Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn
P. Bamberger
P. Schneider

Algorithmen und Datenstrukturen
Sommersemester 2020
Musterlösung Übungsblatt 8

Abgabe: Mittwoch, 08.07.2020, 16:00 Uhr.

Aufgabe 1: Maschinenbelegungsplanung - Teil 1 *(12 Punkte)*

Gegeben seien n Jobs, durchnummeriert von 0 bis $n-1$. Diese sollen von einer Maschine bearbeitet werden. Manche Jobs sind voneinander abhängig. Ist ein Job j von einem anderen Job i abhängig, so muss i vor j ausgeführt werden.
In dieser Aufgabe sollen Sie einen Algorithmus entwerfen, der als Eingabe die Anzahl n der Jobs, sowie eine Auflistung von Abhängigkeiten zwischen Jobs erhält und eine gültige Bearbeitungsreihenfolge (ohne Verletzung einer Abhängigkeit) ausgibt.
Gibt es keine gültige Bearbeitungsreihenfolge soll der Algorithmus dies erkennen und eine entsprechende Ausgabe (bspw. `None`) machen. Die Abhängigkeiten zwischen Jobs sind gegeben als *gerichteter* Graph $G = (V, E)$ mit Knoten $V = \{0, \dots, n-1\}$ und Kante $(i, j) \in E$, falls Job j von Job i abhängt.

(a) Implementieren Sie einen Algorithmus mit der beschriebenen Funktion dessen Laufzeit höchstens linear in der Größe von G ist. Nehmen Sie an, dass der Abhängigkeitsgraph als Adjazenzliste vorliegt. Sie können die Vorlage `Scheduling.py` benutzen. Eine Klasse zur Graphenrepräsentation als *AdjacencyList* mit den zugehörigen Funktionalitäten ist in `AdjacencyList.py` definiert. *(8 Punkte)*

(b) Lesen Sie den Abhängigkeitsgraph aus der Datei `dag.txt` in eine Adjazenzliste ein (Sie können dazu die Funktion `read_graph_from_file` aus `Scheduling.py` benutzen) und führen Sie Ihren Algorithmus darauf aus. Kopieren Sie das Ergebnis in Ihre `erfahrungen.txt`. *(4 Punkte)*

Musterlösung

(a) Siehe `Scheduling.py` der Musterlösung.

(b) Eine mögliche Bearbeitungsreihenfolge ist (vgl. Abbildung 1):

```
[35, 23, 7, 31, 10, 1, 0, 12, 45, 15, 6, 11, 48, 3, 18, 26, 4, 24, 14, 42, 27, 47,
33, 16, 41, 19, 37, 22, 49, 34, 20, 29, 5, 17, 8, 25, 13, 36, 43, 21, 40, 9, 46, 44,
39, 32, 2, 28, 38, 30]
```

Aufgabe 2: Maschinenbelegungsplanung - Teil 2 *(8 Punkte)*

Wir betrachten das gleiche Problem wie in Aufgabe 1 mit dem Unterschied, dass jeder Job nun von höchstens einem anderen Job abhängt (d.h. jeder Job hat Eingangsgrad ≤ 1 im Abhängigkeitsgraphen) und dass eine gültige Bearbeitungsreihenfolge exisitiert, d.h. der Abhängigkeitsgraph ist zyklenfrei.
Angenommen es stehen beliebig viele Maschinen zur Verfügung, welche die Jobs parallel ausführen können. Ziel ist es für jeden Job zu bestimmen, wann dieser frühestens ausgeführt wird, wenn man

