Programming Assignment 7

Looking for Cycles in a Graph

Submission Deadline: 11:55PM, Sunday Nov. 16, 2014

Submissions only via Moodle

An alternative way to represent a graph is as an *adjacency list*. Assume the nodes of the graph are numbered from 1 to N. An adjacency list representation of a graph lists for each node i, the set of immediate neighbouring nodes of node i, i.e., all those nodes j such that there is an edge between (i, j) in the graph.

A possible way of representing an adjacency list is

```
type adj_list = (int * (int list)) list;;
```

This is a list of elements, each of the form $(i, [j_1; \ldots; j_k])$ which means that there are edges from i to each of j_1 ... j_k.

Depending on the operations, if one wishes to search for a node's adjacency list quickly, one may use a structure such as a binary search tree, where finding a node's information is quicker.

In this assignment, you will be given as input a directed graph in adjacency list form, with nodes labeled from 1 to N for some N. The problem is to list all possible cycles that appear in the graph. In addition, you should find the longest cycle. Recall that a cycle is a sequence of nodes $u_0, u_1, \ldots u_k$ such that all the edges (u_i, u_{i+1}) exist for $0 \leq i < k$ and also the edge (u_k, u_0) exists.

The input will be provided as an `(int * int list) list`. You will need to write a function `validGraph : (int * int list) list -> bool` that verifies that the given input is a graph. Note that the node ids in the list `l` could be any integers and need not be in the range 1 . . .`Length l`.

You should use your Stack package (possibly modified). You must document your programs adequately, and test your program. You should also analyse the time and space required.