Learn DAA : From B K Sharma

TCS-503: Design and Analysis of Algorithms

Graph Algorithms
BFS and DFS

Unit IV

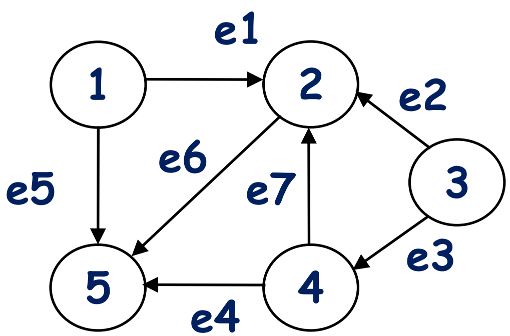
- Graph Algorithms:
  - Elementary Graphs algorithms: BFS and DFS
  - Minimum Spanning Trees
  - Single-Source Shortest Paths
  - All-Pairs Shortest Paths
  - Maximum Flow and
  - Traveling Salesman Problem

Learn DAA : From B K Sharma

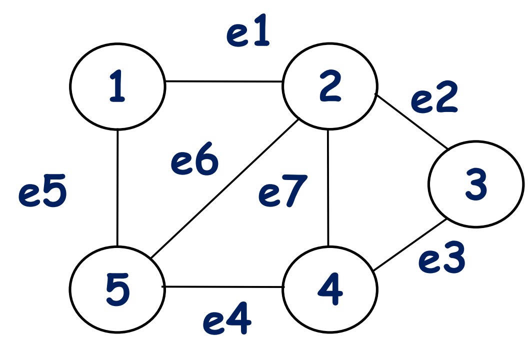
What is graph?

Graph is a non-linear data structure.

A graph G = (V, E) (directed or undirected)



Directed Graph



Undirected Graph

G=({1,2,3,4,5},{e1,e2,e3,e4,e5,e6,e7})

V = set of vertices, E = set of edges

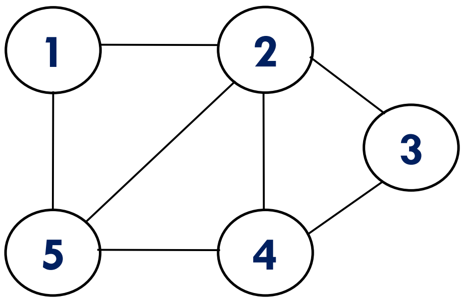
Learn DAA : From B K Sharma

What do you mean by Traversal of a Graph?

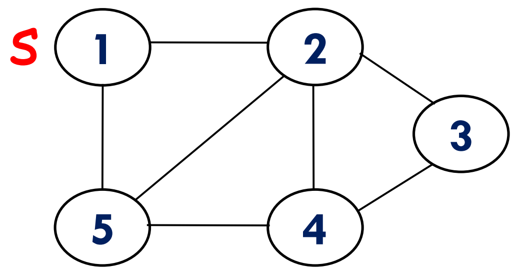
Accessing and processing each vertex of a graph exactly once is called traversal of a graph.

Let source vertex is 1.

1→5 → **2** → **4** → **3**



Graph Traversal Techniques



Breadth First Search
BFS
Implemented using Queue

Source Vertex s is given
d[v]: Distance of v from source vertex s
π[v] – predecessor of v

To keep track of progress use, three Colors: White ,Gray and Black

Initially All vertices are colored white.
When being discovered(i.e. when put in the Queue), becomes gray .
After all its adjacent vertices are discovered, it becomes back.

Depth First Search
DFS
Implemented using Stack

Source Vertex s is given
time=Global Time
d[v] = discovery time
f[v] = finishing time (done with examining v's adjacency list)

To keep track of progress use, three Colors: White ,Gray and Black

Initially All vertices are colored white. When being discovered(i.e. when put in the Stack, becomes gray . After all its adjacent vertices are discovered, it becomes black.

Learn DAA : From B K Sharma

white: undiscovered
gray: discovered
black: finished

BFS(G,s)

```
for each vertex u in V[G] – {s}
        do color[u] ← white
            d[u] ← ∝
            π[u] ← nil
color[s] ← gray
d[s] ← 0
 π[s] ← nil
Q ← Φ
enqueue(Q,s)
while Q ≠ Φ
        do u ← dequeue(Q)
            for each v in Adj[u]
                do if color[v] = white
                    then color[v] ← gray
                        d[v] ← d[u] + 1
                        π[v] ← u
                        enqueue(Q,v)
            color[u] ← black
```

Lines 1–4: Paint every vertex white. Set d[u] to infinity for every vertex u. Set Parent of every vertex to NIL

Line 5: Paint the source vertex s gray (why?)

