Due: Thu May 9th at the start of class

Homework #5

CMSC351H - Spring 2019

PRINT Name ____________________________:

- Grades depend on neatness and clarity.
- Write your answers with enough detail about your approach, so that the grader will be able to understand it easily. You should ALWAYS prove the correctness of your algorithms either directly or by referring to a proof in the book.
- Write your answers in the spaces provided. If needed, attach other pages.
- The grades would be out of 16. Four problems would be selected and everyone’s grade would be based only on those problems. You will also get 4 bonus points for trying to solve all problems.

1. [Prob 5.20, pg. 116] Let $x_1, x_2, ..., x_n$ be a set of integers, and let $S = \sum_{i=1}^{n} x_i$. Design an algorithm to partition the set into two subsets of equal sum, or determine that it is impossible to do so. The algorithm should run in time $O(nS)$. 
2. [Prob 7.2, pg. 248] Let $G = (V, E)$ be a connected, undirected graph, and let $T$ be a DFS tree of $G$ rooted at $v$.
   
a. Let $H$ be an arbitrary induced subgraph of $G$. Show that the intersection of $H$ and $T$ is not necessarily a spanning tree of $H$.

   b. Let $R$ be a subtree of $T$, and let $S$ be the subgraph of $G$ induced by the vertices in $R$. Prove that $R$ could be a DFS tree of $S$. 
3. Give an example that the Dijkstra algorithm is not working properly with negative edge weights. The graph should not contain a negative cycle (a negative cycle is a cycle which the sum of the weights of its edges is negative). Briefly explain why Dijkstra does not work in your example.
4. Recall that given a graph $G$, the Bellman-Ford algorithm runs for $n - 1$ iterations, where $n$ is the number of vertices in $G$. Prove that $G$ contains a negative cycle, if and only if at least one edge gets relaxed if we run the algorithm for one extra iteration.

```plaintext
procedure BellmanFord(list vertices, list edges, vertex source)
    // Step 1: initialize graph
    for each vertex v in vertices:
        if v is source then v.distance := 0
        else v.distance := infinity

    // Step 2: relax edges repeatedly
    for i from 1 to size(vertices)-1:
        for each edge uv in edges: // Try relaxing the edge from u to v
            u := uv.source
            v := uv.destination
            if u.distance + uv.weight < v.distance: // uv gets relaxed
                v.distance := u.distance + uv.weight
```
5. Given a string $s$ of length $n$ from a 4-letter alphabet (A,C,G,T), and integer $k$, design an algorithm to determine if there exists a substring of arbitrary length $m$, $k < m \leq n$, that occurs more than once in $s$ (i.e., is there a repeated substring of length greater than $k$). The algorithm should run in $O((n - k)^2)$. HINT: construct a directed graph where each vertex corresponds to those substrings of length $k$ occurring in $s$, with edges $(u,v)$ if the $k$-1 suffix of substring $u$ matches the $k$-1 prefix of $v$. 
6. A 2SAT instance is an instance of Satisfiability in which each clause contains two literals. Design a polynomial algorithm which decides whether a given 2SAT is satisfiable. (Hint: Make a graph based on the input)
7. Consider the weighted graph below which is represented by its adjacency matrix. Run the Dijkstra algorithm starting from vertex 1. Write the vertices in the order which they are marked. Now run the Prim’s algorithm starting from vertex 1. Again write the vertices in the order which they are marked.

\[
\begin{bmatrix}
0 & 5 & 0 & 0 & 8 & 0 & 6 \\
5 & 0 & 0 & 7 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 3 \\
0 & 7 & 0 & 0 & 2 & 0 & 0 \\
8 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 4 \\
6 & 0 & 3 & 0 & 0 & 4 & 0
\end{bmatrix}
\]
8. A Hamiltonian path in graph $G=(V,E)$ is a simple path that includes every vertex in $V$. Design an algorithm that runs in linear time, to determine if a Hamiltonian path exists in a given directed acyclic graph.
9. [Prob. 7.61, pg. 256]. Let $G=(V,E)$ be a connected, undirected graph, and let $T$ be a minimum cost spanning tree of $G$. Suppose the cost of one edge $e$ in $G$ is changed. Discuss the conditions in which $T$ is no longer a MCST of $G$. Design an efficient algorithm to either find a new MCST or to determine that $T$ is still an MCST. ($e$ may or may not belong to $T$).
10. Consider an $N \times N$ (N even) board of alternating black and white squares (like a chess board). Prove that if one removes an arbitrary black square and an arbitrary white square, the rest of the board can be covered by dominoes (of size 2x1). (State the problem as a graph, and think how finding a Hamiltonian paths can be used prove this).
Let $G=(V,E)$ be an undirected graph, such that each vertex is associated with some task. Two vertices are connected if the corresponding tasks cannot be performed at the same time (e.g., they need the same resource). This is the only limit on concurrency. Any set of tasks such that no two of them are connected can be performed in one step. Prove that the following problem is NP-complete: Given a graph $G=(V,E)$, and an integer parameter $k$, determine whether all corresponding tasks can be performed in at most $k$ steps.