CS301
Data Structure and Algorithms
Lecture 12: Applications of BFS and DFS

Pandav Patel
Assistant Professor

Computer Engineering Department
Dharmsinh Desai University
Nadiad, Gujarat, India

Objective

- To learn applications of BFS and DFS traversals

Overview

Single source shortest path for unweighted graph

- BFS can be used to find minimum distance of all nodes from a given node when graph is not weighted. It can be done by keeping the counts of hops required to reach nodes in the graph from a starting node.
- Time compexity: $O(V + E)$

Connectivity in undirected graph (Terminology)

- An undirected graph is said to be connected if there is a path between every pair of vertices
- An undirected graph is said to be disconnected if there exist two vertices which are not endpoints of any path in G
- A connected component is a maximal connected subgraph of an undirected graph
- An undirected graph has only one connected component if it is connected graph
- An undirected graph has more than one connected components if it is disconnected graph

Is given undirected graph connected?

```
count ←0
For each vertex v in the graph do
    If v is unvisited then
        count ←count + 1
        DFS(v) // can use BFS as well
If count = 1 then
    Write("It is connected graph")
Else
    Write("It is not connected graph")
```

- Time complexity? $O(V + E)$

Count connected components in undirected graph

```
count ←0
For each vertex v in the graph do
    If v is unvisited then
        count ←count + 1
        DFS(v) // can use BFS as well
return count
```

- Time complexity? $O(V + E)$

Connectivity in directed graph (Terminology)

- A directed graph is said to be strongly connected if for each pair of vertices u and v, there exists a directed path from u to v and v to u
- A directed graph is said to be semiconnected (a.k.a. unilaterally connected) if for each pair of vertices u and v, there exists a directed path from u to v or v to u
- A directed graph is said to be weakly connected if replacing all of its directed edges with undirected edges produces a connected (undirected) graph
- All strongly connected graphs are semiconnected and all semiconnected graphs are weakly connected. But reverse is not true.
- The strong components are the maximal strongly connected subgraphs of a directed graph (Find number of strongly connected components)

Is given directed graph weakly connected?

- How to find out if a given directed graph is weakly connected or not?
  - Convert all the directed edges to undirected edges. Apply BFS/DFS on resultant undirected graph, if it is connected (undirected) then original directed graph is weakly connected.
  - Time complexity? $O(V + E)$

Is given directed graph semiconnected?

- One solution is to try BFS/DFS from each vertex until we find a vertex from where there exists a path to all other vertices. If no such vertex exist then graph is not semiconnected.
- How can we prove that above solution works?
- Can't there be a semiconnected graph where no vertex has path to all the vertices?
- To prove that the proposed solution works, we need to prove that in a semiconnected graph there must exist a vertex from which all vertices are reachable.

Is given directed graph semiconnected? (cont...)

- Proof of the proposed approach
  - Consider a vertex Vx from which maximum vertices are reachable
  - If not all vertices are reachable from Vx then there must exist a vertex Vy which is not reachable from Vx
  - In that case path must exist from Vy to Vx as graph is semiconnected
  - But if there exists a path from Vy to Vx then from Vy we can reach Vx and all other vertices reachable from Vx. And Vx would not be vertex from which maximum vertices are reachable.
  - We can apply this argument repeatedly until we find a vertex from which all the vertices are reachable.
  - Time complexity? $O(V^2 + V * E)$
- For better solution click here and check this too

Is given directed graph strongly connected?

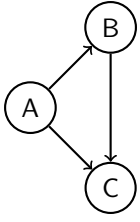
- Naive solution is to run BFS/DFS from all the vertices until we find a vertex from which not all vertices are reachable.
  - If such a vertex exists then it is not strongly connected otherwise it is strongly connected
  - Time complexity? $O(V^2 + V * E)$
- Another solution is
  1. Randomly pick a vertex v
  2. Run BFS/DFS starting from vertex v.
     - If not all vertices are reachable then it is not strongly connected, stop here.
  3. Reverse the direction of all edges and then run BFS/DFS from vertex v
     - If not all vertices are reachable then it is not strongly connected, stop here.
     - If all vertices are reachable the it is strongly connected.

Is given directed graph strongly connected? (cont...)

- Proof of the proposed approach
  - If all vertices are reachable from v in a reversed graph, then there must exist path from each vertex to v in original graph. So if we randomly pick vertex u then there must exist path from u to v in original graph.
  - In second step (DFS/BFS on original graph), we proved that all vertices are reachable from vertex v. So if we randomly pick vertex u then there must exist path from v to u in original graph.
  - If we randomly pick two vertices $v1$ and $v2$ then there must exist following paths
    - $v1 \rightarrow v$
    - $v \rightarrow v2$
    - $v2 \rightarrow v$
    - $v \rightarrow v1$
  - Based on above four paths, we can argue that there must exist paths from $v1$ to $v2$ and $v2$ to $v1$
- Time complexity? $O(V + E)$

Detecting a cycle in a graph (directed / undirected)

- Cycle can be easily detected in undirected graph. During DFS/BFS, if there exists an edge from vertex v to already visited vertex (except immediate parent of v in DFS/BFS tree) then graph contains a cycle. (Assumption: No parallel edges)
  - Is same true for undirected graph? No. Why?
  - Consider following example

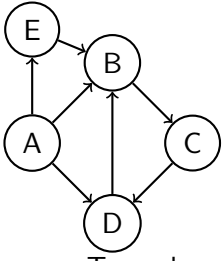


  - If DFS starts from A and then goes to C, backtracks from C to A and then from A reaches B. From B, vertex C is reachable and C is already visited. But given graph does not have any cycle.

