Homework 05 — Solutions

CS 624, 2024 Spring

1. Problem 22.1-1 (p592).

   Given an adjacency-list representation of a directed graph, how long does it take to compute the out-degree of every vertex? How long does it take to compute the in-degrees?

   Add `indegree` and `outdegree` fields to each vertex, initialized to 0. Loop over every vertex u and then loop over every node v in the adjacency list for u. In each iteration, we increment the outdegree of u and the indegree of v. This takes time $O(|V| + |E|)$.

2. Problem 22.2-1

   Show the d and π values that result from running breadth-first search on the directed graph of Figure 22.2(a), using vertex 3 as the source.

vertex	d	π
1	∞	NIL
2	3	4
3	0	NIL
4	2	5
5	1	3
6	1	3

3. Problem 22.2-2

   Show the d and π values that result from running breadth-first search on the undirected graph of Figure 22.3, using vertex u as the source.

vertex	d	π
r	4	s
s	3	w
t	1	u
u	0	NIL
v	5	r
w	2	t
x	1	u
y	1	u

4. Problem 22.2-3 (p602) — Note, corrected in 3rd printing.

   Show that using a single bit to store each vertex color suffices by arguing that the BFS procedure would produce the same result if **line 18** were removed.

   The algorithm only compares a node's color against white, so once the node is marked gray it will never be revisited. There is no need to mark it black.

5. Problem 22.2-4 (p602)

What is the running time of BFS if we represent its input graph by an adjacency matrix and modify the algorithm to handle this form of input?

Then we have to scan the whole row of the adjacency matrix to find a node's neighbors, so it takes $O(|V| + |V|^2) = O(|V|^2)$ time.

6. Problem 22.2-7 (p602)

There are two types of professional wrestlers: "babyfaces" ("good guys") and "heels" ("bad guys"). Between any pair of professional wrestlers, there may or may not be a rivalry. Suppose we have n professional wrestlers and we have a list of r pairs of wrestlers for which there are rivalries. Give an $O(n + r)$-time algorithm that determines whether it is possible to designate some of the wrestlers as babyfaces and the remainder as heels such that each rivalry is between a babyface and a heel. If it is possible to perform such a designation, your algorithm should produce it.

Build an undirected graph $G = (V, E)$, where V has n elements. Add an edge between each pair of wrestlers with a rivalry. Constructing the graph takes $O(n + r)$ time, assuming reasonable representations.

We need to check whether G is *bipartite*. That is, its vertices can be divided into two sets, A and B, such that every edge connects a vertex in A to a vertex in B. If a connected graph is bipartite, then a BFS from any vertex will assign A and B distance (d) values of *opposite parities*. Since the graph might not be connected, we must iterate the BFS to explore the whole graph.

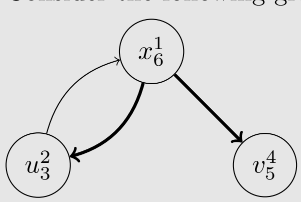
Perform an iterated BFS to explore the whole graph. That is, for each vertex v, if v has not been discovered in an earlier step, run BFS(G, v). This will explore the entire graph, even if there are multiple components. This step takes $O(|V| + |E|) = O(n + r)$ time.

That leads to the candidate assignment: Let the vertices with even d values be the "babyfaces", and let the vertices with odd d values be the "heels". Now we need to traverse the graph once more to check that every edge connects one babyface to one heel. This step takes $O(n + r)$ time.

7. Problem 22.3-8 (p611)

Give a counterexample to the conjecture that if a directed graph G contains a path from u to v and if $u.d < v.d$ in in a depth-first search of G, then v is a descendant of u in the depth-first forest produced.

Consider the following graph. The exploration tree is bolded.



8. Problem 22.3-9 (p612)

Give a counterexample to the conjecture that if a directed graph G contains a path from u to v then any depth-first search must result in $v.d \leq u.f$.

Same example as above.

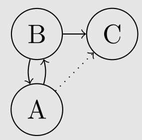
9. Problem 22.5-1 (p620)

How can the number of strongly connected components of a graph change if a new edge is added?

For each possibility, give an example that illustrates it.

It can decrease or stay the same; it cannot increase. (Adding an edge does not invalidate any existing paths.)

Here is an example where adding the dotted edge makes the number of SCCs (2) stay the same:

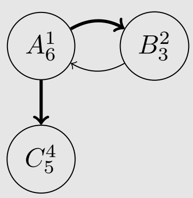


Adding a single edge can reduce the number of SCCs from any number (up to $|V|$) down to 1. For example, conside the linear graph $v_1 \to v_2 \to \cdots \to v_n$. Adding a backwards edge $v_n \to v_k$ reduces the number of SCCs from n to k, for any k between 1 and $n - 1$.

10. Problem 22.5-3 (p620)

Professor Bacon claims that the algorithm for strongly connected components would be simpler if it used the original (instead of the transpose) graph in the second depth-first search and scanned the vertices in order of increasing finishing times. Does this simpler algorithm always produce correct results?

No. Consider the following graph with DFS timestamps:



The first finished node is B, but there is a path in the original graph from B to C, which is not part of the same strongly connected component.