Graph Traversal/Graph Searching

- Searching a graph means systematically following the edges of the graph so as to visit the vertices of the graph.
- A graph-searching algorithm can discover much about the structure of a graph.
- Many algorithms begin by searching their input graph to obtain this structural information. Other graph algorithms are organized as simple elaborations of basic graph-searching algorithms. Techniques for searching a graph are at the heart of the field of graph algorithms.

Basic Graph Searching Algorithms

1. Breadth-First Search (BFS)

2. Depth-First Search (DFS)

Breadth-First Search (BFS)

- Breadth-first search is a way to find all the vertices reachable from the a given source vertex, s.
- BFS traverse a connected component of a given graph and defines a spanning tree.
- Intuitive Idea:
  - send a wave out from source s.
  - The wave hits all vertices 1 edge from s.
  - From there, the wave hits all vertices 2 edges from s etc.

Strategy of BFS

- BFS starts at a given vertex s which is at level 0.
- First Stage:
  - we visit all the vertices that are at the distance of one edge away from s.
  - When we visit there, we make those vertices as "visited" the vertices adjacent to the start vertex s - these vertices are placed into level 1.
- Second Stage:
  - we visit all the new vertices we can reach at the distance of two edges away from the source vertex s (Note that all such vertices will go via some vertex of level 1).
  - These new vertices, which are adjacent to level 1 vertices and not previously assigned to a level, are placed into level 2, and so on.
- The BFS traversal terminates when every vertex has been visited.

BFS

To keep track of the progress, BFS colors each vertex. Each vertex of the graph is in one of three states:

- Undiscovered
- Discovered but not fully explored
- Fully explored

The state of a vertex u is stored in a color variable as follows:

- $\text{COLOR}[u] = \text{WHITE}$ - for the "undiscovered" state,
- $\text{COLOR}[u] = \text{GRAY}$ - for the "discovered but not fully explored" state,
- $\text{COLOR}[u] = \text{BLACK}$ - for the "fully explored" state.

Algorithm $\mathrm{BFS}(G, s)$

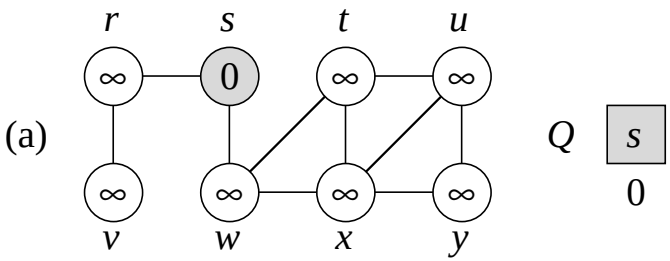
The $\mathrm{BFS}(G, s)$ algorithm develops a breadth-first search tree with the source vertex, s, as its root. The parent or predecessor of any other vertex in the tree is the vertex from which it was first discovered. For each vertex, v, the parent of v is placed in the variable $p[v]$. Another variable, $d[v]$, computed by BFS contains the number of tree edges on the path from s to v. The breadth-first search uses a FIFO queue, Q, to store GRAY vertices.

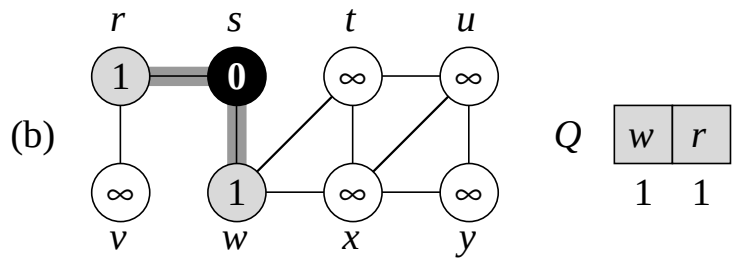
Algorithm 1: BFS(V, E, s)

```
1  for each u ∈ V \ {s} do
2  |    COLOR[u]=WHITE;
3  |    d[u] = ∞;
4  |    π[u] =NIL;
5  end
6  COLOR[s]=GRAY;
7  d[s] = 0;
8  π[s]=NIL;
9  Q = {};
10 ENQUEUE(Q, s);
11 while Q ≠ ∅ do
12 |    u=DEQUEUE(Q);
13 |    for each v ∈ Adj[u] do
14 |    |    if COLOR[v]=WHITE then
15 |    |    |    COLOR[v]=GRAY;
16 |    |    |    d[v] = d[u] + 1;
17 |    |    |    π[v] = u;
18 |    |    |    ENQUEUE(Q, v);
19 |    |    end
20 |    |    DEQUEUE(Q);
21 |    end
22 |    COLOR[u]=BLACK;
23 end
```

