CH-231-A

Algorithms and Data Structures

ADS

Lecture 34

Dr. Kinga Lipskoch

Spring 2020

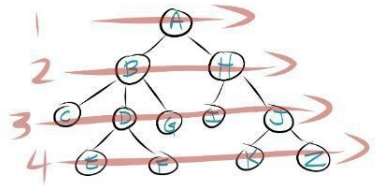
Breadth-First Search (BFS)

Problem:

- Given (directed or undirected) graph $G = (V, E)$ and a starting vertex $s \in V$.
- Systematically explore all vertices reachable from s.

BFS strategy:

- First find all vertices of distance 1 from s, then of distance 2, then of distance 3, etc.

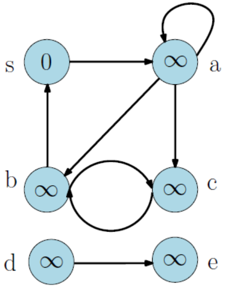


BFS Approach

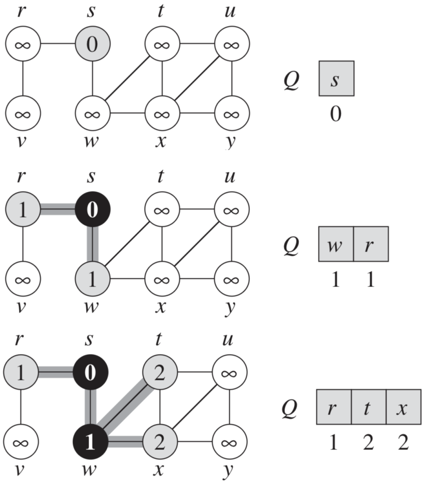
- Use adjacency-list representation.
- Use a color attribute for each *vertex* $\in \{\text{white}, \text{gray}, \text{black}\}$.
  - white: not detected yet
  - gray: just detected, waiting for us to explore their adjacency lists
  - black: done, all neighbors have been visited
- Store all gray vertices in a queue (FIFO principle).
- In addition, store for each vertex an attribute with the (topological) distance to starting vertex s.
- Finally, also store a pointer to the predecessor.

BFS Algorithm

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

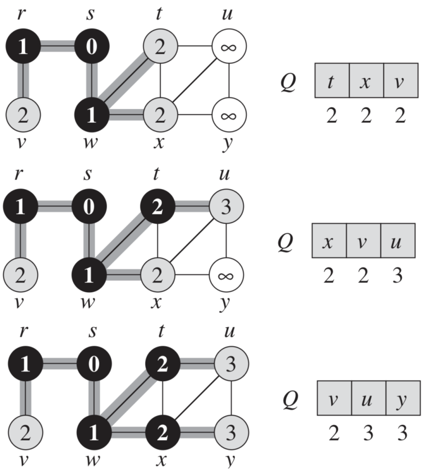


BFS Example (1)




```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

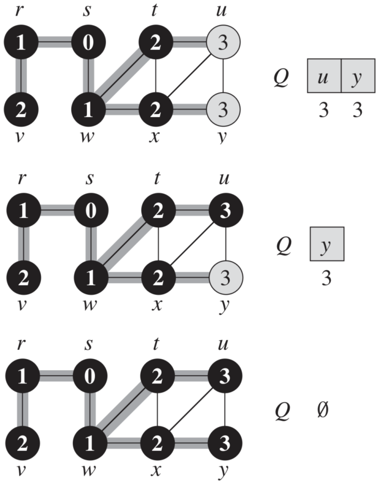
BFS Example (2)



BFS(G, s)

```
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

BFS Example (3)




```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = Ø
ENQUEUE(Q, s)
while Q ≠ Ø
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

BFS Analysis

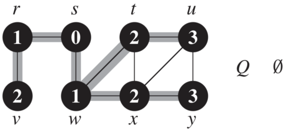
- Each vertex is enqueued and dequeued once.
- Each queue operation is O(1).
- Total time for queue operations is $O(|V|)$.
- Loop over adjacency list of all vertices is in total $\Theta(|E|)$.
- Together, we get a time complexity of $O(|V| + |E|)$.

Breadth-First Tree

- When storing the predecessors, we can construct the predecessor subgraph $G_\pi = (V_\pi, E_\pi)$ of G with

$$V_\pi = \{v \in V \mid v.\pi \neq NIL\} \cup \{s\}$$
$$E_\pi = \{(v.\pi, v) \mid v \in V_\pi - \{s\}\}$$

- This subgraph represents a tree structure.
- It is called the breadth-first tree.
- It contains a unique path from s to every vertex in V_π.
- All these paths are shortest paths in G.



Depth-First Search (DFS)

DFS Strategy:

First follow one path all the way to its end, before we step back to follow the next path ($u.d$ and $u.f$ are start/finish time for vertex processing)

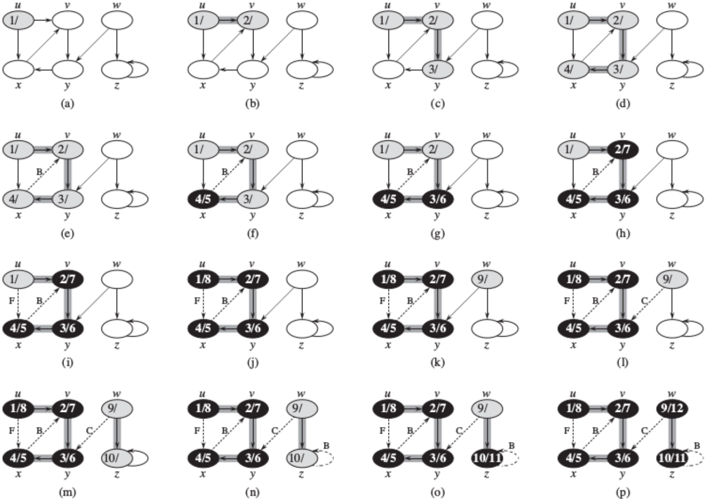
DFS(*G*)

