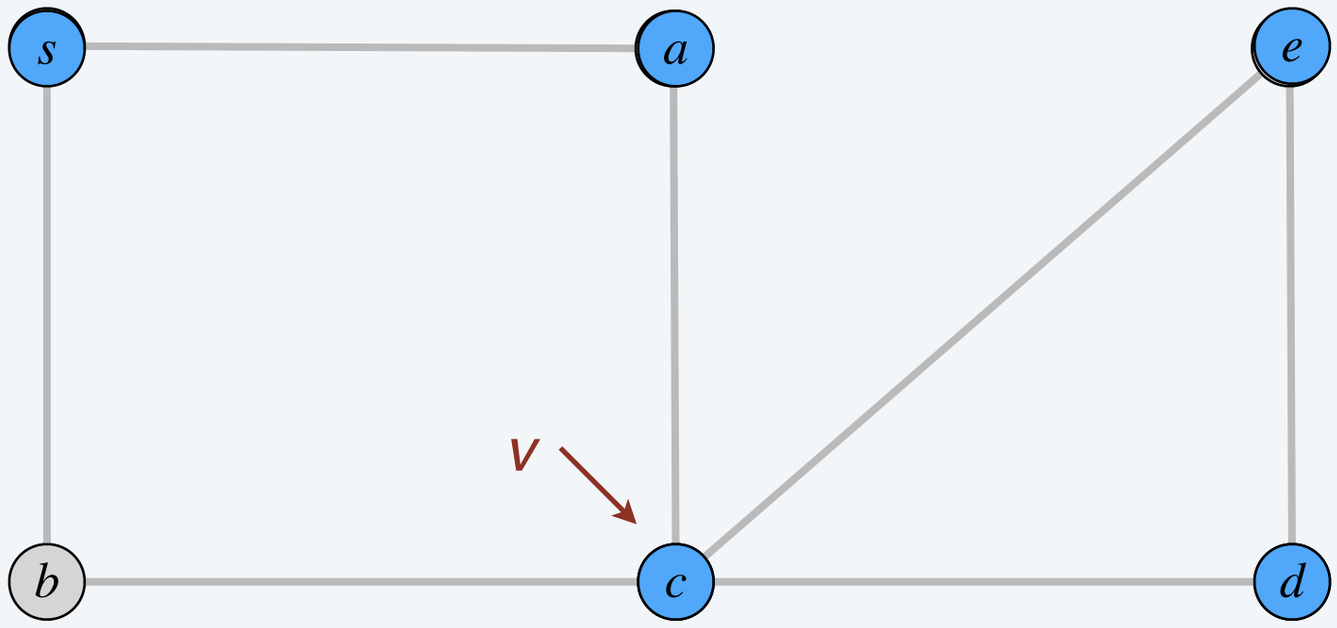
  remove ("pop") the vertex v from the front of S

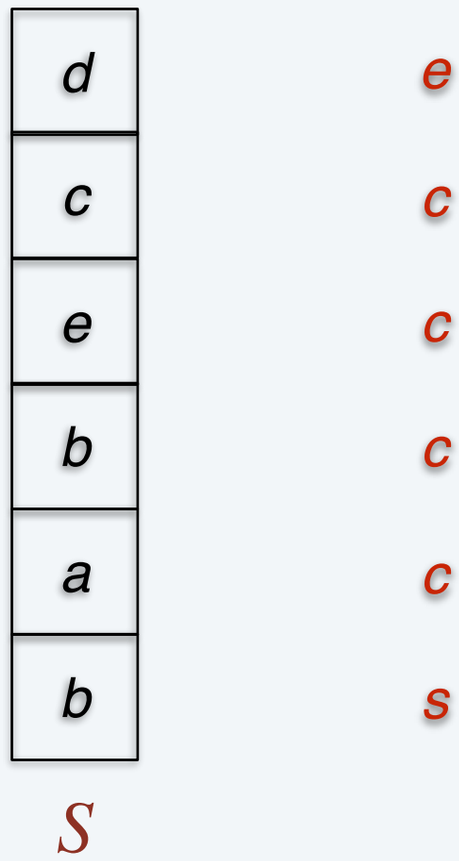
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