Example of BFS

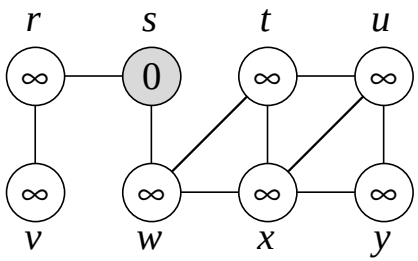
Example of BFS





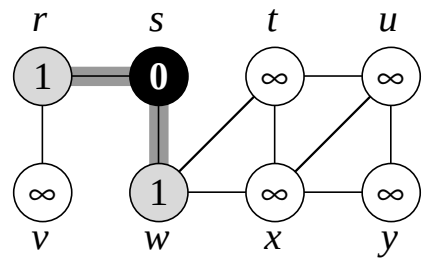
Example of BFS

(a)



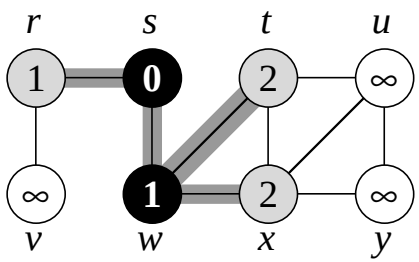
Q | s |
0

(b)



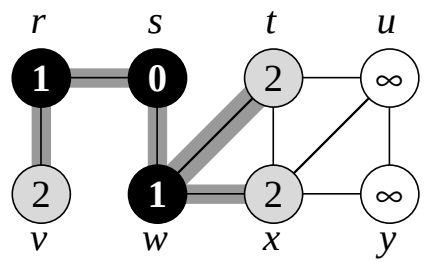
Q | w | r |
1 1

(c)



Q | r | t | x |
1 2 2

(d)

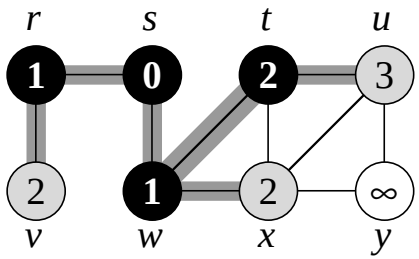


Q | t | x | v |
2 2 2

Example of BFS

Example of BFS

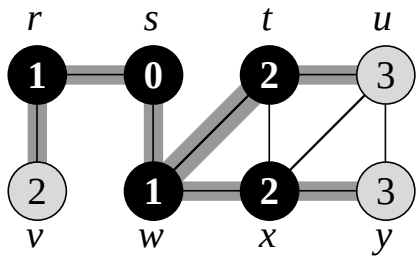
(e)



Q

x	v	u
2	2	3

(f)

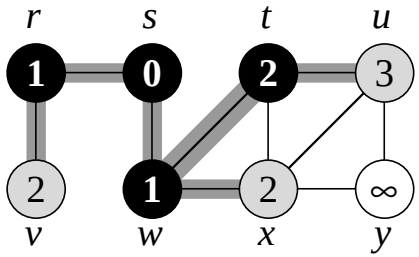


Q

v	u	y
2	3	3

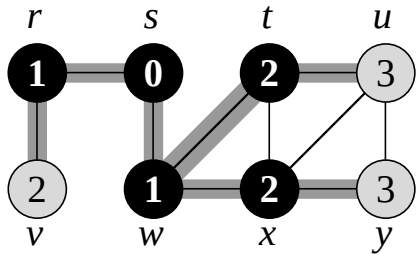
Example of BFS

(e)



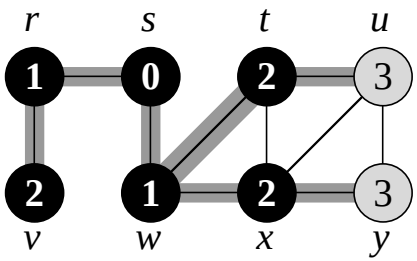
Q | x | v | u |
2 2 3

(f)



