Algorithms and Data Structures
Jacobs University Bremen
Dr. Florian Rabe

Homework 9
given: 2017-04-06

Homework 9

You have to submit your solutions as announced in the lecture.
Unless mentioned otherwise, all problems are due 2017-04-20, 8:00.
There will be no deadline extensions unless mentioned otherwise in the lecture.

Problem 9.1 *Graphs*

Points: 10

Implement a type *Graph* for the following special case of graphs:
A graph consists of

- the number $numNodes : int$ of nodes (assuming $N = \mathbb{Z}_{numNodes} = \{0, \ldots, numNodes - 1\}$)

supports the side-effect free operations

- $getEdge(i : int, j : int) : Option[int]$
- $incoming(i : int) : List[int \times int]$
- $outgoing(i : int) : List[int \times int]$

and is mutable via operations

- $addNode() : unit$
- $addEdge(from : int, to : int, weight : int) : unit$

All operations should be efficient—so use the double adjacency list data structure.
Write a large graph on paper (include the graph in your submission) and write a program that builds it.

Problem 9.2 *BFS*

Points: 6

Implement a function for BFS in a graph, i.e.,

precondition: $0 \leq start < G.numNodes$
postcondition: $BFS(G, start)$ is list of nodes reachable in G from $start$ in BFS order
fun $BFS(G : Graph,\ start : int) : List[int]\ =$
$\quad\ldots$

Test your program on the example graph from above.

Problem 9.3

Points: 10

Implement Kruskal's algorithm.
You do not have to implement $isSetOfTrees$ with optimal efficiency—any implementation is fine.
Test your program on the example graph from above.