Algorithms Homework 8

Liam Dillingham

November 13, 2018

1 Question 22.2-9

Let $G = (V, E)$ be a connected, undirected graph. Give an $\mathcal{O}(V + E)$-time algorithm to compute a path in G that traverses each edge in E exactly once in each direction. Describe how you can find your way out of a maze if you are given a large supply of pennies.

After trying many examples on paper, I have come up with the idea that the best way to do this is to perform a depth-first search. For some edge $e \in E$ connected to our source node s, we follow a path from s to its neighbor v, each time checking two cases:

- Node v has no unvisited edges
- Node v is the source node s

Once either of these cases is satisfied, we follow our path in reverse direction until we reach our node s again, thus crossing each edge on our path exactly once in each direction. Then, we select another unvisited path and do the same until all edges have been visited once in each direction.

```
// all colors are initially assumed to be white
// Assume that G.Adj has nodes related to themselves
COMPUTE-PATH(G, s)

    // s has unvisited neighbors
    for each vertex v in G.adj[s] where v.color != black
        if v == s
          while !STACK.isEmpty()
              v = STACK.pop()

        // v has no non-visited neighbors
        else if all u in G.adj[v] have u.color == black
          while !STACK.isEmpty()
              v = STACK.pop()

        else
            STACK.push(v)
            COMPUTE-PATH(G, v)
// end
```

2 Question 22.3-7

Rewrite the procedure DFS, using a stack to elimate recursion.

```
DFS(G) // no recursion
    for each vertex u in G.V
        u.color = WHITE
        u.pi = NIL
    time = 0
    STACK is an empty stack
    for each vertex u in G.V
        if u.color == WHITE
            STACK.push(u)
            DFS-VISIT(G, u)

DFS-VISIT(G, u)
    while !STACK.isEmpty()
        v = STACK.pop()
        time = time + 1
        v.d = time
        for each w in G.Adj[v]
            if w.color == WHITE
                w.color = GREY
                w.pi = v
                STACK.push(w)
        time = time + 1
        v.f = time
// end
```

3 Question 22.3-10

Modify the pseudocode for depth-first search so that it prints out every edge in the directed graph G, together with its type. Show what modifications, if any, you need to make if G is undirected.

---

4 Question 22.3-12

Show that we can use a depth-first search of an undirected graph G to identify the connected components of G, and that the depth-first forest contains as many trees as G has connected components. More precisely, show how to modify depth-first search so that it assigns to each vertex v an integer label $v.cc$ between 1 and k, where k isi the number of connected components of G, such that $u.cc = v.cc$ if and only if u and v are in the same connected component.

---

5 Question 24.1-3

Given a weighted, directed graph $G = (V, E)$ with no negative-weight cycles, let m be the maximum over all the vertices $v \in V$ of the minimum number of edges in a shortest path from the source s to v. (Here, the shortest path is by weight, not the number of edges). Suggest a simple change to the Bellman-Ford algorithm that allows it to terminate in $m + 1$ passes, even if m is not known in advance.