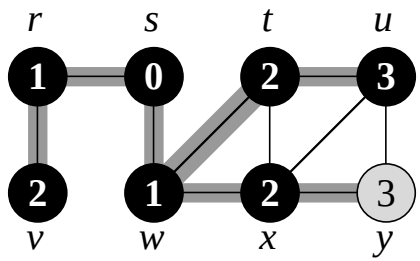
Q | v | u | y |
2 3 3

(g)



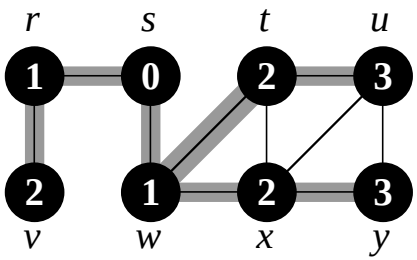
Q | u | y |
3 3

(h)



Q | y |
3

(i)



Q $\emptyset$

Analysis of BFS

- The while-loop in BFS is executed at most $|V|$ times. The reason is that every vertex enqueued at most once. So, we have $O(|V|)$.
- The for-loop inside the while-loop is executed at most $|E|$ times if G is a directed graph or $2 \cdot |E|$ times if G is undirected. The reason is that every vertex dequeued at most once and we examine the edge uv only when u is dequeued. Therefore, every edge examined at most once if directed, at most twice if undirected. So, we have $O(E)$.

Depth-First Search(DFS)

- DFS is also a systematic way to find all the vertices reachable from a source vertex, s.
- Like BFS, DFS traverse a connected component of a given graph and defines a spanning tree.
- Intuitive Idea:
  - It methodically explore every edge. We start over from different vertices as necessary. As soon as we discover a vertex, DFS starts exploring from it (unlike BFS, which puts a vertex on a queue so that it explores from it later).

Strategy of DFS I

- DFS selects a source vertex s in the graph and mark it as "visited"
- Now the vertex s becomes our current vertex. Then, we traverse the graph by considering an arbitrary edge uv from the current vertex u.
- If the edge uv takes to a marked vertex v, then we back down to the vertex u.
- If edge uv takes to an unmarked vertex, then we mark the vertex v and make it our current vertex, and repeat the above computation.
- Sooner or later, we will get to a "dead end," meaning all the edges from our current vertex u takes us to marked vertices.

Strategy of DFS II

- To get out of this, we back down along the edge that brought us here to vertex u and go back to a previously marked vertex v.
- We again make the vertex v our current vertex and start repeating the above computation for any edge that we missed earlier.
- If all of v's edges take to marked vertices, then we again back down to the vertex we came from to get to vertex v, and repeat the computation at that vertex.
- Thus, we continue to back down the path that we have traced so far until we find a vertex that has yet unexplored edges, at which point we take one such edge and continue the traversal.

Strategy of DFS III

- When the DFS has backtracked all the way back to the original source vertex, s, it has built a DFS tree of all vertices reachable from that source.
- If there still undiscovered vertices in the graph then it selects one of them as the source for another DFS tree. The result is a forest of DFS-trees.
- Note that the edges lead to new vertices are called discovery or tree edges and the edges lead to already visited (marked) vertices are called back edges.

To keep track of the progress, DFS also colors each vertex. Each vertex of the graph is in one of three states:

- Undiscovered
- Discovered but not fully explored
- Fully explored

Strategy of DFS IV

The state of a vertex u is stored in a color variable as follows:

- $\text{COLOR}[u] = \text{WHITE}$ - for the "undiscovered" state,
- $\text{COLOR}[u] = \text{GRAY}$ - for the "discovered but not fully explored" state,
- $\text{COLOR}[u] = \text{BLACK}$ - for the "fully explored" state.

Algorithm DFS(V, E)

The DFS forms a depth-first forest comprised of more than one depth-first trees. Each tree is made of edges uv such that u is GRAY and v is WHITE when edge uv is explored.