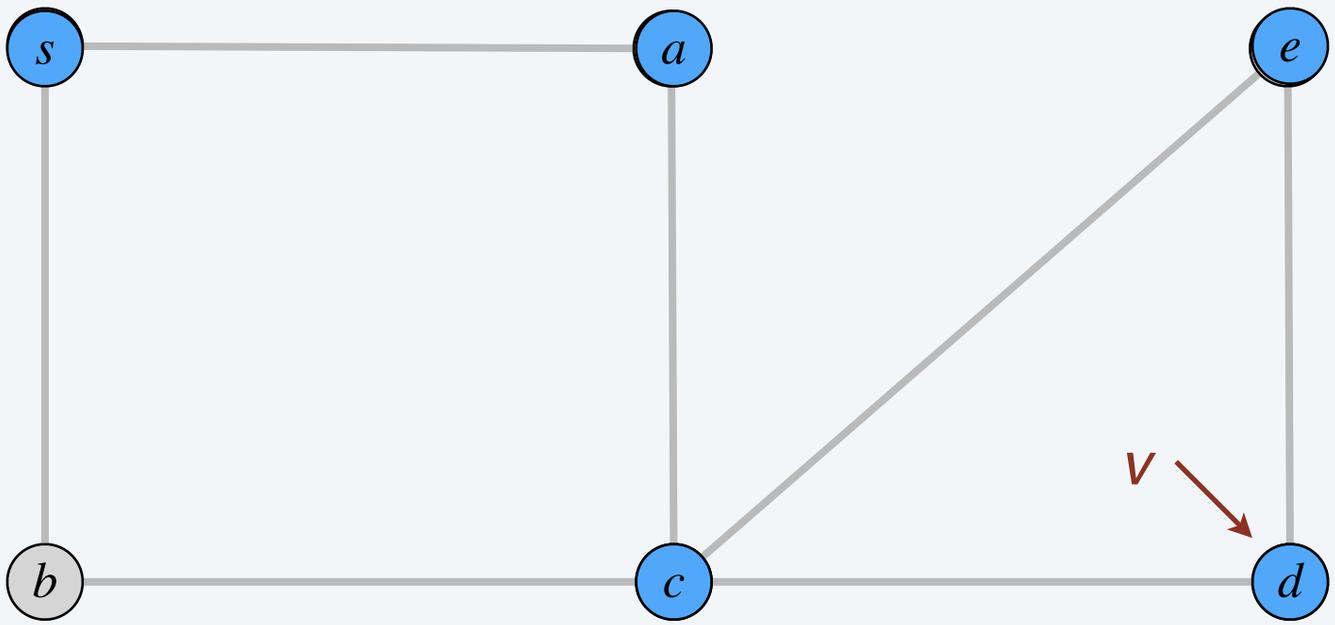
  remove ("pop") the vertex v from the front of S

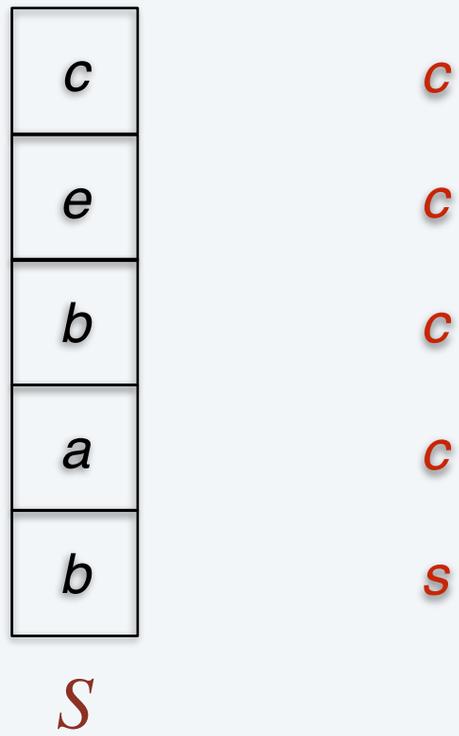
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

