Universität Innsbruck - Institut für Informatik
Prof. Clemens Ballarin, Robert Binna, Friedrich Neurauter, Francois Scharffe and Sarah Winkler

1 July 2008

Proseminar Algorithmen und Datenstrukturen

Exercise Sheet 14

Exercise 1 (Depth-first Search)

Given a directed graph $G = (V, E)$ and some node $v \in V$, write a C program that numbers all nodes in the graph according to the order in which they are traversed during depth-first search; the number associated with v should be 1. Make sure that all nodes are numbered, and that all numbers are different! Use adjacency matrices to represent graphs!

See listing *dfs_number.c*

Exercise 2 (Depth-first Search)

Given a connected, undirected graph $G = (V, E)$, write a C program that computes a spanning tree of G using depth-first search. Use adjacency matrices to represent graphs!

See listing *dfs_spantree.c*

Exercise 3 (Connected Components)

Given an undirected graph, provide pseudo-code for the computation of its connected components.

Exercise 4 (Connected Components)

Given some node v of an undirected graph, provide pseudo-code for the computation of the number of edges in the connected component v belongs to.

Algorithm 1 Connected components of an undirected graph

Input: G: undirected graph

```
begin
component_number := 1
while v is unmarked in G do
  call DFS(v, component_number) /*DFS marking nodes with their comp-nr */
  component_number := component_number + 1
end
```

Algorithm 2 $DFS\_CountEdges$, number of edges in connected component of a node

Input: G: undirected graph, v: some node of G
Output: twice the number of egdes of the component v belongs to

```
begin
mark(v)
for each w adjacent to v do
  edgecount := edgecount + 1
  if w is unmarked then
    call DFS_CountEdges(G, w)
end
```