Due 3/7/05

1. In this problem you will modify a basic depth-first search (DFS) algorithm to find connected components of an undirected graph. Assume we are given the adjacency list representation of an undirected graph. There is an array $M[1..n]$, with entries initially 0, that is used to indicate if a vertex has been visited. The basic recursive DFS algorithm is

\[
\text{DFS}(i: \text{vertex})
\]
\[
M[i] := 1;
\]
\[
\text{for each vertex } j \text{ adjacent to } i \text{ do}
\]
\[
\quad \text{if } M[j] = 0 \text{ then DFS}(j)
\]
\[
\text{end DFS}
\]

This DFS algorithm will only search the vertices that are reachable from the vertex where the algorithm is first called. Thus, we need to apply it to all the vertices.

\[
\text{Main}
\]
\[
\quad \text{for each vertex } i \text{ do}
\]
\[
\quad \quad \text{if } M[i] = 0 \text{ then DFS}(i)
\]
\[
\text{end Main}
\]

Modify these algorithms so that $i$ and $j$ are in the same connected component if and only if $M[i] = M[j]$.

2. Problem 23-4 on page 577 of CLRS.

3. One of the most famous algorithms in computer science is Dijkstra’s algorithm which finds the shortest path from a single source in a weighted directed graph. This algorithms is used to find "best" routes in the Internet. Let $G = (V, E)$ be a directed graph with weight $w(i, j) > 0$ for each $(i, j) \in E$. Let $s \in V$ be the source vertex. We will compute $d(i)$ and $p(i)$ for each vertex $i$ where $d(i)$ is the length of the shortest past from $s$ to $i$ and $p(i)$ is the predecessor of $i$ on a shortest path from $s$ to $i$. Initially, $d(s) = 0$ and $d(i) = \infty$ for all other $i$. Initially $p(i) = 0$ for all $i$. Initially, let $Q = V$

\[
\text{Dijkstra}
\]
\[
\quad \text{while } Q \text{ is not empty do}
\]
\[
\quad \quad \text{choose } i \text{ from } Q \text{ with minimal } d(i);
\]
\[
\quad \quad \text{remove } i \text{ from } Q;
\]
\[
\quad \quad \text{for each } j \text{ adjacent to } i
\]
\[
\end Dijkstra
\]
if \( d(j) > d(i) + w(i,j) \) then
\[
\begin{align*}
    d(j) &:= d(i) + w(i,j); \\
p(j) &:= i
\end{align*}
\]
end{Dijkstra}

It can be shown as an invariant that if \( i \) is not in \( Q \) then the current value of \( d(i) \) is the length of the shortest path from \( s \) to \( i \) and \( p(i) \) is the predecessor on such a path.

In this problem you will show how Dijkstra’s algorithm can be adapted to solve the problem of **maximally reliable path**. In this problem we are given a weighted directed graph where the weight of the edge \((i, j)\) represents the probability that the edge \((i, j)\) will be available for any path. This probability is just a real number \( r(i, j) \) where \( 0 \leq r(i, j) \leq 1 \). The value \( 1 - r(i, j) \) is the probability that edge \((i, j)\) will fail. We assume that edges fail independently. Modify Dijkstra’s algorithm to solve the problem, given \( s \) and \( t \) determine the most reliable path from \( s \) to \( t \). The reliability of a path is the product of availability probabilities of the edges on the path.