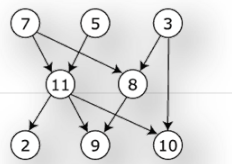
Prolog and Haskell

This assignment will be weighted at 50% of other assignments due to its brevity.



Haskell Graphs

Overview

A simple way to represent a directed graph is via a list of edges. An edge can be thought of as a pair of nodes where the first element is the “source” node and the second element is the “to” node. For simplicity, we will assume that nodes are represented as integers. For example, in Haskell, we could define the following types:

```
type Node = Integer
type Edge = (Integer, Integer)
type Graph = [Edge]
type Path = [Node]
```

The code below shows how to define two graphs labeled g and h using these type definitions.

```
g :: Graph
g = [(1,2), (1,3), (2,3), (2,4), (3,4)]
h :: Graph
h = [(1,2), (1,3), (2,1), (3,2), (4,4)]
```

Problems

You must write Haskell Graph processing functions. Each function must include a signature.

1. `nodes` . This function accepts a graph and returns a list of nodes given in ascending order. For example
   - nodes g => [1,2,3,4]

2. `neighbors` . This function accepts a node N and a graph G. The function returns a list of all nodes that are connected to N in G where N is the source. For example
   - neighbors 2 g => [3,4]
   - neighbors 4 g => []
   - neighbors 4 h => [4]

3. `detach` . This function accepts a node N and a graph G and returns the graph that results from removing N from G. For example
   - detach 3 g => [(1,2), (2,4)]

4. `paths` . This function accepts two nodes, N1 and N2 and a graph G. This function returns a list of all possible cycle-free paths starting at N1 and terminating at N2 in G. For example
   - paths 2 2 g => [[2]]
   - paths 3 2 g => []
   - paths 1 4 g => [[1,2,3,4], [1,2,4], [1,3,4]]

Prolog Graphs

Overview

Assume that the predicate `edge(a,b,10)` means that there is a directed edge between nodes 'a' and 'b' of a graph and that the edge has a cost of 10 units. Assume that the edge cost will always be a positive integer value. Write the following functors.

Problems

1. `path(FROM, TO, COST)` . This is satisfiable if the graph contains at least one path that starts at FROM and terminates at TO and has a cost of COST.

2. `path(FROM, TO, COST, P)` . This functor is satisfiable if the graph contains a path P that starts at FROM and terminates at TO and has a cost of COST. If there are multiple such paths, the functor should retrieve them all.

Prolog Lists

Overview

Just as in Scheme and Haskell, list processing is a basic feature of Prolog. Write the following Prolog functors.

Problems

1. `grows(L)` . This predicate is satisfied only when L is a list of numbers in ascending order and contains no duplicates. For example,
    a. grows([1,3,4]) succeeds
    b. grows([3,3,4]) fails
    c. grows([1,2,3,1]) fails

2. `sumrunner(L,S)` . This predicate is satisfiable when L is a list of integers and S is a list of the prefix sums. For example
    a. sumrunner([3,12,1,-4,9], [3,15,16,12,21]) succeeds
    b. sumrunner([-3,-4,-4], R) succeeds when R = [-3, -7, -11]

Requirements

Submit your code in GitLab under a folder named "hw6". Within this folder you will have two files: one named *graph.hs* (containing the haskell code) and one named *logic.pl* (containing all Prolog code).