Algorithm 2: DFS(V, E)

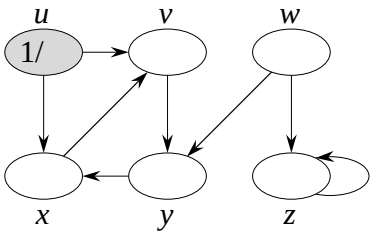
```
1  for each u ∈ V do
2  |    COLOR[u]=WHITE;
3  |    π[u]=NIL;
4  end
5  i = 0;
6  for each u ∈ V do
7  |    if COLOR[u]=WHITE then
8  |    |    DFS-VISIT(u)
9  |    end
10 end
```

Algorithm 3: DFS-VISIT(u)

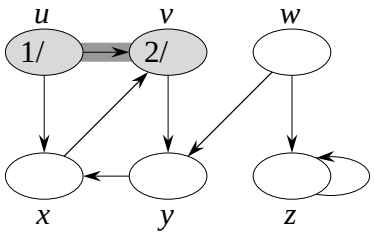
```
1  COLOR[u]=GRAY;
2  i = i + 1;
3  d[u] = i;
4  for each v ∈ Adj[u] do
5  |    if COLOR[v]=WHITE then
6  |    |    π[v] = u;
7  |    |    DFS-VISIT(v);
8  |    end
9  |    COLOR[u]=BLACK;
10 |    i = i + 1;
11 |    f[u] = i;
12 end
```

Example of DFS

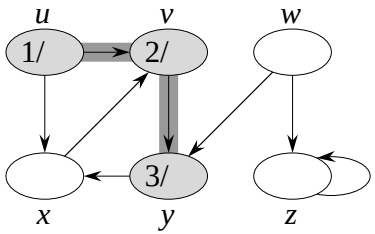
Example of DFS



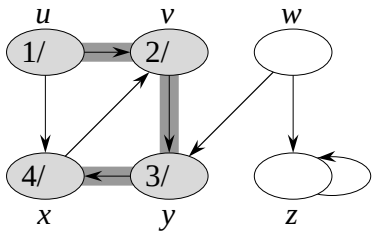
(a)



(b)

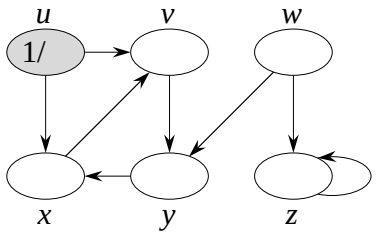


(c)

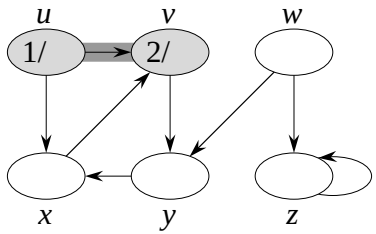


(d)

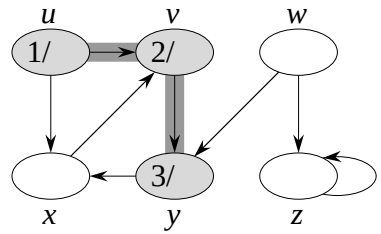
Example of DFS



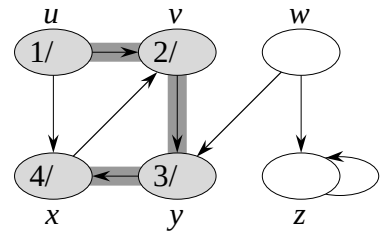
(a)



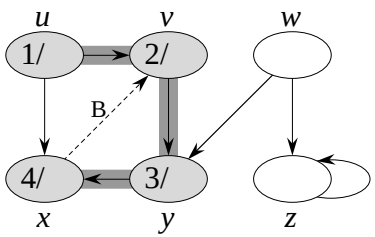
(b)



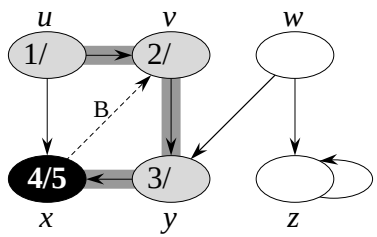
(c)



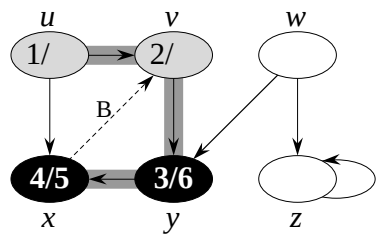
(d)



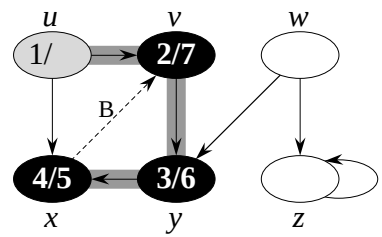
(e)



(f)