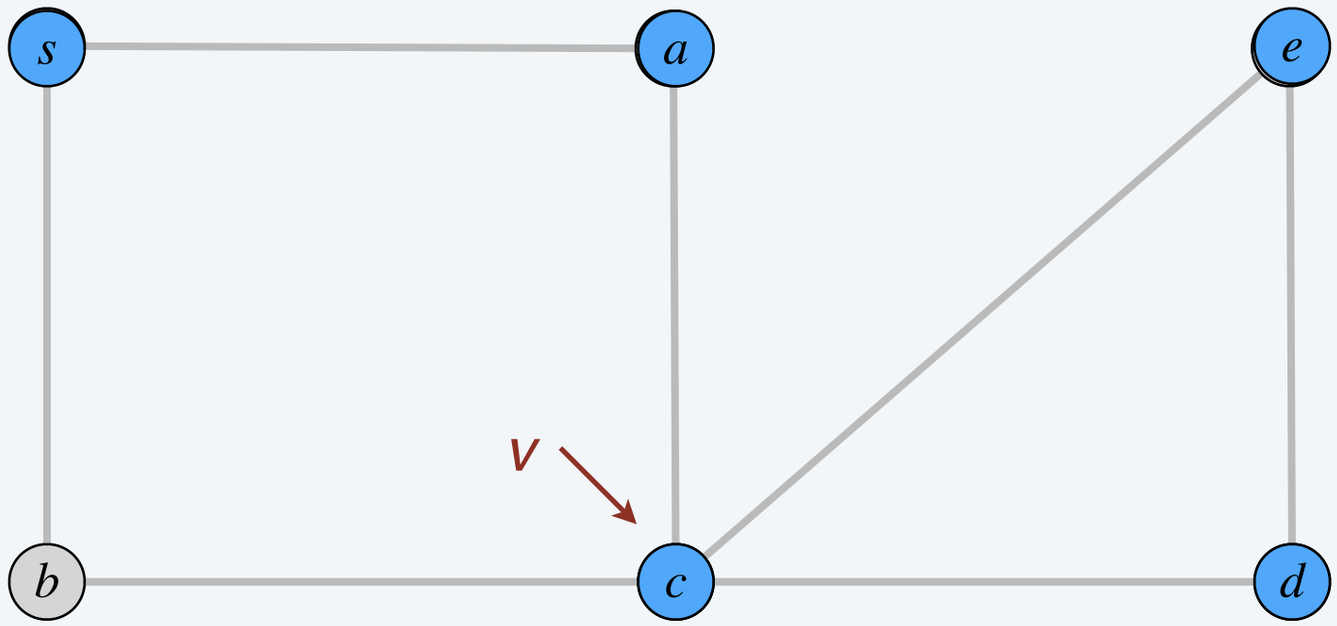
  remove ("pop") the vertex v from the front of S

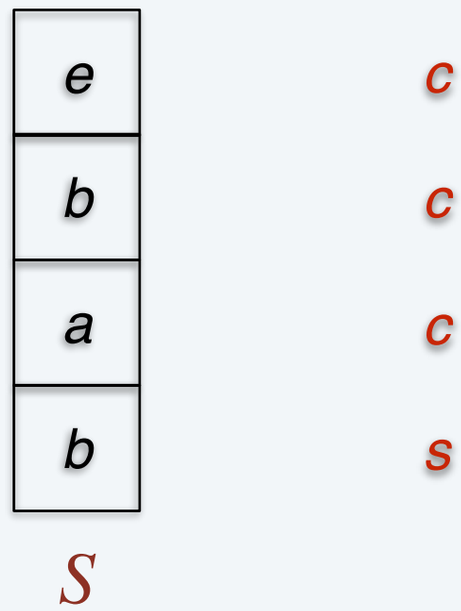
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

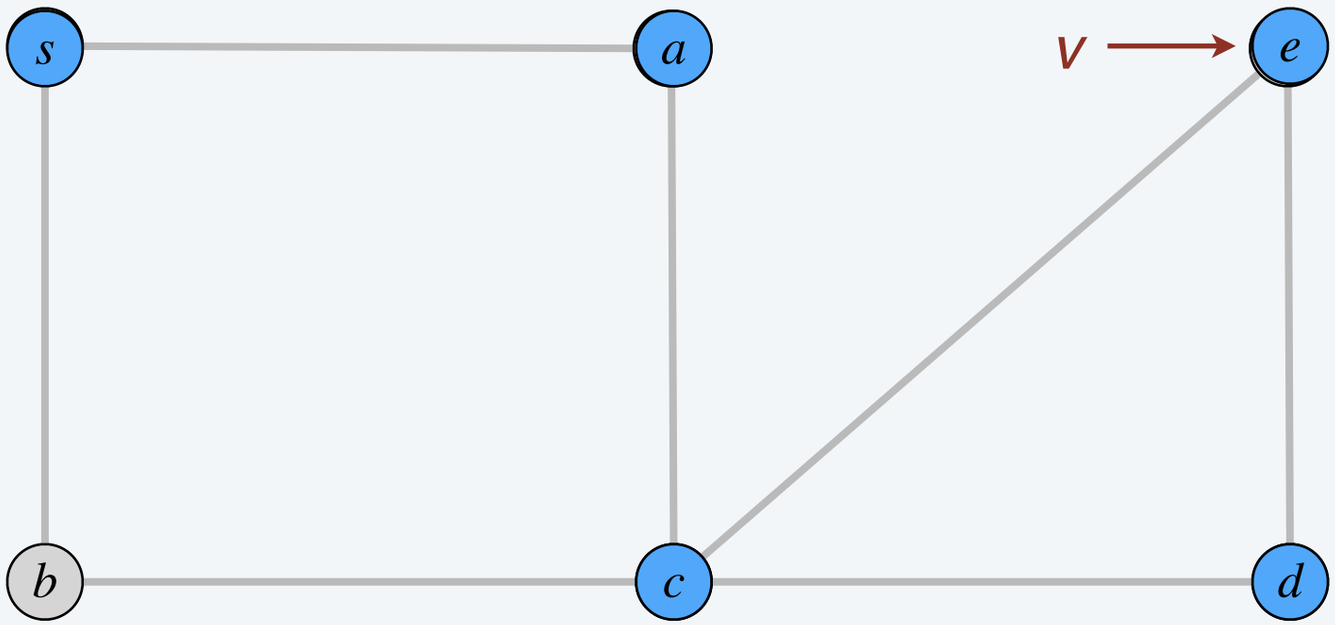
  remove ("pop") the vertex v from the front of S

