CS/ECE 374 FALL 2018
Homework 8 Problem 3

Zhe Zhang (zzhan157@illinois.edu)
Ray Ying (xinruiy2@illinois.edu)
Anqi Yao (anqiyao2@illinois.edu)

Since you are taking an algorithms class you decided to create a fun candy hunting game for Halloween. You set up a maze with one way streets that can be thought of as a directed graph $G = (V, E)$. Each node v in the maze has $w(v)$ amount of candy located at v.

- Each of your friends, starting at a given node s, has to figure out the maximum amount of candy they can collect. Note that candy at node v can be collected only once even if the node v is visited again on the way to some other place.
- Your friends complain that they can collect more candy if they get to choose the starting node. You agree to their their request and ask them to maximize the amount of candy they can collect starting at any node they choose.

Before you ask your friends to solve the game you need to know how to do it yourself! Describe efficient algorithms for both variants. Ideally your algorithm should run in linear time. *Hint:* Consider what happens if G is strongly connected and if it is a DAG.
No proof necessary if you use reductions to standard algorithms via graph transformations and simple steps. Otherwise you need to prove the correctness.

Solution: 1.(1) If G is strongly connected, then the maximum amount of candy they can collect is just the sum of w(v) for each node v because no matter what is the start node, we can go through the entire graph.
Running time: O(m+n)

(2) If G is a DAG.
Let s be the source node and we create a topological ordering of all vertices where for every directed edge (u,v) from vertex u to vertex v, u comes before v in the ordering. The topological ordering could be represented as $v_1, v_2, ..., v_n$. dist[v_i] is used to record the amount of candy collected from s to node v_i.

```
findMaximum:
    dist[] = {$-\infty, -\infty, ...$}
    dist[s] = $w(s)$
    for every node $v_i$ in the topological ordering
        for every node $v_j$ in Adj($v_i$)
            dist[$v_j$] = max{dist[$v_j$], dist[$v_i$]+w($v_j$)}
```

dist[v_n] is the maximum amount of candy they can collect.
The topological sorting takes O(m+n) time. For each vertex, it runs a loop for all the adjacent vertices. The total number of vertices is O(n) and the number of adjacent vertices if O(m). The overall time complexity of this algorithm is O(m+n).

2.(1) If G is strongly connected, then the maximum amount of candy they can collect is just the sum of w(v) for each node v because no matter what is the start node, we can go through the entire graph.

Running time: O(m+n)

(2) If G is a DAG. We could assign weight 0 to each edge of the G and split each vertex v into two vertices v_1 and v_2 with a edge of weight w(v) from v_1 to v_2 where w(v) is the amount of candy located at v. Then we get a new graph G' which is also a DAG. Then the problem of maximizing the amount of candy they can collect becomes finding the longest path in G' with positive length edges. We make a new graph G'' by multiplying -1 to each edge of G' so that $G'' = -G'$. Then the problem becomes finding the shortest path in G" with negative length edges. Thus we could perform the Dijkstra's Algorithm so solve this problem. We could do this in linear time O(m+n).

■