Elementary Graph Algorithms

Chapter 22

Graph Representation

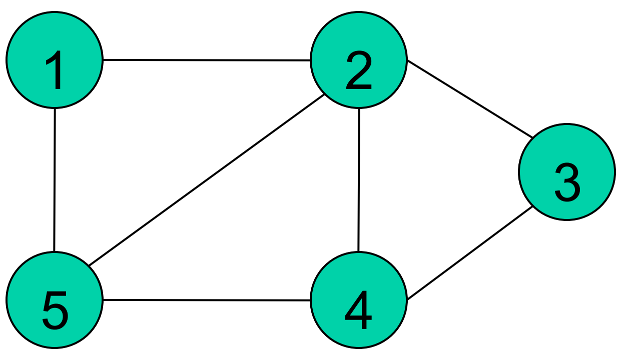
- Given a graph G = (V, E)
- The graph may be directed or undirected
- There are two common ways to represent graphs for algorithms:
  - Adjacency lists.
  - Adjacency matrix.

Running Times

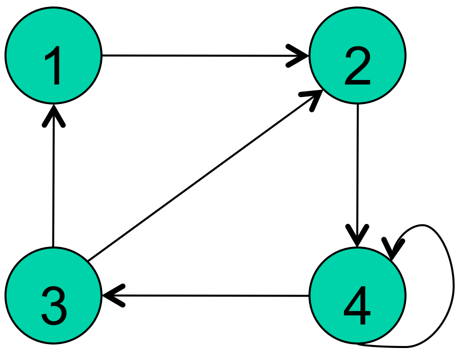
- We will be talking about the running time of graph algorithms in terms of both Vertices |V| and Edges |E|
- We can remove the cardinality when in asymptotic notation
  - Example: O(V + E)

Adjacency Lists

- Array Adj of |V| lists, one per vertex
- Vertex u's list has all vertices v such that $(u,v) \in E$
- Example:



Example

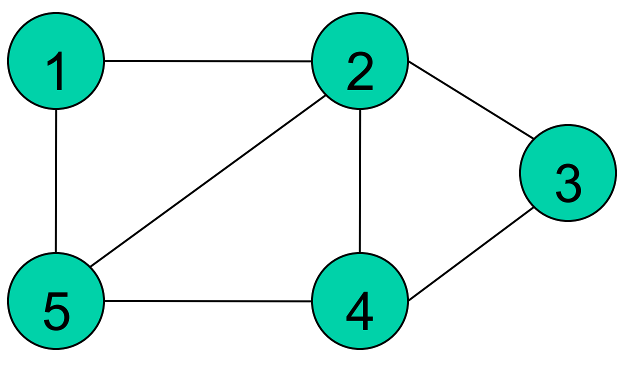


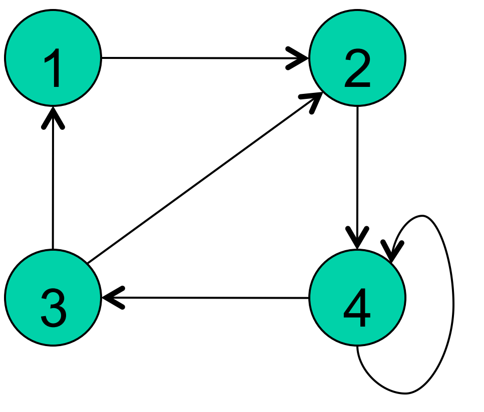
- Space:
- Time to list all vertices adjacent to u:
- Time to determine if (u,v) is an edge:

Adjacency Matrix

- |V| x |V| matrix A = (a_{ij})

$$a_{ij} = \begin{cases} 1 & if\,(i,j) \in E \\ 0 & otherwise \end{cases}$$





Adjacency Matrix

- Space:
- Time to list all vertices adjacent to u:
- Time to determine if (u,v) is an edge:

- What about weighted graphs?

