Graphs

December 1, 2024

Task 1: Time Complexity Analysis

Let $G = (V, E)$ be a directed graph with $n = |V|$ vertices. We aim to find, for each vertex u, the smallest labeled vertex v reachable from u.

Method 2: Transpose Graph and Modified DFS

Adjacency List Representation

- **Time to construct G^T:**
  - For each edge (u, v) in G, add an edge (v, u) in G^T.
  - Time to construct G^T:

$$T_{\text{transpose}} = |E|$$

- **Time for DFS Traversals:**
  - Each vertex is visited exactly once during all DFS traversals.
  - Each edge is traversed at most once.
  - Time for DFS traversals:

$$T_{\text{DFS}} = n + |E|$$

- **Total Time Complexity:**
  - Total time:

$$T(n) = T_{\text{transpose}} + T_{\text{DFS}} = |E| + (n + |E|) = n + 2|E|$$

Conclusion

Method 2 using an adjacency list representation is the most efficient, with a time complexity of $T(n) = n + 2|E|$, which is linear in the size of the graph.

Task 2: Algorithms

Method 2: Transpose Graph and Modified DFS (Adjacency List)

Algorithm 1 FindSmallestReachableVertices_Method2(G)

Input: Graph G with vertices V and edges E
Output: A mapping from each vertex u to the smallest reachable vertex v
$G_T \leftarrow \text{TransposeGraph}(G)$
$\mathit{marked} \leftarrow \emptyset$
$\mathit{result} \leftarrow \{\}$
for each vertex v in V in increasing order **do**
    if $v \notin \mathit{marked}$ **then**
        $\mathit{visited} \leftarrow \emptyset$
        $\mathit{component} \leftarrow []$
        DFS_Method2(G_T, v, $\mathit{visited}$, $\mathit{component}$, marked)
        for each u in $\mathit{component}$ **do**
            $\mathit{result}[u] \leftarrow v$
        end for
        $\mathit{marked} \leftarrow \mathit{marked} \cup \mathit{visited}$
    end if
end for
return result

Algorithm 2 DFS_Method2(G_T, u, $\mathit{visited}$, $\mathit{component}$, marked)

if $u \in \mathit{marked}$ **then**
    return
end if
$\mathit{visited}$.add(u)
$\mathit{component}$.append(u)
for each neighbor v of u in G_T **do**
    if $v \notin \mathit{visited}$ **and** $v \notin \mathit{marked}$ **then**
        DFS_Method2(G_T, v, $\mathit{visited}$, $\mathit{component}$, marked)
    end if
end for

Algorithm 3 TransposeGraph(G)

$G_T \leftarrow$ empty adjacency list
for each vertex u in G **do**
  for each neighbor v of u **do**
    $G_T[v]$.append(u)
  end for
end for
return G_T

Task 3: Description of Significant Edge Cases Tested

Implementation for testing these edge cases can be found here.

- **Disconnected Graph:**
  - A graph where some vertices have no edges connecting them to other vertices.
  - Ensures that the algorithm correctly identifies each vertex as its own smallest reachable vertex when there are no connections.
- **Graph with Self-Loops:**
  - Vertices that have edges pointing to themselves.
  - Tests the algorithm's ability to handle cycles of length one without getting stuck in infinite loops.
- **Complete Graph:**
  - Every vertex has an edge to every other vertex.
  - Ensures that the algorithm efficiently handles graphs with the maximum number of edges and correctly identifies the smallest vertex reachable from any vertex.
- **Graph with Cycles:**
  - Graphs that contain cycles of length greater than one.
  - Tests the algorithm's ability to navigate through cycles without redundant processing and correctly determine the smallest reachable vertex.
- **Single Vertex:**
  - The simplest possible graph containing only one vertex and no edges.
  - Ensures that the algorithm can handle minimal input gracefully.