Homework 7

MCS-375

Fall 2011

Exercise 22.2-x1 (1 point)

The textbook's Exercise 22.2-3 (page 602) suggests that the BFS algorithm (page 595) would work just as well without line 18. (If you have an older printing of the book, it erroneously says lines 5 and 14. The third printing corrected this to line 18.) This means there would only be two colors of vertices. When we we coded our Mapper for the Land of Gack, we didn't keep track of vertex colors at all. Translating back into the book's pseudocode, our approach amounted to replacing the test whether $v.color$ is WHITE with a test whether $v.d$ is ∞. Explain why this is equivalent.

Exercise 22.2-x2 (3 points)

A problem dating at least to the 9th century asks how a farmer is to cross a river with his three posessions: a wolf, a goat, and a cabbage. He has available a boat that can carry him along with at most one possession at a time. The goat cannot be together with either the wolf or the cabbage unless the farmer is also present.

We can model this problem using a graph. Each vertex in the graph corresponds to a safe state. For example, we could lable one vertex FWGC to indicate that the farmer, wolf, goat, and cabbage are all on the starting side of the river. Another vertex is labeled WC to indicate that the wolf and cabbage are on the starting side. (Implicitly, the farmer and goat are on the destination side.) There is an edge connecting two vertices if a single river crossing can transform one state into the other. For example, FWGC is adjacent to WC because the farmer can row across with the goat.

Draw the graph. It should have ten labeled vertices and ten edges. You can label vertices using the code illustrated in the previous paragraph. For

the state in which all four participants are on the destination side, you can use the label —.

Explain how the textbook's BFS algorithm could be used to solve this problem. What is the start vertex? What information stored by BFS indicates the number of river crossings that would be needed? How could the specific sequence of river crossings be deduced from the information BFS stores?

Perform the breadth-first search by hand, labeling each vertex with its d value. What is the number of river crossings needed to solve the problem? Alternatively, you can write a program to do this search.

Exercise 22.4-2 (3 points)

This exercise appears on page 614. The phrase "linear time" should be interpreted as $O(V + E)$. The word "simple" can be omitted from "simple paths" without changing the problem.

Exercise 23.2-8 (2 points)

This exercise appears on pages 617-618.

Exercise 24.1-x1 (3 points)

Professor Ross proposes modifying the BELLMAN-FORD algorithm and its RELAX subprocedure as follows:

```
RELAX(u, v, w)
1  if v.d > u.d + w(u, v)
2        v.d = u.d + w(u, v)
3        v.π = u
4        return TRUE
5  return FALSE
```

```
BELLMAN-FORD(G, w, s)
1  INIT-SINGLE-SOURCE(G, s)
2  keepGoing = TRUE
3  while keepGoing
4       keepGoing = FALSE
5       for each edge (u, v) ∈ G.E
6            if RELAX(u, v, w)
7                 keepGoing = TRUE
8  return TRUE
```

1. Professor Ross's version behaves differently from the textbook's on any digraph with a negative-weight cycle reachable from the start vertex. What does each version do for such a digraph?

2. For some digraphs that don't have negative-weight cycles, Professor Ross's version will perform fewer calls to RELAX than the textbook's version. Give an example of such a digraph, together with a designation of the start vertex. Your example should be correct no matter what order either version processes $G.E$ in. Your example should have all vertices reachable from the start vertex.

3. For some digraphs that don't have negative-weight cycles, Professor Ross's version will perform more calls to RELAX than the textbook's version. Give an example of such a digraph, together with a designation of the start vertex. Your example should be correct no matter what order either version processes $G.E$ in. Your example should have all vertices reachable from the start vertex.