Problem Set 7

> **Due dates:** Electronic submission of the pdf file of this homework is due on **3/24/2023 before 11:59pm** on canvas. The homework must be typeset with LaTeX to receive any credit. All answers must be formulated in your own words.

Watch out for additional material that will appear on Thursday! Deadline is on Friday, as usual.
Name: Huy Quang Lai

Resources. Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford. *Introduction to Algorithms*. The MIT Press.

On my honor, as an Aggie, I have neither given nor received any unauthorized aid on any portion of the academic work included in this assignment. Furthermore, I have disclosed all resources (people, books, web sites, etc.) that have been used to prepare this homework.

Signature: ____________________________

Read the chapters on "Elementary Graph Algorithms" and "Single-Source Shortest Paths" in our textbook before attempting to answer these questions.

Problem 1 (20 points)**.** Give an algorithm that determines whether or not a given undirected graph $G = (V, E)$ contains a cycle. Your algorithm should run in $O(V)$ time, independent of the number $|E|$ of edges.

Solution.

An undirected graph is acyclic if and only if DFS yields no back edges

```
bool DFS(size_t v, size_t parent,
const vector<vector<size_t>>& graph, vector<bool>& visited) {
    visited[v] = true;
    for (size_t u : graph[v]) {
        if (!visited[u]) {
            if (DFS(u, v, graph, visited))
                return true;
        }
        else if (u != parent)
            return true;
    }
    return false;
}

bool has_cycle(const vector<vector<int>>& graph) {
    size_t n = graph.size();
    vector<bool> visited(n, false);
    for (size_t v = 0; v < n; ++v)
        if (!visited[v])
            if (DFS(v, -1, graph, visited))
                return true;
    return false;
}
```

Problem 2 (20 points)**.** Given a weighted, directed graph $G = (V, E)$ with no negative-weight cycles, let m be the maximum over all 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.

Solution.

By the upper bound theory, we know that after m iterations, no d values will ever change. Therefore, under induction, no d values will change in the $(m+1)$-th iteration. However, since the m value is not knowsn in advance, the algorithm cannot iteract exactly m times and then terminate.

Problem 3 (20 points)**.** Suppose that we change line 4 of Dijkstra's algorithm in our textbook to the following.

4 **while** $|Q| > 1$

This change causes the while loop to execute $|V| - 1$ times instead of $|V|$ times. Is this proposed algorithm correct? [Use the version of Dijkstra's algorithm from the textbook]

Solution. This proposed change will not provide a correct output.
Dijkstra's algorithm relies on visiting all vertices in the graph to ensure that the shortest path to each vertex is found. By changing the while loop condition to $|Q| > 1$, the algorithm will terminate early before all vertices have been visited.

Consider a scenario where the graph has n vertices and the shortest path from the source vertex to the destination vertex requires visiting all n vertices. With the modified while loop condition, the algorithm would terminate after $n - 1$ iterations, leaving the shortest path undiscovered.

Therefore, it is essential to have the while loop execute $|V|$ times to ensure that all vertices are visited, and the algorithm finds the correct shortest paths.

Problem 4 (40 points)**.** Help Professor Charlie Eppes find the most likely escape routes of thieves that robbed a bookstore on Texas Avenue in College Station. The map will be published on Thursday evening. In preparation, you might want to implement Dijkstra's single-source shortest path algorithm, so that you can join the manhunt on Thursday evening. [Edge weight 1 means very desirable street, weight 2 means less desirable street]

Solution. Destination Nodes, 6, 8, 9, 15, 16, 22
Since the graph is undirected a connection exists both ways. The format of the graph is `source <->{destination} weight`.

```
1 <->{2} 1, 1 <->{11} 1, 2 <->{3} 1, 2 <->{21} 1, 3 <->{4} 1
3 <->{8} 2, 4 <->{5} 1, 5 <->{6} 2, 5 <->{7} 1, 6 <->{7} 1
7 <->{8} 1, 8 <->{9} 1, 9 <->{10} 1, 9 <->{19} 1
10 <->{11} 1, 10 <->{18} 1, 11 <->{12} 2, 11 <->{17} 1
12 <->{13} 2, 13 <->{14} 2, 13 <->{21} 1, 14 <->{15} 1
14 <->{16} 1, 14 <->{20} 1, 16 <->{17} 1, 17 <->{18} 1
18 <->{19} 2, 20 <->{21} 2, 20 <->{22} 1, 21 <->{22} 2
```

Using Dijkstra's algorithm, results in the following shortest paths to the Destination Nodes.

```
1 --> 2 --> 3 --> 4 --> 5 --> 6 distance: 6
1 --> 2 --> 3 --> 8 distance: 4
1 --> 11 --> 10 --> 9 distance: 3
1 --> 11 --> 17 --> 16 --> 14 --> 15 distance: 5
1 --> 11 --> 17 --> 16 distance: 3
1 --> 2 --> 21 --> 22 distance: 4
```