Algorithms

Assignment: Graphs

<table>
<thead>
<tr>
<th>Grade</th>
</tr>
</thead>
</table>

Name: ......................................................................................................................................

Id: ......................................................................................................................................

Good Luck!
1. Let $G$ be a simple undirected graph with $n$ vertices and $m$ edges. Apply the following process on $G$:

- As long as $G$ contains at least one vertex with degree 1 (exactly one neighbor), remove one of these vertices and the edge that contains it from $G$.

For which graphs does this process terminate with a graph that has exactly one vertex? Explain.

2. The following two algorithms are applied to a simple undirected graph with $n$ vertices in the set $V$.

(a) What could you say about set $S$ when the following algorithm terminates? Justify your answer.

```
let $S$ be a set containing an arbitrary vertex $v \in V$: $S = \{v\}$
let $R$ be the set off all the neighbors of $v$: $R = \{\text{neighbors}(v)\}$
while $R$ is not empty do
    let $v \in R$ be an arbitrary vertex from $R$
    add $v$ to $S$: $S = S \cup \{v\}$
    omit $v$ from $R$: $R = R \setminus \{v\}$
    omit from $R$ all vertices that are not neighbors of $v$: $R = R \setminus \{\text{nonneighbors}(v)\}$
```

(b) What could you say about set $S$ when the following algorithm terminates? Justify your answer.

```
let $S$ be the empty set.
let $R = V$ be the set off all vertices
while $R$ is not empty do
    let $v \in R$ be an arbitrary vertex from $R$
    add $v$ to $S$: $S = S \cup \{v\}$
    omit $v$ from $R$: $R = R \setminus \{v\}$
    omit from $R$ all vertices that are neighbors of $v$: $R = R \setminus \{\text{neighbors}(v)\}$
```

3. Let $G$ be a simple undirected graph with the $n$ vertices $\{1,2,\ldots,n\}$ and $m$ edges. Define $H$ as follows. The vertices of $H$ are the vertices of $G$. The edge $(u,v)$ is in $H$ if either $(u,v)$ is an edge in $G$ or there exists another vertex $w$ such that both $(u,w)$ and $(w,v)$ are edges in $G$.

In other words, there exists an edge $(u,v)$ in $H$ if and only if there exists a path of length at most 2 from $u$ to $v$ in $G$.

(a) Which is $H$ if $G$ is the complete graph? Explain.
(b) Which is $H$ if $G$ is the null graph? Explain.
(c) Which is $H$ if $G$ is the complete bipartite graph? Explain.
(d) Is $H$ a tree if $G$ is a tree? Explain.

(e) Assume $G$ is represented by an adjacency matrix. Describe an algorithm to generate the adjacency matrix of $H$. Justify the correctness of your algorithm. What is the complexity of your algorithm? Explain.

(f) Assume $G$ is represented by sorted adjacency lists. Describe an algorithm to generate the sorted adjacency lists of $H$. Justify the correctness of your algorithm. What is the complexity of your algorithm? Explain.
4. A legal coloring of a simple and undirected graph is an assignment of colors (positive integers) to the vertices in a way that any two adjacent vertices (neighbors) must be colored with different colors. The following is a greedy algorithm that colors the vertices of an input graph with the colors: 1, 2, 3, \ldots. Assume that the graph has \( n \) vertices denoted by: \( v_1, v_2, \ldots, v_n \).

\[
\text{for } i = 1 \text{ to } n \text{ do } color(v_i) = 0
\]
\[
\text{for } i = 1 \text{ to } n \text{ do }
\]
\[
c = 1
\]
\[
\text{while } (v_i \text{ has a neighbor } v_j \text{ such that } color(v_j) = c) \text{ do } c = c + 1
\]
\[
\text{end-for}
\]

Note that the algorithm does not specify how to implement the condition of the \textbf{while} loop. An implementation may include additional data structures beyond the regular graph representation (adjacency lists or an adjacency matrix).

(a) Find an assignment of the letters \( A, B, \ldots, J \) to the vertices of the Petersen graph for which the greedy algorithm that orders the vertices in alphabetical order would use 4 colors. Near each vertex, write its letter and color.

(b) Let \( G \) be a \( \Delta \)-regular graph. That is, any vertex has exactly \( \Delta \) neighbors. Prove, that the greedy algorithm needs only the colors 1, \ldots, \( \Delta + 1 \). Prove this for any value of \( n \) and \( 0 \leq \Delta < n \).

