311 Home Work 3

Ben purdy

April 15, 2020

(1) The reverse (or transpose) of a directed graph $G = (V, E)$ is the graph $G^rev = (V, E^rev)$, where $Erev = \{(v, u) \in V \times V : (u, v) \in E\}$. Thus, G^rev is G with all its edges reversed. Describe efficient algorithms for computing G^rev from G, for both the adjacency-list and adjacency-matrix representations of G. Analyze the running times of your algorithms.

for a graph as represented as a linked list of edges per vertex
(i.e., adjacency lists).

Algorithm 1: How to write algorithms

```
TGraph;
for i = 0; i < vertices; i++ do
    for j = 0; j < v.edges; j++ do
        Add reversed edge to TGraph;
    instructions;
```

Proof. Now in this algorithm you touch every vertice in the list giving use O(v) and also over the hold list we find E edges so the total opporations is given as V+E which equals $O(V + E)$

□

Adjacency matrix representation

Algorithm 2: How to write algorithms

```
TGraph;
for i = 0; i < rows i++ do
    for j = 0; j < cols; j++ do
        swap(G[i, j], G[j, i]);
    instructions;
```

Proof. Trivially since you touch every point in a 2d array the time complexity becomes $O(n^2)$ □

(2) Give an algorithm that determines whether or not a given undirected graph contains a cycle. Your algorithm should run in $O(V)$ time, independent of $|E|$

Algorithm 3: DfsCycle

```
while root.edge has edges and cycle != 1 do
    if root.edge != visited then
        mark root.edge as visited;
        int cycle = DfsCycle(root.edge);
    else
        //found a cycle;
        return 1;
    next root edge;
return cycle;
```

Proof. this algorithm runs a DFS. We know a DFS runs in $O(V+E)$time. . Since this is a acyclic graph that means there are $|E| \leq |V|+1$ edges in the graph. This in turn means that the worst case running time is when we don't find a cycle and we finish the algorithm. giving us an upper bound of $O(V+1) = O(V)$

□

(3) A directed graph G is semiconnected if, for any two vertices $u, v \in V$, there is a path from u to v or a path from v to u (or both). Give an algorithm which determines if a given graph is semiconnected.

Solution.

Algorithm 4: semiconnected

a topological sort on G to get the ordering of its vertices;
for *i=0; from i to k* **do**
  if *there is no edge from vi to vi+1* **then**
    return FALSE
return true;

□

(4) Let $G = (V, E)$ be a directed graph where $V = 1, 2, ..., n$ such that n is odd, i.e., $n = 2k + 1$ for some integer $k > 0$. Given a vertex v, let TO_v be the set of all vertices from which there is a path to v. Let $FROM_v$ be the set of all vertices for which there is a path from $v, i.e.,$

T Ov = u—There is a path from u to v, F ROMv = w—There is a path from v to w. A vertex v is called the center vertex of G if all of the following conditions hold:

- $|TO_v| = |FROM_v| = k$, i.e., both TO_v and $FROM_v$ have exactly k vertices.
- $TO_v \cap FROM_v = 0$, i.e, TO_v and $FROM_v$ are disjoint.

Design an algorithm that gets a graph G (with an odd number of vertices) as input and determines if the graph has a center vertex or not. If the graph has a center vertex, then the algorithm must output it. Describe your algorithm and derive the time complexity

Solution.

I DO NOT NOW

□