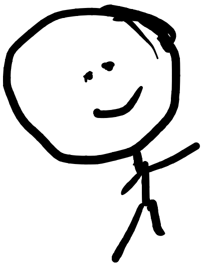


CSC 226 Summer 2023 Lab 4: BFS/DFS review

Ali Mortazavi

May 31, 2023

1 Intro

In this lab, we'll be going over DFS/BFS. Since there are tons of great online resources out there, my presentation will heavily rely on them.

2 Common Adjectives for Graphs

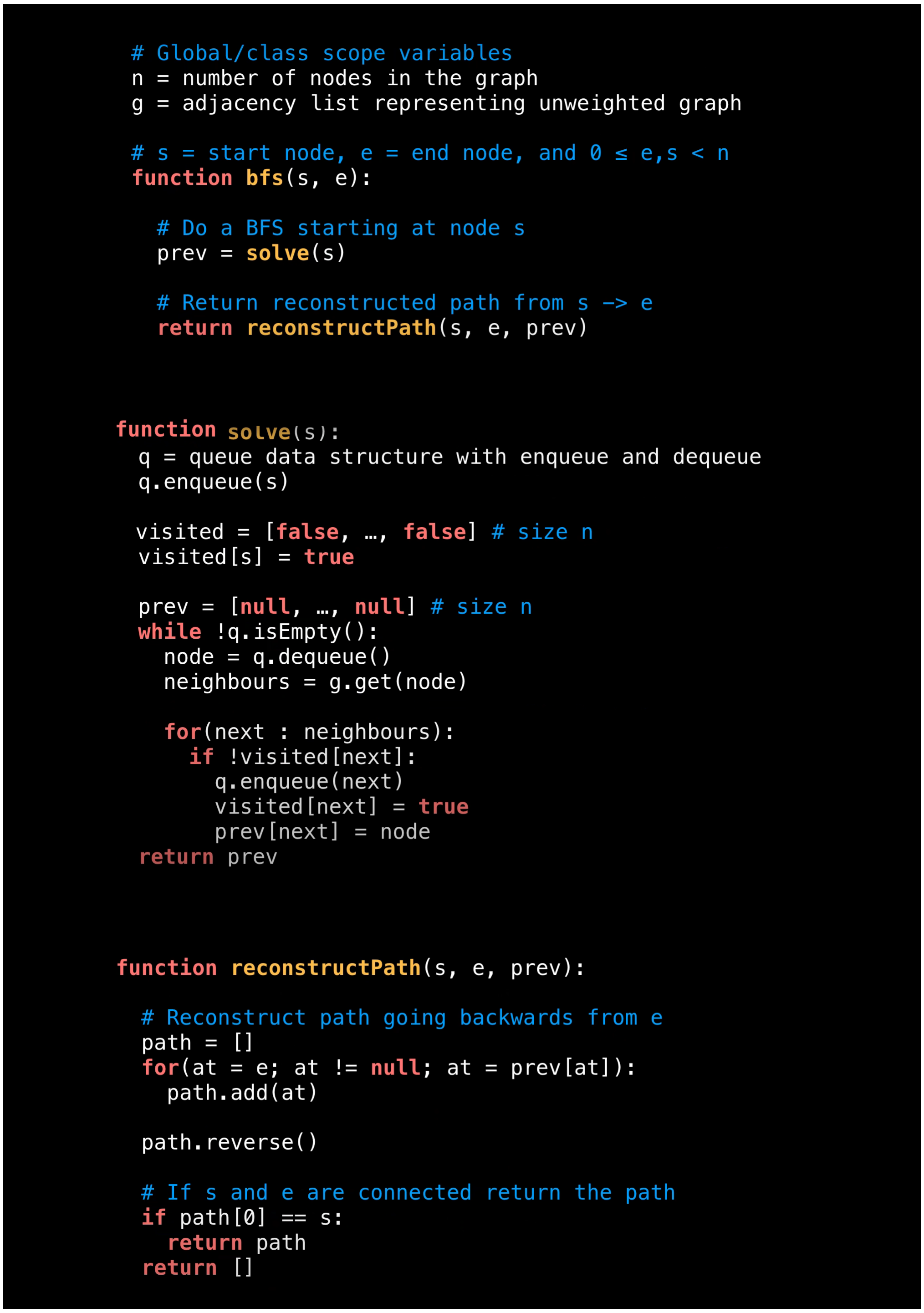
- undirected graph
- directed/undirected graph
- weighted/unweighted graph
- acyclic graph
  1. Usually we use this adjective for *directed* graph. DAG refers to "Directed Acyclic Graph"
  2. Undirected Acyclic Graph is either a single tree or a forest (collection of trees)
- connected graph

