Rolf Wanka Erlangen, 23. Oktober 2019

Übungen zur Vorlesung

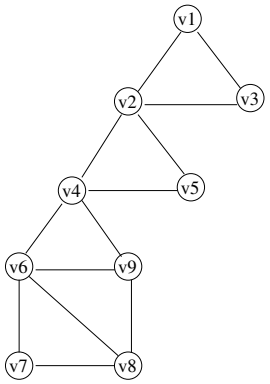
Effiziente kombinatorische Algorithmen

WS 2019/20

Blatt 1

AUFGABE 1:

Gegeben sei der links dargestellte Graph G.



(a) Stellen Sie die Inzidenzlisten für die einzelnen Knoten auf. Die Reihenfolge ist durch die Knotennumerierung gegeben: Sind zwei Knoten v_i und v_j Elemente derselben Liste, so erscheint v_i vor v_j genau dann, wenn $i < j$.

(b) Wenden Sie den aus der Vorlesung bekannten Tiefensuchalgorithmus für ungerichtete Graphen (also den Algorithmus $\mathrm{DFS_u}$) auf den Graphen G an, d. h. geben Sie jeweils dfnr, T und B an.

(i) Verwenden Sie v_1 als Startknoten.

(ii) Verwenden Sie v_6 als Startknoten.

AUFGABE 2:

(a) Geben Sie einen Algorithmus in Pseudocode an, der einen durch seine Inzidenzlisten gegebenen Graphen $G = (V,E)$ folgendermaßen traversiert (Breitensuche) und dabei die Menge der benötigten Kanten T und die Besuchsreihenfolge der Knoten *bfnr* als Array mit Einträgen aus $1 \ldots |V|$ ausgibt:

- Beginne bei einem beliebigen Knoten $r \in V$.
- Besuche zunächst alle Nachbarn von r, besuche dann die Nachbarn der Nachbarn (falls sie noch nicht besucht wurden), usw.

Hinweis: Nutzen Sie dabei aus, dass Sie Knoten mit BESUCHT/UNBESUCHT-Flags versehen können.

(b) Wenden Sie Ihren Algorithmus auf den Graphen aus Aufgabe 1 an.

i) Verwenden Sie v_1 als Startknoten.

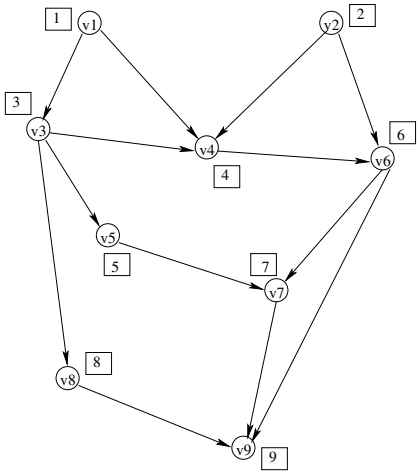
ii) Verwenden Sie v_6 als Startknoten.

AUFGABE 3:

Gegeben ist ein Digraph $G = (V,E)$ mit der Knotenmenge $V = \{v_1, \ldots, v_n\}$. Jedem Knoten v_i ist eine natürliche Zahl m_i zugeordnet. Die Knoten $v_1, \ldots, v_n$ heißen *topologisch sortiert* genau dann, wenn für jede Kante $e = (v_i, v_j) \in E$ gilt: $m_i < m_j$.
Auf der nächsten Seite finden Sie ein Beispiel für einen topologisch sortierten Graphen.
Bemerkung: Ein Digraph kann mehrere topologische Sortierungen haben.

(a) Finden Sie ein Gegenbeispiel, warum die Ausgabe *dfnr* des in der Vorlesung besprochenen Tiefensuchalgorithmus in der Regel keine topologische Sortierung definiert.

(b) Finden Sie ein Gegenbeispiel, warum die Ausgabe *bfnr* ihres Breitensuchalgorithmus aus Aufgabe 2 in der Regel keine topologische Sortierung definiert.



(c) Geben Sie informell einen Algorithmus an, um eine topologische Sortierung zu berechnen, und wenden Sie Ihren Algorithmus auf obigen Graphen an.

(d) Ermitteln Sie die Laufzeit Ihres Algorithmus.

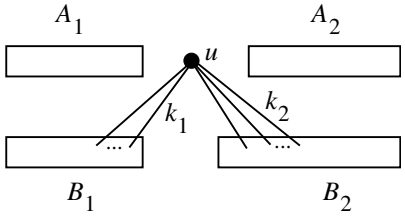
Zur Vorbereitung der Aufgabe 4 definieren wir im folgenden einige weitere graphentheoretische Begriffe:

Sei $G = (V, E)$ ein ungerichteter Graph. Der *Grad* Δ_G von G ist der maximal auftretende Knotengrad, d. h. $\Delta_G = \max\{d_G(u) \mid u \in V\}$. G heißt *regulär* vom Grad Δ_G, wenn alle Knoten den Grad Δ_G haben. G heißt *bipartit*, wenn man die Knoten des Graphen in zwei disjunkte Mengen A und B derart aufteilen kann (Schreibweise: $V = A \dot\cup B$), daß für alle Kanten $\{u, v\} \in E$ gilt: $u \in A \wedge v \in B$.

AUFGABE 4:
Sei $G = (A \dot\cup B, E)$ ein zusammenhängender, regulärer, bipartiter Graph vom Grad k, $k \geq 2$.

Zeigen Sie: G ist zweifach zusammenhängend.

Hinweis: Nehmen Sie an, G wäre zusammenhängend, aber nicht zweifach zusammenhängend, d. h. es gibt mindestens einen Artikulationspunkt u. OBdA sei $u \in A$. Betrachten Sie folgende Abbildung.



Knoten u trennt also $A_1 \dot\cup B_1$ von $A_2 \dot\cup B_2$. Beantworten Sie folgende Fragen:

(a) Wieviele Kanten verlassen A_1 in Richtung B_1 ?

(b) Wieviele Kanten verlassen B_1 in Richtung A_1 ?

(c) Geben Sie mit den Ergebnissen aus (a) und (b) $|B_1|$ in Abhängigkeit von $|A_1|$ an und finden Sie den Widerspruch.