Marburg
University

Department of Computer Science
Prof. Dr. Sebastian Wild

Date: 2025-12-12
Version: 2025-12-12 11:40

Sheet 9 for
Effiziente Algorithmen (Winter 2025/26)

Hand In: *Until 2025-12-19 18:00, on ILIAS.*

Problem 1

30 points

Consider the $(7, 4)$ Hamming code from the lecture.

a) Given the message `0101`, determine the parity bits and the block to be transmitted.

b) Is `1111111` a valid block, i.e., have (detectable) errors occurred?

Problem 2

10 + 30 points

We can obtain a simpler implementation of depth-first search via a recursive method such as the following Java implementation:

```
public class DepthFirstSearch {
    Graph G;  boolean[] visited;

    public DepthFirstSearch(Graph G) {
        this.G = G; visited = new boolean[G.n];
    }

    public void dfs(int s) {
        visited[s] = true;
        for (int v : G.adj[s])
            if (!visited[v]) dfs(v);
    }
}
```

a) Which problem does this implementation run into even on moderately large graphs?

b) A straight-forward solution for the problem from a) is to avoid the recursion. We can do so without the iterator-based approach from the lecture, e.g., using the following, much simpler Java-Code:

```java
public void dfsIterativeSimple(int s) {
    Stack<Integer> todo = new LinkedStack<>();
    todo.push(s);
    while (!todo.empty()) {
        int v = todo.pop();
        visited[v] = true;
        for (int w : G.adj[v])
            if (!visited[w]) todo.push(w);
    }
}
```

Analyze the running time and space usage of `dfsIterativeSimple` and compare the result with the DFS from class.

Problem 3

30 points

Given a digraph G such that there is a path from a vertex u to a vertex v, prove or disprove the following: In every depth-first search, v becomes *active* before u becomes *done*.

Problem 4

20 + 50 points

A *superstar* is a person who is known by every other person, but who knows no other person.

a) If you model a set of n people and the relation "a knows b" as a digraph, i.e. "a knows b" $\iff$ $(a, b) \in E$, how can you characterize a superstar?

b) Let the digraph from part a) be given as an $n \times n$ adjacency matrix A. Design an algorithm that, given A, decides in $o(n^2)$ time whether the population modeled by A contains a superstar.

Justify the correctness of your algorithm and prove a runtime bound that serves the purpose of the exercise.