Algorithms: Solutions 9

```
                            X
                            X
                            X
                            X
                            X
                            X
                            X
                            X
                            X
                            X
number of                   X
homeworks                   X
                            X
                            X
                            X
             X              X
             X              X
             X  X  X        X
             X  X  X     X  X
             X  X  X     X  X
          X  X  X  X     X  X
          X  X  X  X     X  X
       X  X  X  X  X  X  X  X
       -----------------------
       3  4  5  6  7  8  9  10
              grades
```

Problem 1
Give an example of a directed graph with negative-weight edges for which Dijkstra's algorithm produces incorrect answers.

(A) (B)

If we apply Dijkstra's algorithm to the graph in Figure A, with vertex s as the source, then it processes the vertices in the following order: s, v, w, u. Thus, it constructs the "shortest-paths" tree shown in Figure B, where the path from s to w is *not* shortest.

Problem 2
Suppose that G is a weighted directed acyclic graph, u and v are its vertices, and the graph has at least one path from u to v. Give an algorithm LONGEST-PATH(G, u, v) that finds the *maximum-weight path* from u to v.

We replace all weights in G with their negations, and then apply the algorithm for finding shortest paths in directed acyclic graphs, with u as the source. This algorithm works correctly with negative-weight edges; thus, it finds the shortest path from u to v in the "negated" graph, which is the longest path in the original graph.

Problem 3
Design an algorithm SHORTEST-CYCLE(G, v) that returns the weight of the minimum-weight cycle through v.

We apply Dijkstra's algorithm to find the distances from v to all other vertices, and then use these distances to compute the weight of the shortest cycle. The running time of this procedure is the same as the time of Dijkstra's algorithm.

```
SHORTEST-CYCLE(G, v)
DIJKSTRA(G, v)        ▷ apply Dijkstra's algorithm
min-weight ← ∞        ▷ initialize the shortest-cycle weight
for each u ∈ V[G]
    do for each w ∈ Adj[u]
            do if w = v and min-weight < d[u] + weight(u, w)
                    then min-weight ← d[u] + weight(u, w)
return min-weight
```

Problem 4
Write an algorithm CHECK-LOOP(x) that determines whether a given linked list is "looped." The algorithm's argument x is the first element of the list. If a linked list is looped, then CHECK-LOOP returns TRUE; if the list is not looped, then it returns FALSE.

We may think of a linked list as a one-way road. To check whether it is looped, we simultaneously send two cars, one of which is faster than the other. If the road is *not* looped, the faster car reaches the end in linear time. On the other hand, if the road is looped, the faster car overtakes the other one somewhere on the loop, which also happens in linear time.

To implement this algorithm, we assume that every element y of the linked list has a field $next[y]$, which points to the next element. If y is the last element, then $next[y]$ is NIL.

```
CHECK-LOOP(x)
if x = NIL
    then return FALSE
slow ← x          ▷ position of the slow car
fast ← next[x]    ▷ position of the fast car
while fast ≠ NIL and fast ≠ slow
    do fast ← next[fast]
       if fast ≠ NIL
          then fast ← next[fast]
       slow ← next[slow]
if fast = NIL
    then return FALSE
    else return TRUE
```