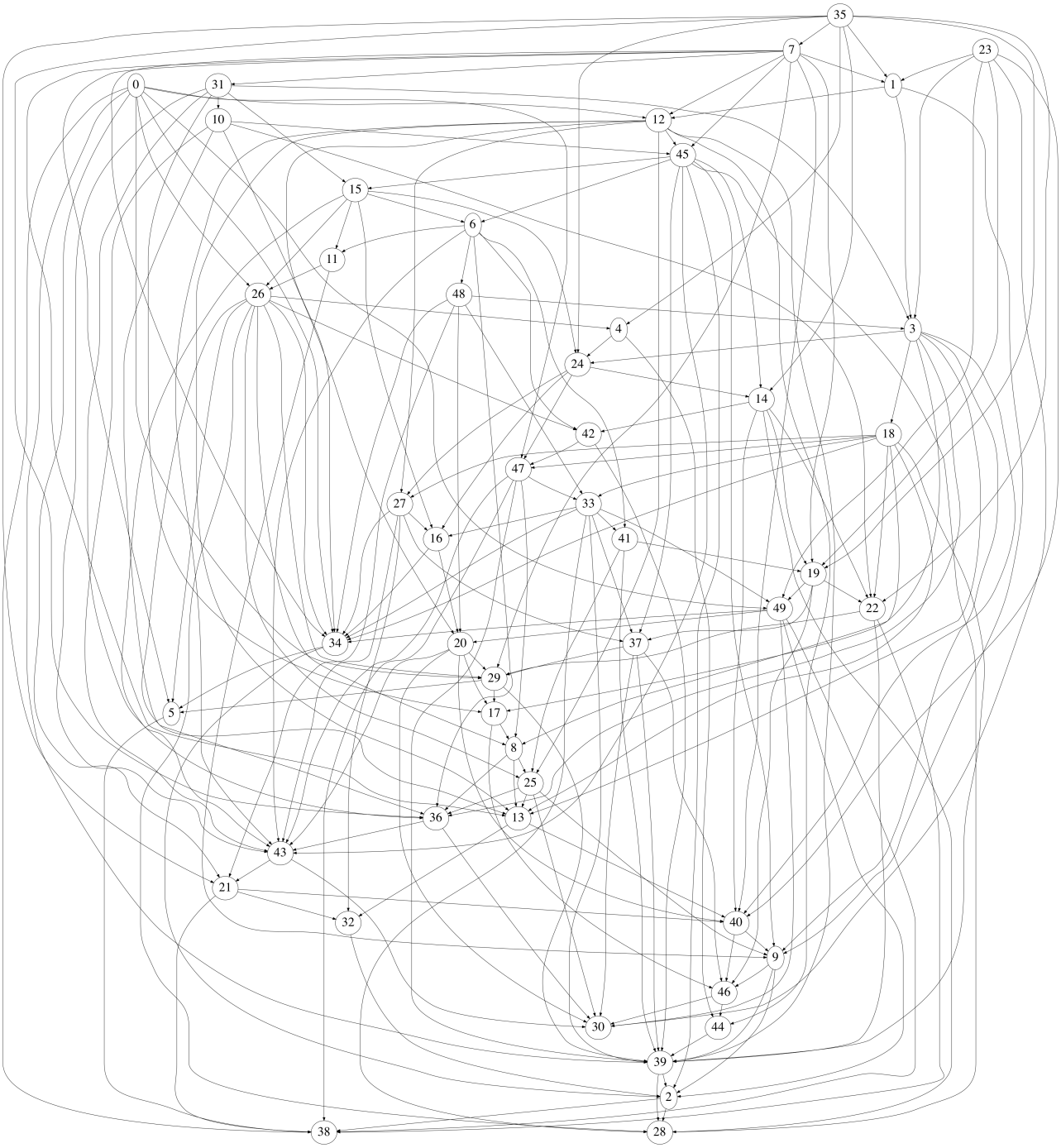

Abb. 1: Abbildung des Graphen aus `dag.txt`.

zum Zeitpunkt 0 startet und annimmt, dass eine Maschine einen Job in einem Zeitschritt erledigen kann. Die Ausgabe soll also ein Array der Länge n sein, wobei Eintrag i den Zeitpunkt angibt, zu dem Job i frühestens ausgeführt werden kann.

