12. Problem Set on

Algorithms, Data Structures and Data Abstraction WS 18/19

Wolfgang Mulzer

Due on 25. January 2019, 10:00, in the respective TA's mailbox

Problem 1 Tries *10 Points*

(a) Draw an uncompressed and a compressed trie for the following set of strings: {ALGORITHMUS, TRIE, BAUM, TORUS, BAHN, TORPEDO}.

(b) Describe an algorithm that outputs all the strings stored in a given uncompressed trie and that visits each edge of the trie at most twice.

Problem 2 Implementation of Tries *10 Points*

Describe briefly, how a concrete implementation of the operations `put`(s, v), `get`(s), `remove`(s), and `succ`(s) in an uncompressed and a compressed trie could be implemented. Here, s is a nonempty string, and v is a value from a set V. What are the running times of your implementations?

Problem 3 Depth-First Search *10 Points*

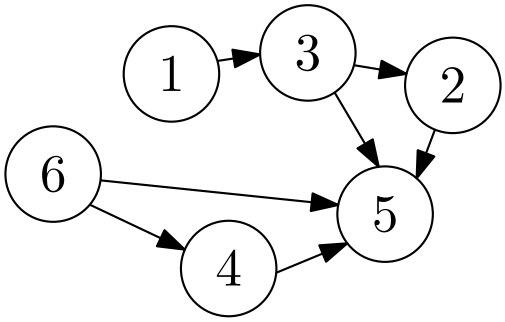
(a) Use depth-first search in order to find a walk in a connected, undirected, simple graph that visits each edge exactly once in each direction.

(b) Let $G = (V, E)$ be a directed acyclic graph (DAG). A *topological ordering* of G is an ordering of the vertices of G such that no edge in G goes from a later to an earlier vertex in the ordering. Consider the following algorithm.

```
dfs(v, topoOrder)
    v.found ← true
    for e in v.outgoingEdges() do
        w ← e.opposite(v)
        if ¬ w.found then
            dfs(w, topoOrder)
    topoOrder.push(v)

topoSort(G)
    topoOrder ← new Stack
    for v in G.vertices() do
        v.found ← false
    for v in G.vertices() do
        if ¬ v.found do
            dfs(v, topoOrder)
```

Execute the algorithm on the following graph. Assume that the vertices in `vertices` and in `outgoingEdges` are ordered according to their numbers. Show the individual steps. How can the topological ordering be obtained in the end?



(c) (*voluntary*, 5 additional points) Prove that the algorithm in (b) is correct.
Hint: Argue by contradiction. Suppose there is an edge e that connects a later vertex to an earlier vertex in the ordering obtained by `topoSort`. Think about what happens when `dfs` encounters the edge e. Note that it makes a difference whether the DFS for the end point of e is still active.

Problem 4 Collections in Java *voluntary, 10 additional points*

(a) In class, you have seen various abstract data types that store sets of elements or of pairs of elements. Name four such ADTs, and give two substantially different implementations for each one.

(b) Read about the *Java Collections Framework*. How is it structured? Where and how can you recognize the ADTs from (a)? How does the framework implement these ADTs?

Remark: If you prefer, and if your TA does not object, you can solve (b) also for the run-time libraries of another programing language, e.g., the STL of C++, the .Net framework, or the Python standard library.