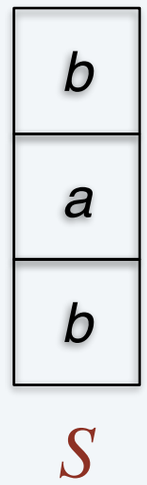
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





c

c

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

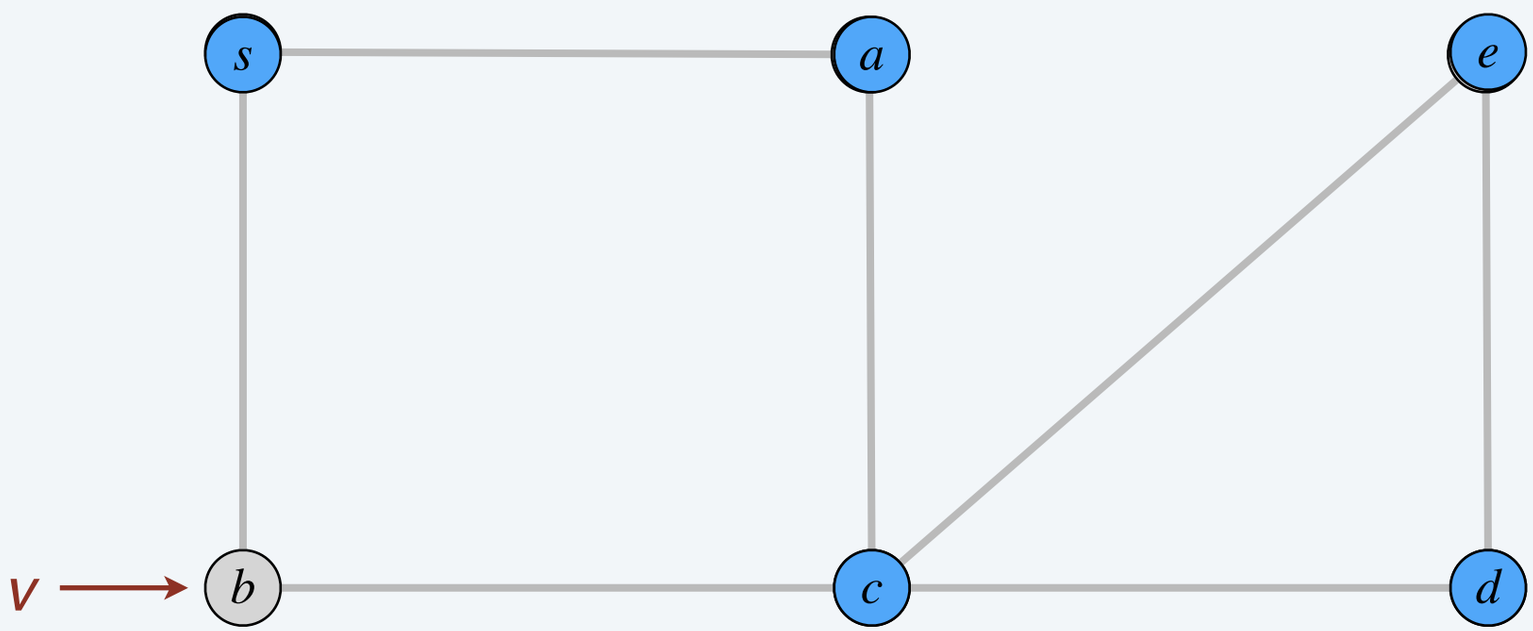
  remove ("pop") the vertex v from the front of S