```
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)
```

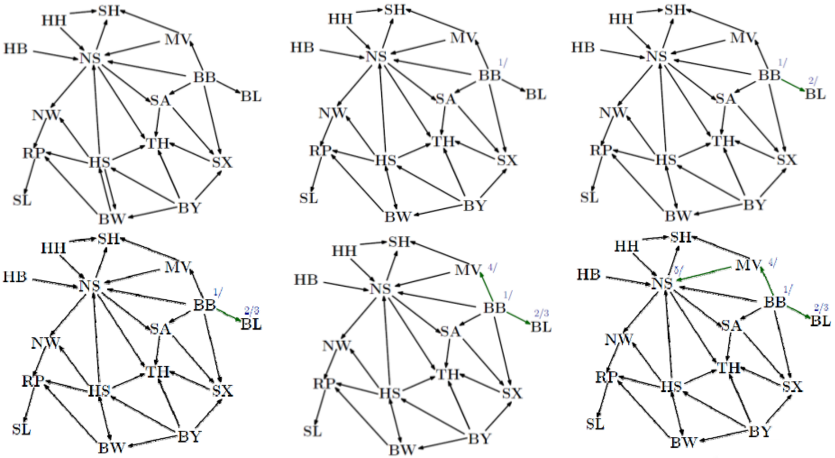
DFS-VISIT(*G*, *u*)

```
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
u.color = BLACK
time = time + 1
u.f = time
```

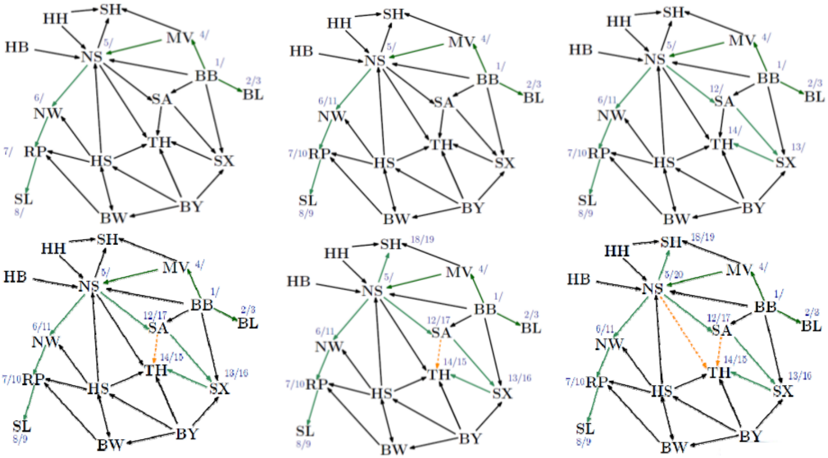
DFS Example



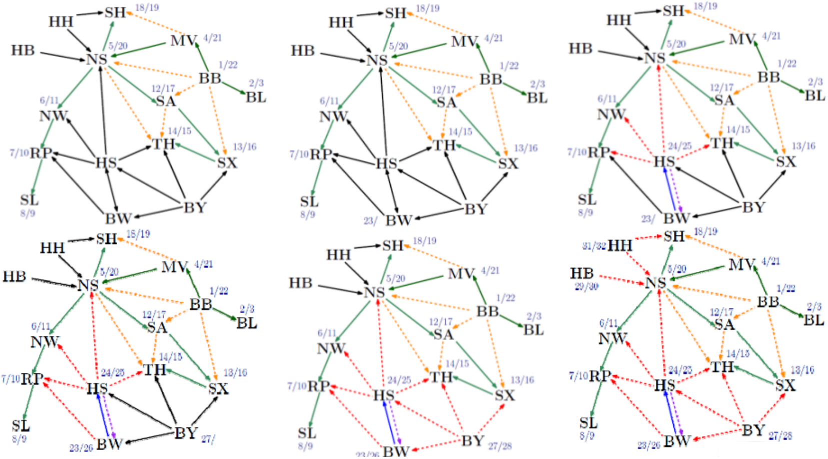
Another DFS Example (1)



Another DFS Example (2)



Another DFS Example (3)



DFS Analysis

```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
u.color = BLACK
time = time + 1
u.f = time
```

Each vertex and each edge is processed once.
Hence, time complexity is $\Theta(|V| + |E|)$.

Edge Types

- Different edge types for (u, v):
  - Tree edges (solid): v is white.
  - Backward edges (purple): v is gray.
  - Forward edges (orange): v is black and $u.d < v.d$
  - Cross edges (red): v is black and $u.d > v.d$
- The tree edges form a forest.
- This is called the depth-first forest.
- In an undirected graph, we have no forward and cross edges.