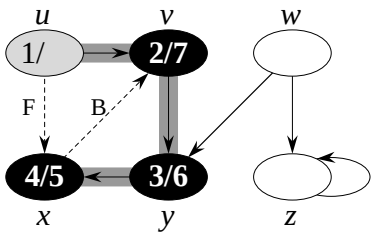
(g)



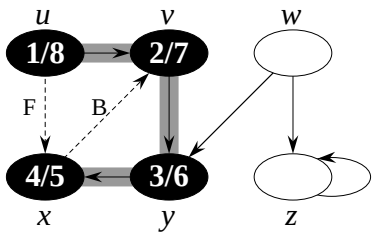
(h)

Example of DFS

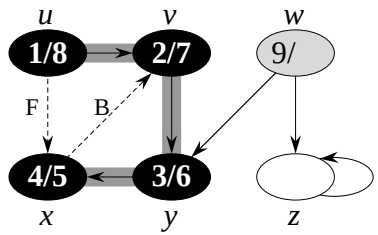
Example of DFS



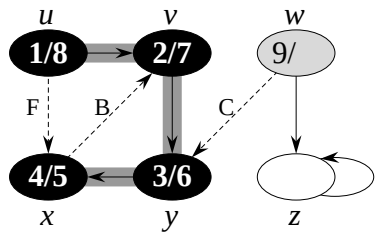
(i)



(j)

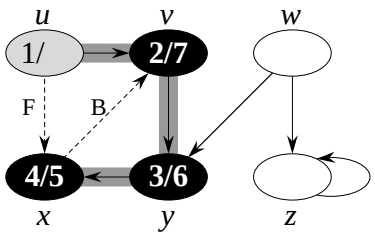


(k)

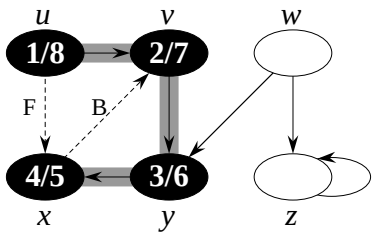


(l)

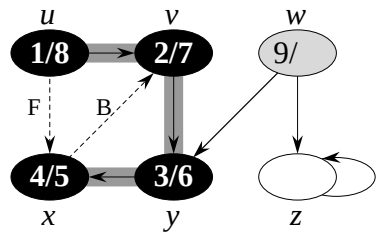
Example of DFS



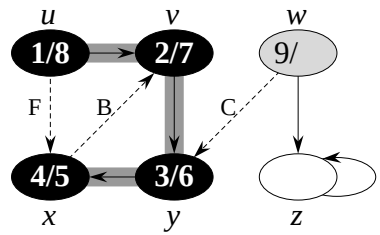
(i)



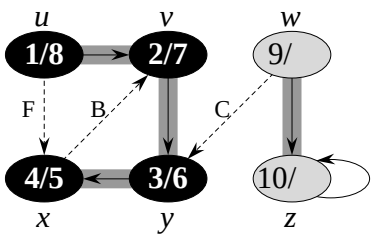
(j)



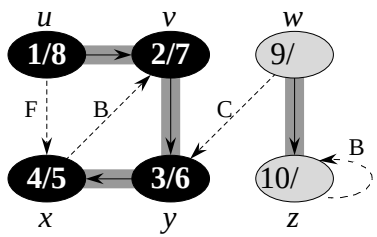
(k)



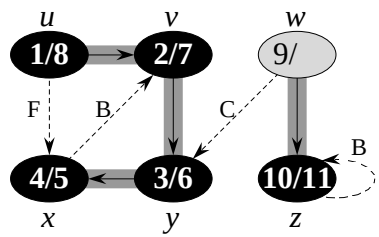
(l)



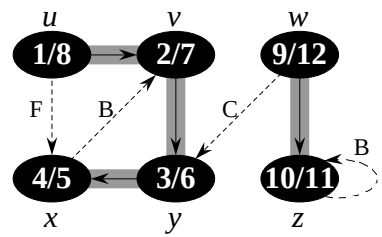
(m)



(n)



(o)



(p)

Analysis of DFS

- The DFS-VISIT is called (from DFS or from itself) once for each vertex in $V[G]$ since each vertex is changed from WHITE to GRAY once.
- The for-loop in DFS-VISIT is executed a total of $|E|$ times for a directed graph or $2 \cdot |E|$ times for an undirected graph since each edge is explored once.
- Moreover, initialization takes $O(|V|)$ time.

Therefore, the running time of DFS is $O(|V| + |E|)$.