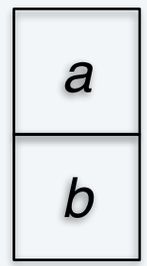
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





c

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

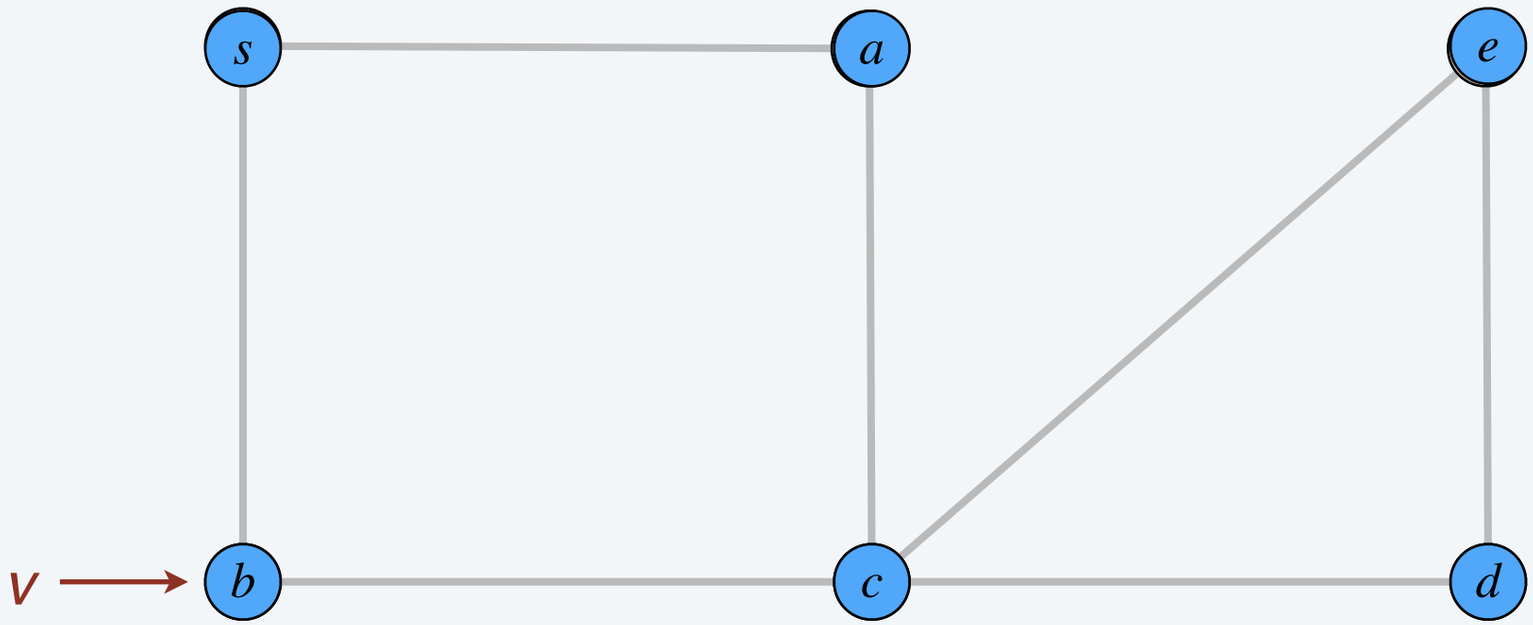
  remove ("pop") the vertex v from the front of S

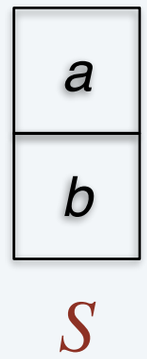
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