Breadth-First Search

- Input: Graph G = (V, E), either directed or undirected, and source vertex s is in V.
- Output:
  - d[v] = distance (smallest # of edges) from s to v, for all v in V.
  - π[v] = u such that (u,v) is last edge on shortest path s->v
- u is v's predecessor
- Set of edges {(π[v],v): v ≠ s} forms a tree

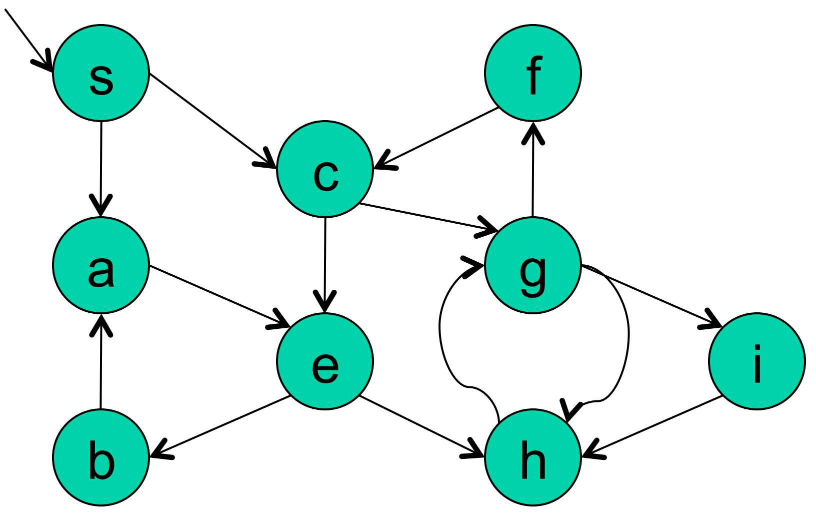
Breadth-First Search

- Idea: Send a wave out from s.
  - First hits all vertices 1 edge from s.
  - From there, hits all vertices 2 edges from s.
  - Etc.
- Use FIFO queue Q to maintain wavefront.
  - v is in Q if and only if wave has hit v but has not come out of v yet

BFS(G, s)

```
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
```

Example



Breadth-First Search

- Will breadth-first search reach all vertices?
- Time = O(   )

Depth-First Search

- Input: G = (V, E), directed or undirected. No source vertex given.
- Output: 2 timestamps on each vertex:
  - d[v] = discovery time
  - f[v] = finishing time
  - π[v] = u such that (u,v) is last edge on shortest path s->v

DFS(G)

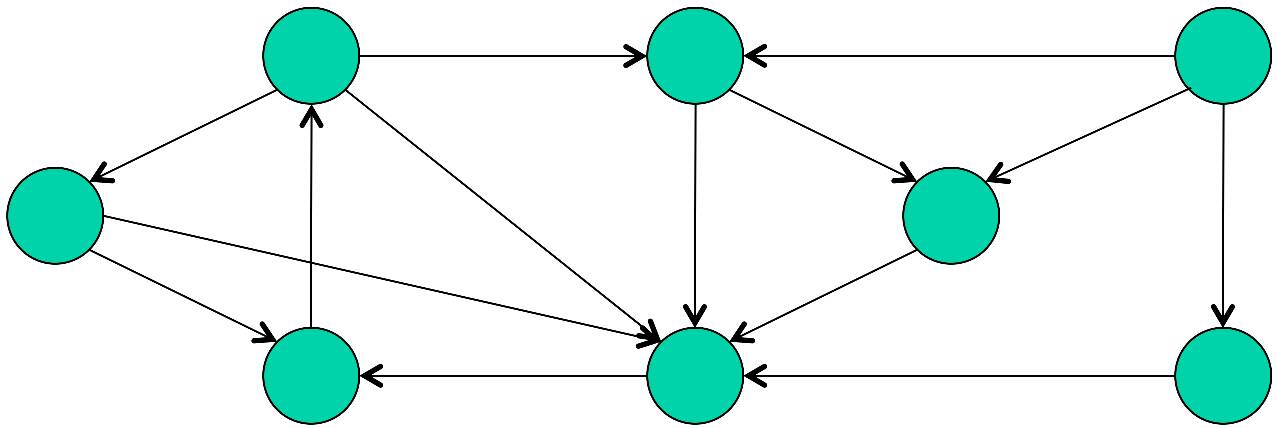
DFS(G)

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

DFS-VISIT(G, u)

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

Example



Depth-First Search

- Running Time =

Classification of Edges

- Tree edge:
- Back edge:
- Forward edge:
- Cross edge:

Your Turn

- Solve exercise 22.3-2 on page 547