Line 6: Initialize d[s] to 0

Line 7: Set parent of s to NIL

Line 8: Initialize Q to Φ (empty)

Line 9: Enqueue s in Q

Line 10: while loop iterates as long as there remains gray vertices.

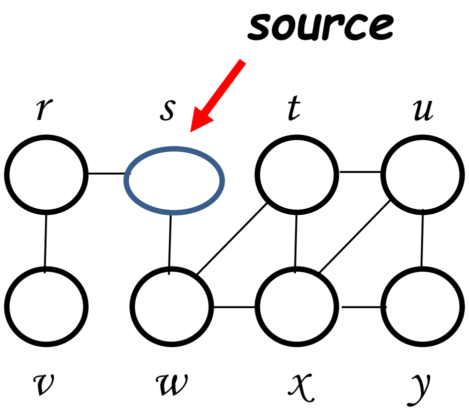
Line 11: Removes the gray vertex u from Q

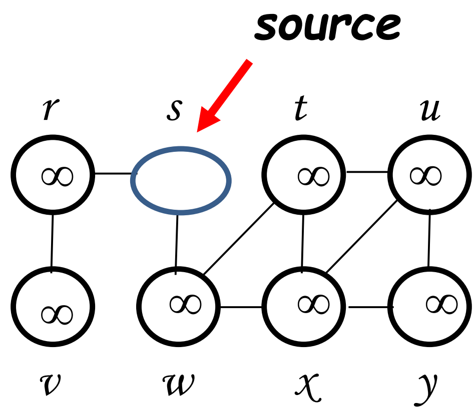
Lines 12–17: The for loop considers the each vertex v in the adjacency list of u. If v is white, then it has not yet been discovered, discover it. It is first grayed, Then d[v] is set to d[u] + 1.Then u is recorded as its parent. Finally, v is placed at the tail of Q.

Line 18: When all the vertices on u's adjacency list have been examined, blacken u.

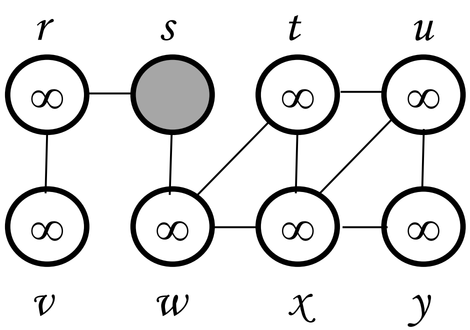
Q: a queue of discovered vertices
color[v]: color of v
d[v]: distance from s to v
π[u]: predecessor of v