c

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

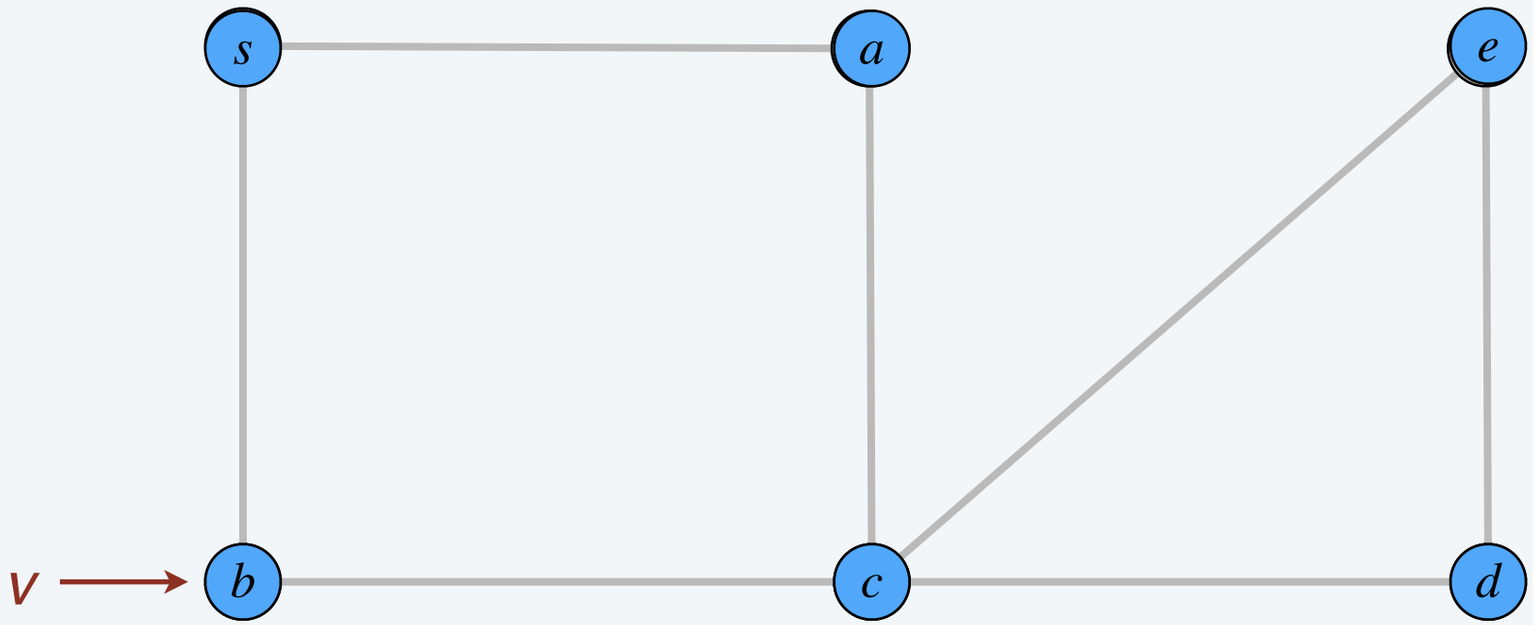
  remove ("pop") the vertex v from the front of S

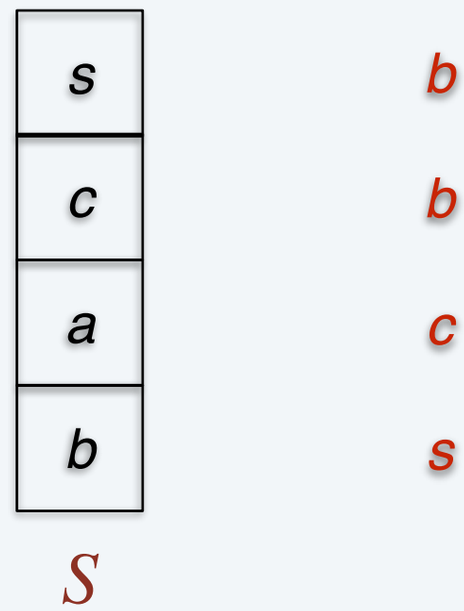
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

