Analysis of Algorithms: Solutions 6

\[
\begin{array}{cccccccccc}
& x & x & x & x & x & x & x & x & x \\
\text{number of homworks} & x & x & x & x & x & x & x & x & x \\
\end{array}
\]

The histogram shows the distribution of grades for the homeworks submitted on time.

**Problem 1**
Write algorithms for converting (a) an adjacency-list representation of a graph into an adjacency matrix and (b) an adjacency matrix into adjacency lists. Give the time complexity of your algorithms.

We denote an adjacency list of a vertex \( u \) by \( \text{Adj-List}[u] \), and an adjacency-matrix element for vertices \( u \) and \( v \) by \( \text{Adj-Matrix}[u,v] \). The time complexity of both algorithms is \( \Theta(V^2) \).

(a) Converting adjacency lists into a matrix.

\[\text{LISTS-TO-MATRIX}(G) \rightarrow G \text{ is represented by adjacency lists.} \]
\[
\text{for each } u \in V[G] \\
\quad \text{do for each } v \in V[G] \\
\quad \quad \text{do } \text{Adj-Matrix}[u,v] \leftarrow 0 \\
\text{for each } u \in V[G] \\
\quad \text{do for each } v \in \text{Adj-List}[u] \\
\quad \quad \text{do } \text{Adj-Matrix}[u,v] \leftarrow 1
\]
(b) Converting an adjacency matrix into lists.

\textsc{Matrix-to-Lists}(G) \xrightarrow{} G is represented by an adjacency matrix.

\textbf{Problem 2} (3 points)
Using Figure 23.3 in the textbook as a model, illustrate the steps of breadth-first search on the directed graph of Figure 23.2(a), with vertex 3 as the source.

The order of painting the vertices is as follows:

\begin{align*}
\text{gray 3} & \quad \text{black 5} \\
\text{gray 5} & \quad \text{black 6} \\
\text{gray 6} & \quad \text{gray 2} \\
\text{black 3} & \quad \text{black 4} \\
\text{gray 4} & \quad \text{black 2}
\end{align*}

\textbf{Problem 3}
The depth-first search algorithm may be used to identify the connected components of an \textit{undirected} graph. Write a modified version of DFS for performing this task.

We use the \textit{component} field of a vertex instead of the color. Initially, this field is set to 0. When the DFS algorithm discovers the vertex, it replaces 0 with the component number.

\textsc{DFS-Components}(G)
\begin{align*}
k & \leftarrow 0 \quad \triangleright \text{Component counter.} \\
\text{for each } & u \in V[G] \\
& \quad \text{do component}[u] \leftarrow 0 \\
\text{for each } & u \in V[G] \\
& \quad \text{do if component}[u] \neq 0 \\
& \quad \quad \text{then } k \leftarrow k + 1 \\
& \quad \quad \text{DFS-Visit}(u, k)
\end{align*}

return $k$

\textsc{DFS-Visit}(u, k)
\begin{align*}
\text{component}[u] & \leftarrow k \quad \triangleright u \text{ has just been discovered.} \\
\text{for each } & v \in Adj[u] \\
& \quad \text{do if component}[v] = 0 \\
& \quad \quad \text{then DFS-Visit}(v, k)
\end{align*}