Detecting a cycle in a directed graph

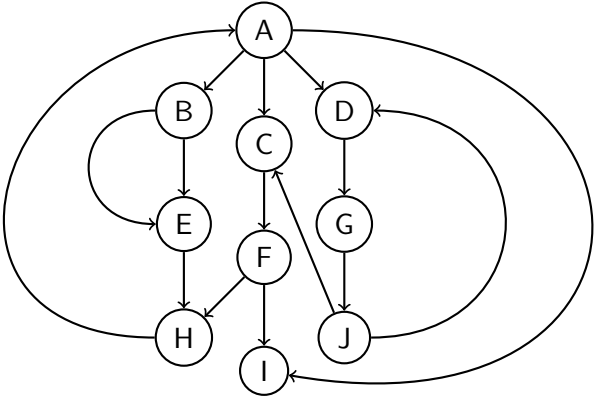
- Vertex coloring during recursive DFS can be used
  - All vertices are colored white at the beginning
  - Color of the vertex will change from white to grey when it is visited first time
  - Color of the vertex will change from grey to black when that vertex and all its adjacent vertices are completely processed (during back tracking).
- At any time during DFS, if there is a an edge from grey vertex to grey vertex (*back edge* - An edge from a vertex to its ancestor in DFS tree), then the graph contains a cycle. Why?
- Edges from grey to white vertices are called *tree edges* and are part of DFS tree
- Try to understand why this approach works
  - Why does an edge from grey vertex to black vertex does not mean a cycle? Because it will not be a *back edge*, but will be *cross edge* or *forward edge*. Why?
    - Grey vertex is ancestor of black vertex in DFS tree - forward edge
    - Grey vertex is not an ancestor of black vertex in DFS tree - cross edge

Detecting a cycle in a directed graph (cont...)



- If DFS of given graph produces ABCDE, then identify tree edges, back edges, forward edges and cross edges.
  - Tree edges: A →B, B →C, C →D, A →E
  - Back edges: D →B
  - Forward edges: A →D
  - Cross edges: E →B
- How about ADBCE and AEBCD? No cross edge in second case.
- Tree edges, back edges, forward edges and cross edges can differ based on starting vertex and sequence in which neighbours are visited.
- At any time, each grey vertex have active stack frame on call stack. All grey vertices are sequentially connected from bottom to top of stack frame.
- Time complexity? O(V + E)
- Can BFS be used to detect a cycle in directed graph? Check here

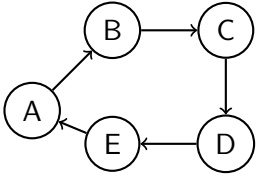
Detecting a cycle in a directed graph (cont...)

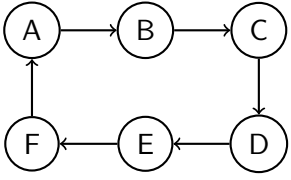


- For DFS traversal resulting in ABEHCFIDGJ, identify tree edges, back edges, forward edges and cross edges.
- F $\rightarrow$H is not forward edge and J $\rightarrow$C is not back edge
- Is back edge guaranteed if cycle is present? Yes. Logical explanation?
- For disconnected graph, apply this algo on each component

Is given graph bipartite?

- What is bipartite graph?
  - If vertices can be divided into two sets such that there is no edge between two vertices of the same set then it is called bipartite graph

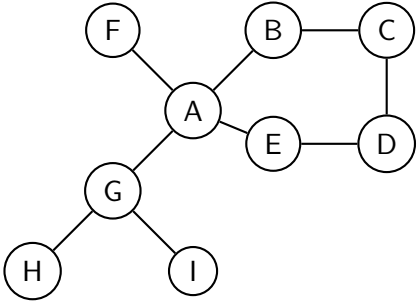




Is given graph bipartite? (cont...)

- Solution
  - Use two colors. Say red and blue.
  - Start BFS from a vertex and color it red
  - During BFS neighbours of red vertex should be colored blue and neighbours of blue vertex should be colored red
  - If during BFS you find already colored neighbour with same color then it is not bipartite. Stop there.
  - If you are able to color all the vertices then it is bipartite. Red and blue vertices represent two separate sets without any edge between two vertices of the same set
- For disconnected graph, apply this to each component
- Solution works for undirected as well as directed graphs
- Either BFS/DFS can be used. Time complexity? O(V + E)

Is given undirected graph biconnected?



- What is *cut vertex* (a.k.a. articulation point)?
  - A vertex in undirected graph whose removal will split the graph into two or more components
- What is biconnected graph?
  - A connected graph without any cut vertex.
  - In other words, for each pair of vertices there exist atleast two vertex-disjoint paths. How to argue this?
- Is given graph biconnected? If no, which vertex is the *cut vertex*? How many are there?
- Can you make given graph biconnected by adding more edges?

Is given undirected graph biconnected? (cont...)

- How to check if given undirected graph is biconnected or not?

  For each vertex v

  Remove v from original graph and check if resultant graph is still connected

  If resultant graph is no longer connected then stop, original graph is not biconnected

  If there does not exist a vertex whose removal splits original graph into two or more components then it is biconnected graph
- Time complexity? $O(V^2 + V * E)$
- Is there better approach? Yes. Search yourself.