

Übungen zur Vorlesung Algorithmen und Datenstrukturen

Blatt 7

Aufgabe P-18:

- Geben Sie die Adjazenzmatrix und die Adjazenzlisten des in Abbildung 1 gezeigten Graphen an.
- Zeigen Sie den Ablauf der Tiefensuche auf dem Graphen in Abbildung 1. Nehmen Sie dabei an, dass in der äußeren Schleife die Knoten in alphabetischer Reihenfolge bearbeitet werden, und dass alle Adjazenzlisten alphabetisch angeordnet sind.

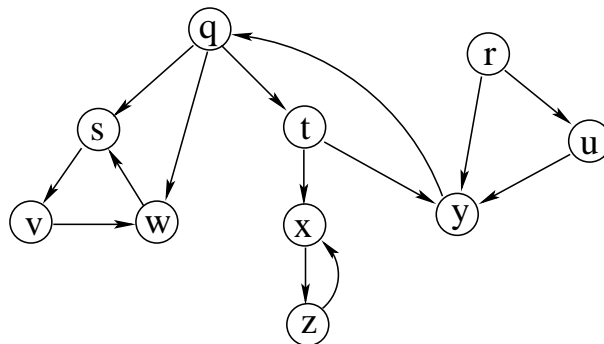


Abbildung 1: Ein gerichteter Graph

- Geben Sie für jeden Knoten die *discovery time* und *finishing time* an, und klassifizieren Sie die Kanten des Graphen.
- Entfernen Sie aus dem Graphen die Kanten (y, q) , (z, x) und (w, s) . Geben Sie eine topologische Sortierung des entstehenden Graphen an unter Verwendung der *finishing time*.

Aufgabe P-19: Tiefensuche kann verwendet werden, um einen ungerichteten Graphen in seine Zusammenhangskomponenten zu zerlegen: Modifizieren Sie

den Algorithmus DFS so, dass er für jeden Knoten v eine Zahl $cc[v]$ zwischen 1 und k , wobei k die Zahl der Zusammenhangskomponenten des Graphen ist, mit ausgibt, derart dass $cc[u] = cc[v]$ genau dann gilt, wenn u und v in der selben Komponente liegen.

Aufgabe H-21:

- a) Lösen sie das Problem aus der vorigen Aufgabe unter Verwendung der Datenstruktur UNION-FIND, d.h. zeigen Sie, wie für einen ungerichteten Graphen $G = (V, E)$ ein effizienter Algorithmus $areConnected(u, v)$ gewonnen werden kann, der für $u, v \in V$ entscheidet, ob es in G einen Pfad zwischen u und v gibt. Wie komplex ist $areConnected(u, v)$ und wie komplex ist der Aufbau der Datenstruktur zuvor?
- b) Vergleichen Sie die Komplexität mit der der Lösung, die Tiefensuche verwendet. Welchen Vorteil hat der Algorithmus aus a) mit UNION-FIND?

Aufgabe H-22: Eine andere Möglichkeit, einen gerichteten azyklischen Graphen topologisch zu sortieren, ist die folgende: man sucht einen Knoten vom Eingangsgrad 0, gibt diesen aus, und entfernt ihn und alle von ihm ausgehenden Kanten, und wiederholt dies bis der Graph leer wird. Der Graph sei als Liste von Knoten v_1, \dots, v_n und Liste von Kanten e_1, \dots, e_m , $e_i = (v_{i_1}, v_{i_2})$ gegeben. Geben Sie einen Algorithmus an, der diese Idee umsetzt und in Zeit $O(|V| + |E|)$ läuft.

Abgabe bis Donnerstag, 2. Juli, 14.00 Uhr im dafür vorgesehenen Briefkasten in der Theresienstraße.