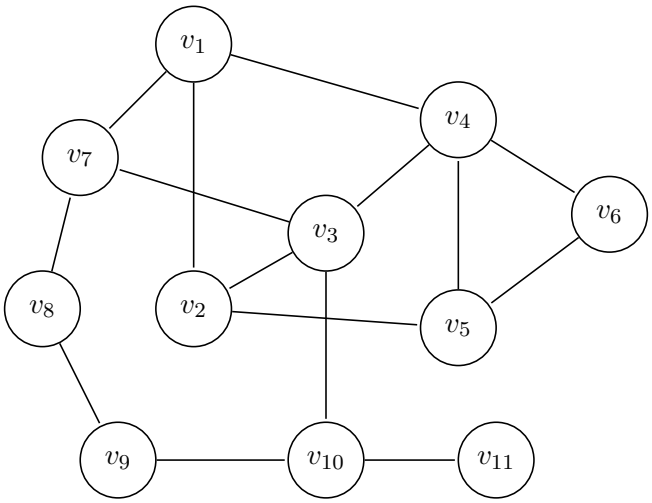
Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn
M. Fuchs, G. Schmid

Algorithmen und Datenstrukturen
Sommersemester 2024
Musterlösung Übungsblatt 8

Abgabe: Dienstag, 18. Juni, 2024, 10:00 Uhr

Aufgabe 1: Breitensuche *(5 Punkte)*

Gegeben folgender ungerichteter Graph G:



a) Geben Sie G als Adjazenzmatrix an. *(1 Punkt)*

b) Geben Sie G als Adjazenzliste an. *(1 Punkt)*

c) Führen Sie eine Breitensuche auf G beginnend auf Knoten v_1 aus. Schreiben Sie dafür die Reihenfolge in welcher die Knoten im Algorithmus markiert (bzw. grau gefärbt werden). Um ein deterministisches Resultat zu bekommen, fügen wir immer den Knoten als nächstes in die FIFO-QUEUE der den kleineren Index hat, sprich, v_i vor v_j wenn $i < j$. *(3 Punkte)*

Musterlösung

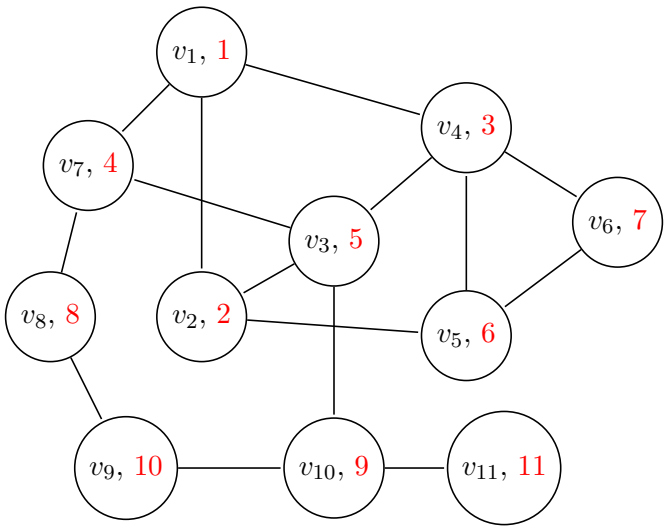
a)

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
v_1	0	1	0	1	0	0	1	0	0	0	0
v_2	1	0	1	0	1	0	0	0	0	0	0
v_3	0	1	0	1	0	0	1	0	0	1	0
v_4	1	0	1	0	1	1	0	0	0	0	0
v_5	0	1	0	1	0	1	0	0	0	0	0
v_6	0	0	0	1	1	0	0	0	0	0	0
v_7	1	0	1	0	0	0	0	1	0	0	0
v_8	0	0	0	0	0	0	1	0	1	0	0
v_9	0	0	0	0	0	0	0	1	0	1	0
v_{10}	0	0	1	0	0	0	0	0	1	0	1
v_{11}	0	0	0	0	0	0	0	0	0	1	0

b)

