Lehrstuhl für Informatik I
– Algorithmen und Komplexität –
Universität Würzburg

Würzburg, October 26, 2022

Prof. Dr. Alexander Wolff
Johannes Zink
Oksana Firman
Marie Diana Sieper

Exercise Sheet #2

Advanced Algorithms (WS 2022/23)

Exercise 1 – Hamiltonian path

A *Hamiltonian path* is a path in a graph that visits each vertex exactly once.

a) Let G be a non-weighted, undirected graph. Describe an algorithm for deciding whether G contains a Hamiltonian path. Use dynamic programming.

   What are the running time and space consumption of your algorithm?

   5 Points

b) How can we alter the algorithm such that it actually outputs the Hamiltonian path? **2 Points**

c) Now let G be an undirected graph that has, for each edge $e \in E(G)$, an edge weight $w(e) \in \mathbb{R}$. Show how to find a shortest Hamiltonian path, i.e., a Hamiltonian path P of smallest total weight $W = \sum_{e \in P} w(e)$. **3 Points**

Exercise 2 – Edge-branching INDEPENDENT SET

In the second lecture we talked about a branching algorithm for MAXIMUM INDEPENDENT SET. The algorithm was based on the following properties:

(Vertex 1) If a vertex is in the independent set, then its neighbours aren't in the independent set.

(Vertex 2) If a vertex is not in the independent set, then in a maximum independent set at least one of its neighbours is in the independent set.

Branching algorithms are often based on such observations about the properties of feasible and/or optimal solutions. We will now design an algorithm based on a different property of independent sets:

(Edge) Consider an edge (v, w). An independent set does not contain both v and w.

a) Design a simple branching algorithm for MAXIMUM INDEPENDENT SET using a single *branching rule* based on the Edge property. Additionally, you can use a *reduction rule* based on the fact that isolated vertices belong to every maximum independent set. **5 Points**

b) Show that the algorithm you designed in (a) runs faster than $\mathcal{O}^*(2^n)$, where n is the number of vertices of the given graph. **3 Points**

c) Show that your running time analysis is tight by constructing a suitable family of worst-case instances. **2 Points**

Please hand in your solutions on Wuecampus until the beginning of the next lecture, that is 14:15 on Wednesday, November 2.