10. Übung (13.–17. Dezember 2021)

Algorithmen und Datenstrukturen

Aufgabe 1 (AGS 3.2.45)

Gegeben sei ein gerichteter, azyklischer Graph $G = (V, E)$ mit $V = \{0, \ldots, \mathtt{n}-1\}$ für ein $\mathtt{n} \geq 1$, d.h. $V \subseteq \mathtt{int}$. Das *topologische Sortieren von* G ist die Berechnung einer bijektiven Abbildung $\mathrm{ord}\colon V \to \{1, \ldots, \mathtt{n}\}$, so dass aus $(v, v') \in E$ folgt $\mathrm{ord}(v) < \mathrm{ord}(v')$.
Programmieren Sie eine Funktion `topsort` in C, die einen beliebigen gerichteten, azyklischen Graph G topologisch sortiert. Dazu seien die Kanten von G durch den Datentyp `struct Edge` dargestellt:

```
struct Edge { int from, to; };
```

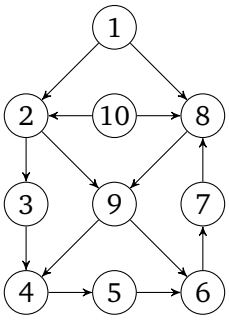
Die Menge E aller Kanten von G ist durch ein Array `edges` über dem Typ `struct Edge` gegeben. Vervollständigen Sie die Funktion `topsort`, so dass die Funktionswerte der Abbildung ord in das Array `ord` eingetragen werden (d.h. $\mathrm{ord}(\mathtt{v}) = \mathtt{ord[v]}$). Weiterhin gelte:

- `e` ist die Länge von `edges`.
- `ord` ist ausreichend lang und wurde für jeden Knoten mit dem Wert -1 initialisiert.

```
void topsort(int n, int e, struct Edge edges[], int ord[]) { ... }
```

Aufgabe 2 (AGS 9.2.11)

Der gerichtete Graph G sei durch folgende Darstellung gegeben:

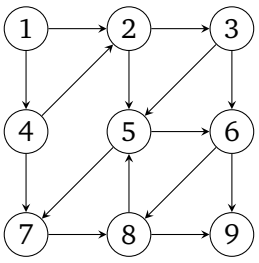


(a) Wenden Sie auf G wiederholt den DFS-Algorithmus mit dem Startknoten 1 an und bestimmen Sie auf diese Weise drei unterschiedliche *depth-first trees*.

(b) Wenden Sie auf G wiederholt den BFS-Algorithmus mit dem Startknoten 1 an und bestimmen Sie auf diese Weise drei unterschiedliche *breadth-first trees*.

Zusatzaufgabe 1 (AGS 9.2.15 ⋆)

Der gerichtete Graph G sei durch die Darstellung rechts gegeben. Zwischenschritte zu den Lösungen brauchen Sie nicht anzugeben.



(a) Wenden Sie auf G wiederholt den DFS-Algorithmus mit dem Startknoten 2 an und bestimmen Sie auf diese Weise 3 unterschiedliche depth-first-trees.

(b) Wenden Sie auf G wiederholt den BFS-Algorithmus mit dem Startknoten 2 an und bestimmen Sie auf diese Weise 3 unterschiedliche breadth-first-trees.