Graph search algorithms

Pramesh Kumar

IIT Delhi

February 15, 2024

Outline

Search algorithms

BFS

DFS

Topological ordering

Search algorithms

They attempt to find all the nodes with a desired property.

Example(s).

1. Finding all nodes in the network reachable using a directed path from a given node.
2. Finding all nodes in the network that can reach a given node along a directed path.
3. Identify all connected components of a network.
4. Determining whether a given network is bipartite.
5. Identify a directed cycle in the network, otherwise if the network is acyclic, determine the topological order of nodes ($order(i) < order(j), \forall (i,j) \in A$) .

Search algorithm

Input: Graph $G(N, A)$ and source node $s \in N$
procedure $\textsc{Search}(G, s)$
    $mark(i) \leftarrow \texttt{FALSE}, \forall i \in N \backslash \{s\}; mark(s) \leftarrow \texttt{TRUE}$
    $pred(i) \leftarrow \texttt{NA}, \forall i \in N \backslash \{s\}; pred(s) \leftarrow 0$
    $order(i) \leftarrow \texttt{NA}, \forall i \in N \backslash \{s\}; order(s) \leftarrow 0$
    $Q \leftarrow \{s\}$
    while $Q \neq \phi$ **do**
        Remove "next" node i from Q
        for $j \in \delta(i)$ **do**
            if $mark(j) == \texttt{FALSE}$ **then**
                $mark(j) \leftarrow \texttt{TRUE}$
                $pred(j) \leftarrow i$
                $order(j) \leftarrow order(i) + 1$
                $Q \leftarrow Q \cup \{j\}$
            end if
        end for
    end while
end procedure

Search algorithm

- Above algorithm marks all the nodes which are reachable from s along a directed path
- The directed path can be obtained by tracing the predecessors $pred$
- When the algorithm terminates $pred$ helps in obtaining the search tree.
- $order$ helps keep the sequence in which we mark nodes.

Proposition

The search algorithm runs in $O(|N| + |A|)$ time.

Proof.

Lines 3-5 in $O(|N|)$ time. Adding (Line 14) and removing (Line 8) can be performed in $O(1)$ time. Because the procedure scans the adjacency list of each node only when it is removed from Q, it scans each adjacency list at most once. Since the sum of the lengths of all $|N|$ adjacency lists is $\Theta(|A|)$, the total running time of scanning adjacency lists is $O(|N| + |A|)$. Thus, the total running time of the search algorithm is $O(|N| + |A|)$. □

Outline

Search algorithms

BFS

DFS

Topological ordering

Breadth-first search

- If we maintain Q as a queue in the search algorithm, we remove the nodes from the front and add them to the rear.
- This way the algorithm selects nodes in First-In-First-Out (FIFO) order.

Definition (Shortest path). Define $\mathfrak{d}(s, j)$ be the shortest path distance from s to j as the minimum number of links in any path from s to j. If there does not exists any path, then $\mathfrak{d}(s, j) = \infty$. We call the path of length $\mathfrak{d}(s, j)$ as the shortest path from s to j.

Theorem

In the breadth-first search tree, the path from s to any node i is a shortest path.

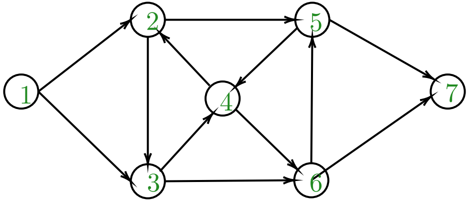
Outline

Depth-first search

- If we maintain Q as a stack in the search algorithm, we remove the nodes from the front and add them to the front.
- This way the algorithm selects nodes in Last-In-First-Out (LIFO) order.
- It searches "deeper" in the graph whenever possible.
- Unlike breadth-first search, its predecessor graph might contain several trees.

Task

Apply BFS and DFS to the following network.



Outline

Search algorithms

BFS

DFS

Topological ordering

Directed acyclic graphs and topological ordering

Definition (Directed acyclic graph (DAG)). A directed graph is DAG if does not contain any directed cycle.

Definition (Topological ordering). We say that a labeling $order$ of a graph is topological ordering if $\forall (i,j) \in A,$ we have $order(i) < order(j)$. A network containing directed cycle cannot be topologically ordered.

Conversely, a directed acyclic graph can be topologically ordered.

Input: Graph $G(N, A)$
Output: Topological ordering $order$ of N
procedure $\text{TopologicalOrdering}(G)$
  $inDegree(i) \leftarrow 0, \forall i \in N$
  $order(i) \leftarrow \texttt{NA}, \forall i \in N$
  $count \leftarrow 1$
  for $(i, j) \leftarrow A$ **do**
    $inDegree(j) \leftarrow inDegree(j) + 1$
  end for
  $Q \leftarrow \{n \in N : inDegree(n) = 0\}$
  while $Q \neq \phi$ **do**
    Remove "next" node i from Q
    $order(j) \leftarrow count$
    $count = count + 1$
    for $j \in FS(i)$ **do**
      $inDegree[j] \leftarrow inDegree[j] - 1$
      if $inDegree[j] == 0$ **then**
        $Q \leftarrow Q \cup \{j\}$
      end if
    end for
  end while
  if $count < |N|$ **then**
    G has cycle(s)
  else
    G is acyclic and return $order$
  end if
  return $order$
end procedure

Connectivity in directed graphs

Definition (Strongly connected graphs). A graph is strongly connected if for every pair of nodes $i, j \in N$, there exists a directed path from i to j. This means that for an arbitrary node $i \in N$, every other node in G is reachable from i and every other node can reach i along a directed path.

We can determine the strong connectivity using two applications of the search algorithm–forward and backward search algorithms.

Thank you!