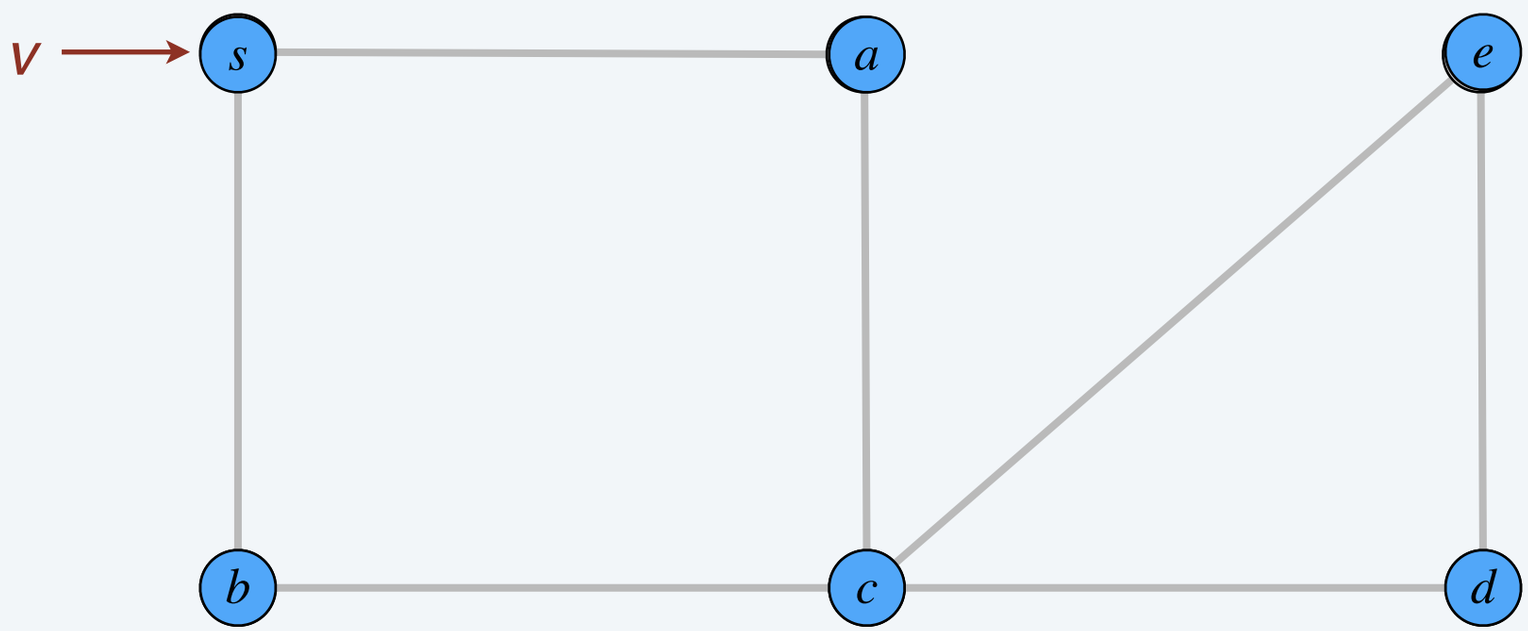
  remove ("pop") the vertex v from the front of S

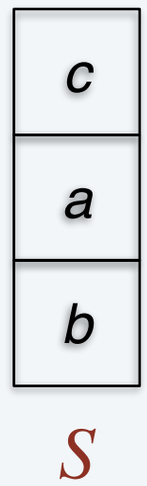
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





b

c

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

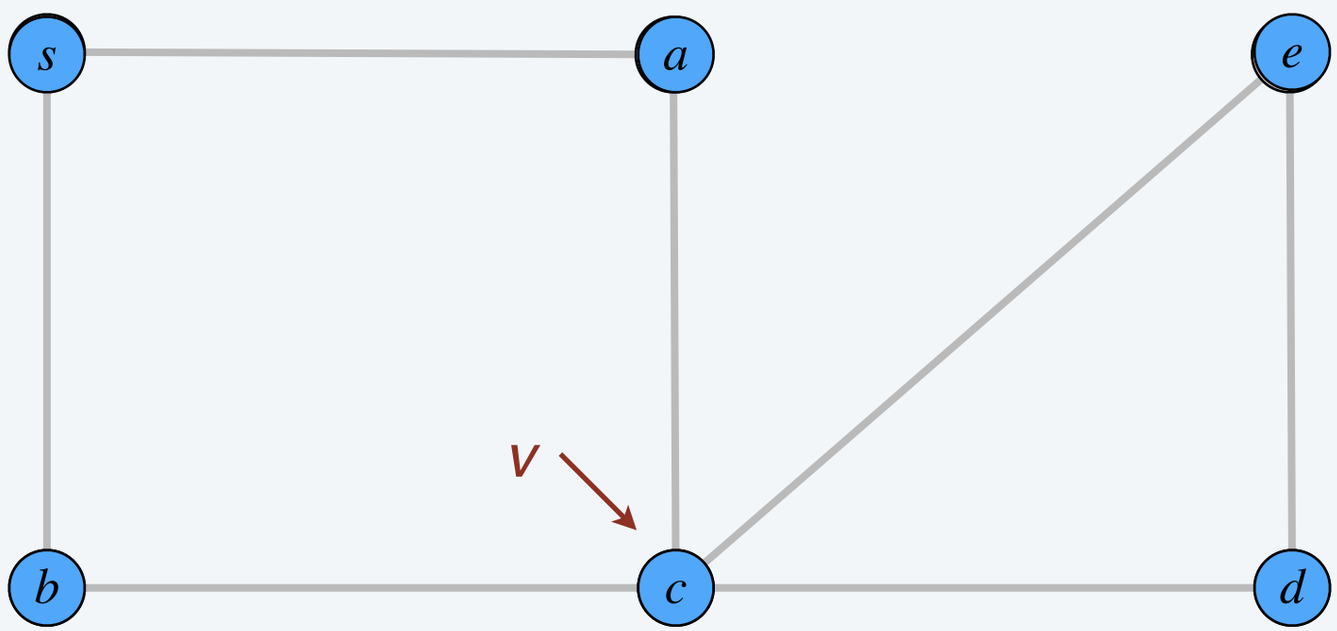
  remove ("pop") the vertex v from the front of S