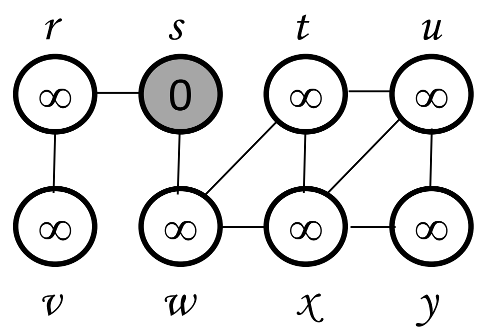
Learn DAA : From B K Sharma





π[r]=NIL
π[t]=NIL
π[u]=NIL
π[v]=NIL
π[w]=NIL
π[x]=NIL
π[y]=NIL

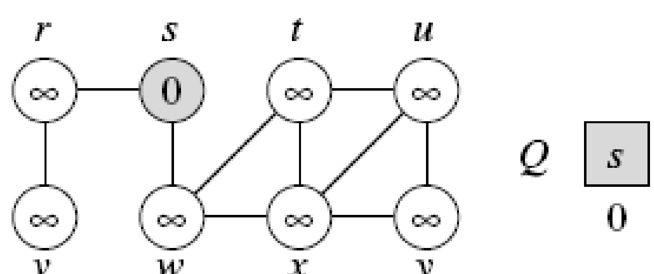




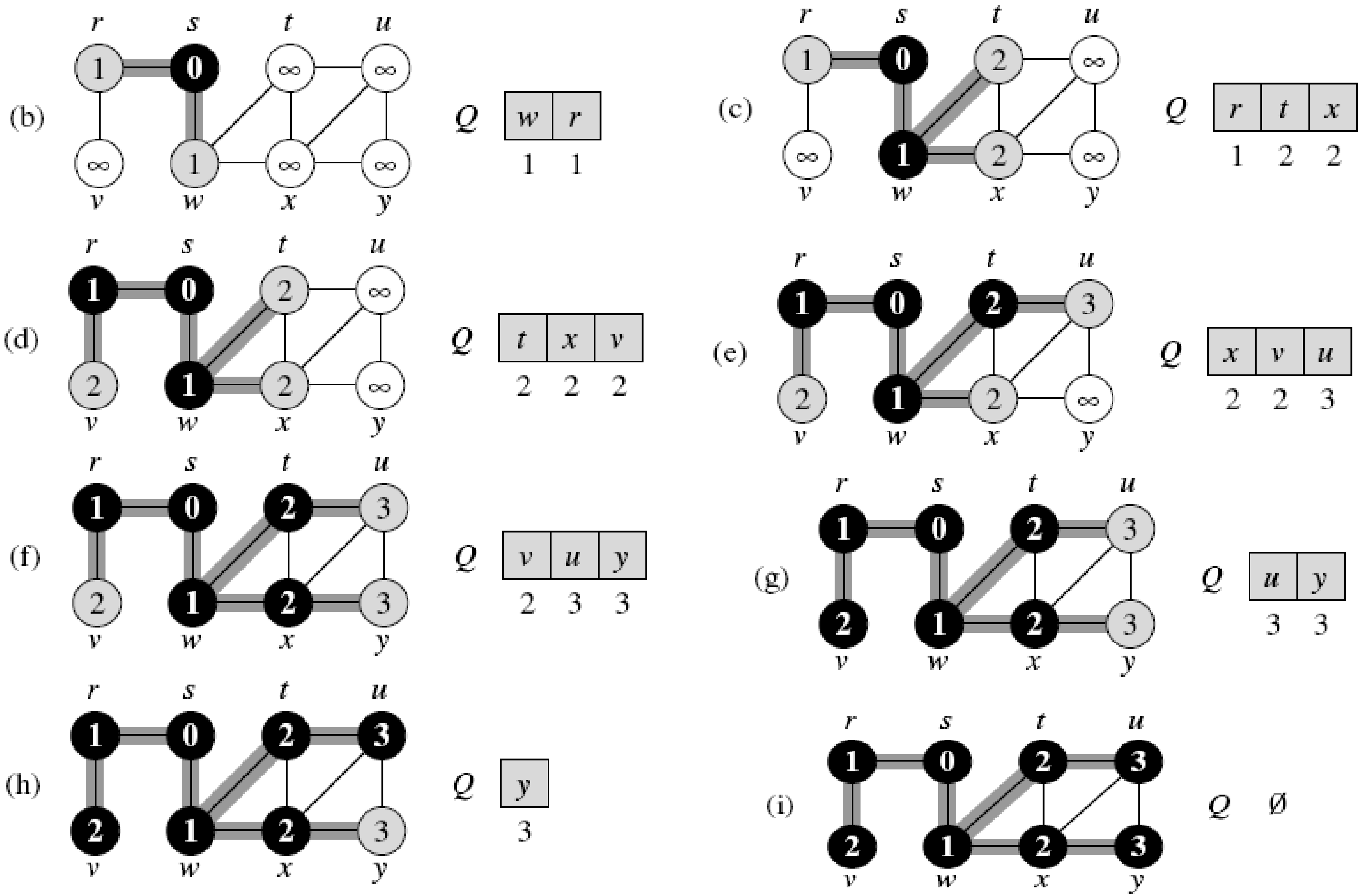
π[s]=NIL

Q: $\boxed{\varnothing}$

(a)