(c) Let \( L(G) \) be the line graph of \( G \). The vertices of \( L(G) \) represent the edges of \( G \). Two vertices in \( L(G) \) are connected by an edge if the corresponding edges in \( G \) share a vertex. Assume that \( G \) is a \( \Delta \)-regular graph. Prove, that the greedy algorithm, applied on the graph \( L(G) \), needs only the colors 1, \ldots, \( 2\Delta - 1 \). Prove this for any value of \( n \) and \( 0 \leq \Delta < n \).

(d) A bipartite graph can be colored with 2 colors. Construct a \textbf{small} bipartite graph and order its vertices such that the greedy algorithm that follows this order would use more than 2 colors.

(e) In the previous part, your example should be a graph with a small number of vertices. In this part, you are asked to scale your solution for infinite values of \( n \). Construct a \textbf{family} of bipartite graphs and order the vertices of each graph in the family such that the greedy algorithm that follows this order would use \( \Omega(n) \) colors. If possible illustrate these graphs and show the order among the vertices. Use the picture to prove the \( \Omega(n) \) claim.

5. Let \( G \) be a simple undirected graph with \( n \) vertices and \( m \) edges. A triangle in a graph is a set of three vertices \( u, v, w \) such that the three edges \((u, v), (v, w), (w, u)\) are in the graph.

(a) Describe a \textbf{correct} algorithm that decides whether or not a graph has a triangle. What is the complexity of your algorithm? Explain your answer.

(b) Describe a \textbf{correct and efficient} algorithm that decides whether or not a graph has a triangle. What is the complexity of your algorithm? Explain your answer.

6. An \textit{independent set} in a graph is a set of vertices such that there is no edge between any pair of vertices in the set.

A \textit{maximum independent set} in a graph is an independent set that has the largest number of vertices among all possible independent sets.
A maximal independent set in a graph is an independent set that is not contained in another independent set. That is, any new vertex that would be added to the set would create a non-independent set.

(a) Find the maximum independent set in a complete bipartite graph that has \( x \) vertices in one side and \( y \) vertices in the other side for \( x + y = n \). What is the size of this set?

(b) Find the smallest possible maximal independent set in a complete bipartite graph that has \( x \) vertices in one side and \( y \) vertices in the other side for \( x + y = n \). What is the size of this set?

(c) Finding a maximum independent set is an NP-Complete problem whereas finding a maximal independent set is not. Describe an efficient (greedy) algorithm that finds one of the maximal independent sets in a graph. What is the worst case running time of your algorithm if the graph is maintained with adjacency lists? What is the worst case running time of your algorithm if the graph is maintained with adjacency matrix? Explain your answers.

7. Let \( G \) be a simple undirected graph with \( n \) vertices and \( m \) edges. A simple path in \( G \) is a path in which all the vertices are different. The length of a simple path is the number of vertices in the path.

A simple path is maximal if it cannot be extended by another vertex. That is, the path \( v_1 - v_2 - \cdots - v_k \) is maximal if both \( v_1 \) and \( v_k \) have no neighbors outside the path.

The following algorithm generates a maximal path:

\[
\text{Algorithm Maximal-Path}
\]

Start with an arbitrary vertex \( v \)
In a “DFS manner” extend \( v \) to one side until it is impossible.
In a “DFS manner” extend \( v \) to the other side until it is impossible.

(a) What is the complexity of Algorithm Maximal-Path when the graph is maintained by adjacency lists? Explain.
What is the complexity of Algorithm Maximal-Path when the graph is maintained by an adjacency matrix? Explain.

(b) The guaranteed performance of Algorithm Maximal-Path is the guaranteed length of the output path no matter which was the starting vertex and how the DFS search was implemented.
For example, the guaranteed performance of Algorithm Maximal-Path on the the Null Graphs is 1 since there are no edges and the guaranteed performance of Algorithm Maximal-Path on the Complete Graphs is \( n \) since all the edges exist.

(c) What is the guaranteed performance of Algorithm Maximal-Path on a Complete Bipartite Graphs in which one size has \( x \) vertices, the other side has \( y \) vertices, \( x + y = n \), and \( x < y \)? Explain.

(d) What is the guaranteed performance of Algorithm Maximal-Path on a Cycle Graph with \( n \) vertices? Explain.

(e) What is the guaranteed performance of Algorithm Maximal-Path on a Star Graph with \( n \) vertices? Explain.

(f) What is the guaranteed performance of Algorithm Maximal-Path on a graphs with \( n \) vertices in which the degree of every vertex is at least \( \delta \) for \( 0 \leq \delta \leq n - 1 \)? Explain.