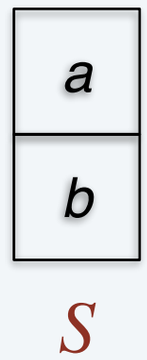
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





c

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

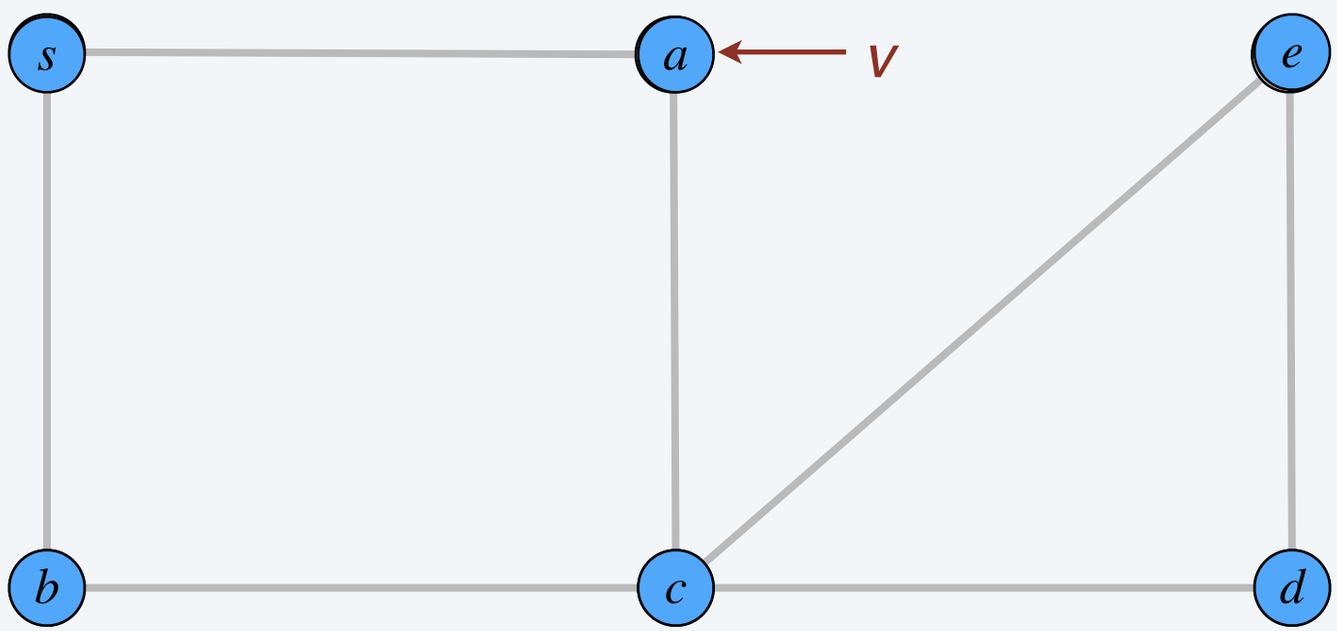
  remove ("pop") the vertex v from the front of S

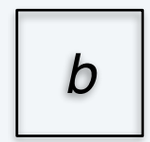
  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S





S

s

DFS demo

mark all vertices as unexplored

$S \leftarrow$ a stack data structure, initialized with s

WHILE S is not empty

  remove ("pop") the vertex v from the front of S

  IF v is unexplored

    mark v as explored

    FOR each edge (v, w) in v's adjacency-list

      add ("push") w to the front of S