3 Graph Representation

1. Adjacency List
2. Adjacency Matrix

Link

4 BFS

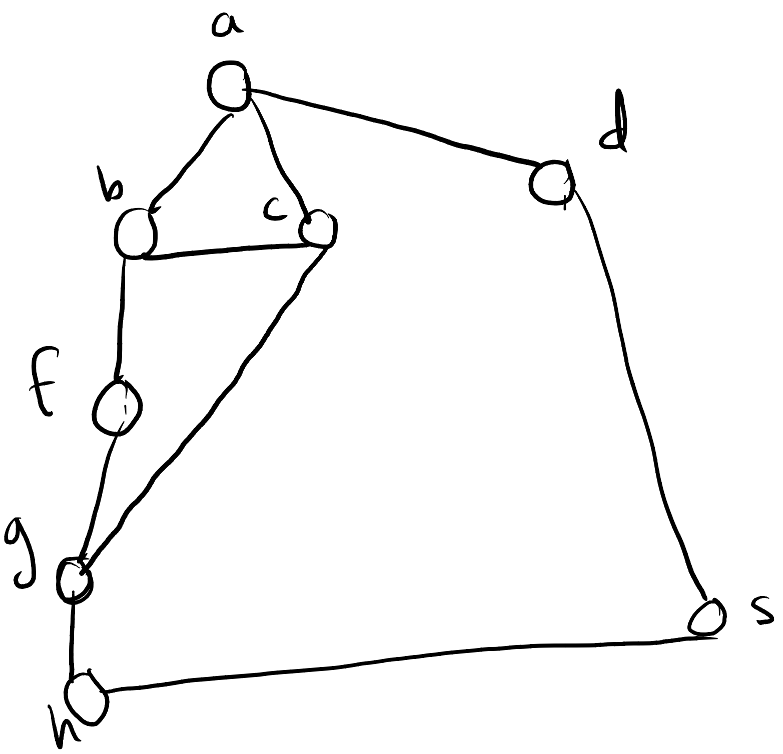


```
# Global/class scope variables
n = number of nodes in the graph
g = adjacency list representing unweighted graph

# s = start node, e = end node, and 0 ≤ e,s < n
function bfs(s, e):

  # Do a BFS starting at node s
  prev = solve(s)

  # Return reconstructed path from s -> e
  return reconstructPath(s, e, prev)



function solve(s):
  q = queue data structure with enqueue and dequeue
  q.enqueue(s)

  visited = [false, …, false] # size n
  visited[s] = true

  prev = [null, …, null] # size n
  while !q.isEmpty():
    node = q.dequeue()
    neighbours = g.get(node)

    for(next : neighbours):
      if !visited[next]:
        q.enqueue(next)
        visited[next] = true
        prev[next] = node
  return prev



function reconstructPath(s, e, prev):

  # Reconstruct path going backwards from e
  path = []
  for(at = e; at != null; at = prev[at]):
    path.add(at)

  path.reverse()

  # If s and e are connected return the path
  if path[0] == s:
    return path
  return []
```

BFS algorithm
Nice Video on DFS
Nice Video on BFS

Question 1. *Run BFS on this graph and show the diagram.*

Graph G:



Adjacency list:

a : b , c, d

b : a, c, f

c : a, b, g

d : a, s,

f : b , g

g , f, c , h

h : g, s

s: d, h

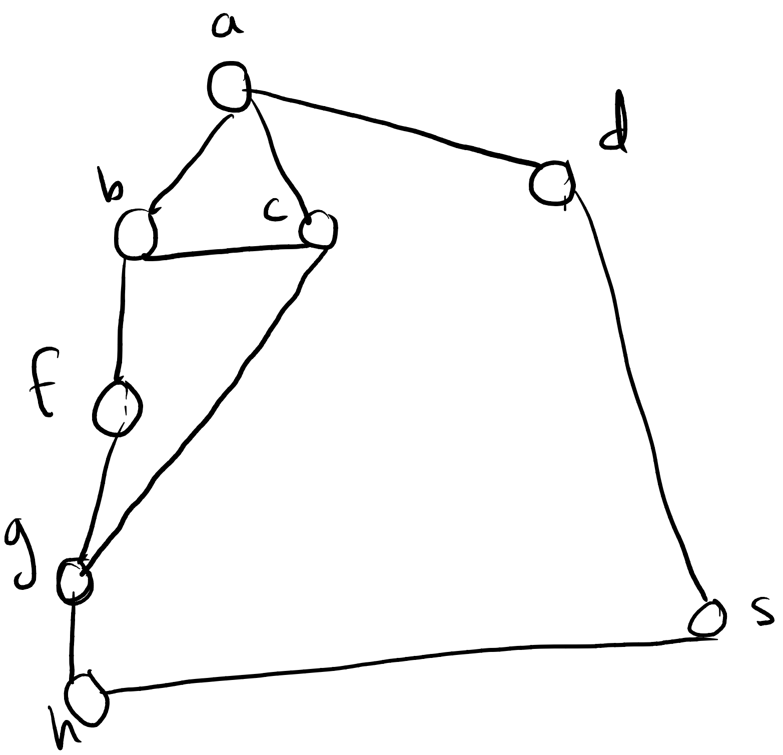
5 DFS and induced traversal tree

You probably learn this from CSC225.

Youtube link A Note

Question 2. *Run DFS on this graph and show the induced traversal tree.*

Graph G : ↔ (Same graph)



Adjacency list:

a : b, c, d

b : a, c, f

c : a, b, g

d : a, s,

f : b, g

g : f, c, h

h : g, s

s : d, h