University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
P. Bamberger
P. Schneider

Algorithms and Datastructures
Summer Term 2020
Exercise Sheet 8

Due: Wednesday, 8th of July, 4 pm.

Exercise 1: Machine Scheduling - Part 1 *(12 Points)*

Assume that n jobs $0 \ldots, n-1$ should be processed by a machine. Some jobs may depend on others. Job j depends on job i means that job i has to be processed before job j.
The task is to design an algorithm that takes as input the number n of jobs as well as the information about the job's dependencies. The output should be a valid schedule, i.e., a list of the jobs in which job i appears before job j if job j depends on job i.
The algorithm should also recognize if there is no valid schedule and give an according output in this case.
The dependencies between the jobs are given as a directed graph $G = (V, E)$ with nodes $V = \{0, \ldots, n-1\}$ and an edge $(i, j) \in E$ if job j depends on job i.

(a) Implement an algorithm with the described functionality, whose runtime is at most linear in the size of G. Assume that the dependency graph is given as an adjacency list. You can use the template `Scheduling.py`. A class for representing graphs as adjacency lists together with some functions is defined in `AdjacencyList.py`. *(8 Points)*

(b) Convert the dependency graph from `dag.txt` into an adjacency list (you may use the function `read_graph_from_file` from `Scheduling.py`). Execute your algorithm on this input and copy your result to `erfahrungen.txt`. *(4 Points)*

Exercise 2: Machine Scheduling - Part 2 *(8 Points)*

We consider the same problem as in exercise 1 with the difference that now each job depends on at most one other job (i.e., each job has in-degree ≤ 1 in the dependency graph). We also assume that there is a valid schedule, i.e., the dependency graph has no cycles.
Assume we have an arbitrary number of machines to process the jobs in parallel. The task is to determine for each job the earliest possible time it can be processed, assuming that we start at time 0 and a machine takes one time step to process one job. The algorithm should return an array A of length n such that $A[i]$ equals the earliest possible time job i can be processed.

(a) Implement an algorithm with the described functionality with runtime at most linear in the size of the depency graph. Assume that the dependency graph is given as an adjacency list. *(6 Points)*

(b) Convert the dependency graph from `forest.txt` into an adjacency list (this graph has the property that each node has in-degree ≤ 1). Use the algorithm from (a) to determine the minimum time needed to finish all jobs. Write your result (this is a single integer) into `erfahrungen.txt`. *(2 Points)*