(a) Implementieren Sie einen Algorithmus mit der beschriebenen Funktion mit Laufzeit höchstens linear in der Größe des Abhängigkeitsgraphen. Nehmen Sie an, dass der Abhängigkeitsgraph als Adjazenzliste vorliegt. *(6 Punkte)*

(b) Lesen Sie den Abhängigkeitsgraph aus der Datei `forest.txt` in eine Adjazenzliste ein (dieser Graph hat die Eigenschaft, dass jeder Knoten Eingangsgrad ≤ 1 hat). Nutzen Sie Ihren Algorithmus aus (a), um den frühstmöglichen Zeitpunkt zu bestimmen, an dem alle Jobs erledigt sind. Schreiben Sie das Ergebnis (dies ist ein einzelner Integer) in Ihre `erfahrungen.txt`. *(2 Punkte)*

Musterlösung

(a) Siehe `Scheduling.py` der Musterlösung.

(b) Der Job mit der größten Tiefe im Abhängigkeitsgraphen hat Tiefe 16. Alle Jobs können also frühestens nach 17 Zeitschritten beendet sein (vgl. Abbildung 2).

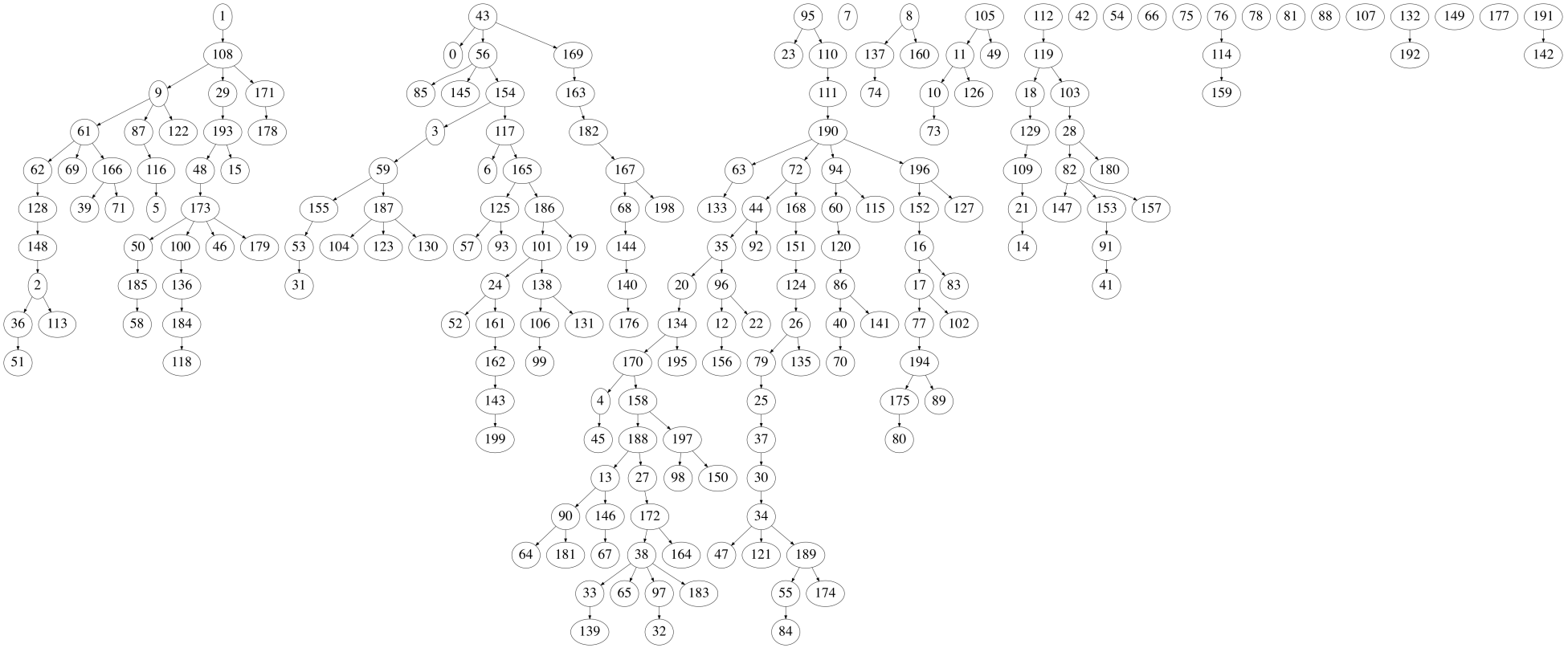

Abb. 2: Abbildung des Graphen aus `forest.txt`.