Learn DAA : From B K Sharma

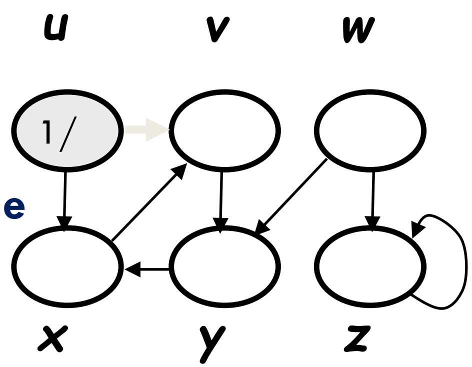


Sequence of vertices traversed: s, w, r, t, x, v, u, y

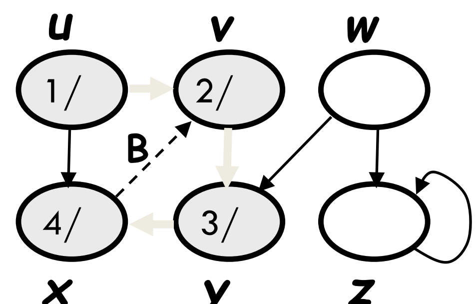
Depth First Search
DFS
Edge Classification

Tree edge (reaches a WHITE vertex):

Tree Edge

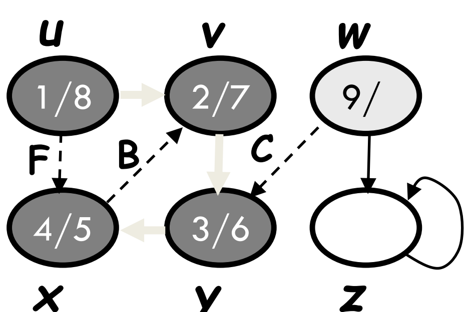


Back edge (reaches a GRAY vertex):

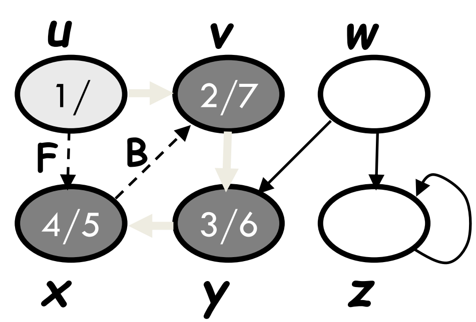


– Self loops (in directed graphs) are also back edges

Cross edge (reaches a BLACK vertex & d[u] > d[v]):



Forward edge (reaches a BLACK vertex & d[u] < d[v]):



Learn DAA : From B K Sharma

Depth First Search
DFS

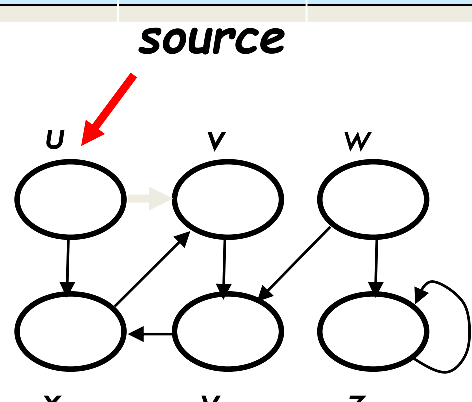
DFS(G)

```
for each vertex u ∈ V[G]
      do color[u] ← white
            π[u] ← NIL
time ← 0
for each vertex u ∈ V[G]
      do if color[u] = white
                then DFS-Visit(u)
```

Paint all vertices white and initialize their predecessor field to NIL

Set the global time counter to 0.

Check each vertex in V in turn, and when a white vertex is found, visit it using DFS-VISIT.



$\pi[u]$=NIL
$\pi[v]$=NIL
$\pi[w]$=NIL
$\pi[x]$=NIL
$\pi[y]$=NIL
$\pi[z]$=NIL

time=0

Depth First Search
DFS

DFS-Visit(*u*)

```
color[u] ← GRAY  ▽ White vertex u has been discovered
time ← time + 1
d[u] ← time                ➢Discovery Time
for each v ∈ Adj[u]        ➢ Explore edge (u,v)
    do if color[v] = WHITE
        then π[v] ← u
            DFS-Visit(v)
color[u] ← BLACK    ▽ Blacken u;  it is finished.
f[u] ← time ← time + 1
```

Examine each vertex v adjacent to u, and recursively visit v if it is white

Record the finishing time in f[u].

Sequence of vertices traversed: x, y, v, u, z, w