- $v_1 : v_2, v_4, v_7$
- $v_2 : v_1, v_3, v_5$
- $v_3 : v_2, v_4, v_7, v_{10}$
- $v_4 : v_1, v_3, v_5, v_6$
- $v_5 : v_2, v_4, v_6$
- $v_6 : v_4, v_5$
- $v_7 : v_1, v_3, v_8$
- $v_8 : v_7, v_9$
- $v_9 : v_8, v_{10}$
- $v_{10} : v_3, v_9, v_{11}$
- $v_{11} : v_{10}$

c)



Aufgabe 2: Tiefensuche *(6 Punkte)*

Wir definieren für jeden Knoten 2 Zeitpunkte (wie in Folie 29):

- $t_{v,1}$: Zeitpunkt wenn Knoten v von der DFS-Suche grau gefärbt wird

- $t_{v,2}$: Zeitpunkt wenn Knoten v von der DFS-Suche schwarz gefärbt wird

Sei ausserdem folgender *gerichteter* Graph $G = (V, E)$ gegeben mit

- $V = \{u_1, u_2, u_3, u_4, u_5\}$
- $E = \{(u_1, u_2), (u_1, u_3), (u_2, u_3), (u_3, u_4), (u_4, u_1), (u_5, u_1), (u_5, u_3), (u_5, u_4)\}$

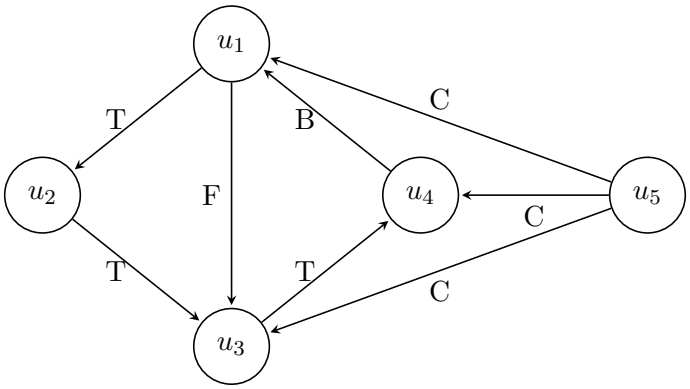
a) Zeichnen Sie G. *(2 Punkte)*

b) Schreiben Sie zu jedem Knoten in G das Bearbeitungsintervall $[t_{v,1}, t_{v,2}]$. Ähnlich wie in Aufgabe 1c), wenn mehrere Knoten als nächstes von der Tiefensuche besucht werden könnten, wird immer derjenige mit kleinstem Index gewählt (und wir starten somit auch bei u_1). *(2 Punkte)*

c) Schreiben Sie zu jeder Kante ob es sich um eine Baumkante, Rückwärtskante, Vorwärtskante oder eine Querkanten handelt. *(2 Punkte)*

Musterlösung

ac) Wir bennenen Baumkanten mit T (Tree Edge), Rückwärtskanten mit B (Backward Edge), Vorwärtskanten mit F (Forward Edge) und Querkanten mit C (Cross Edge).



b)
- $u_1 : [1, 8]$
- $u_2 : [2, 7]$
- $u_3 : [3, 6]$
- $u_4 : [4, 5]$
- $u_5 : [9, 10]$

Aufgabe 3: Zyklenfreie Graphen *(9 Punkte)*

a) Wie viele Kanten m kann ein ungerichteter zusammenhägender Graph mit n Knoten maximal haben? Begründen Sie ihre Antwort. *(2 Punkte)*

b) Zeigen Sie, dass jeder ungerichteter zusammenhägender Graph welcher keinen Zyklus[1] enthält, genau $n - 1$ Kanten hat (wobei n die Anzahl der Knoten des Graphen ist). *(4 Punkte)*
Hinweis: Sie können diese Aussage zum Beispiel per Induktion über $n \geq 1$ zeigen.

c) Gegeben ein ungerichteter zusammenhängender Graph $G = (V, E)$ mit $n = |V|$. Geben Sie einen Algorithmus an der in Zeit $O(n)$ entscheidet ob G einen Zyklus enthält oder nicht. Spezifizieren Sie dafür explizit in welcher Datenstruktur G gegeben sein soll. *(3 Punkte)*

[1] Ein Zyklus (oder auch Kreis) ist ein Pfad $v_1, \ldots, v_k \in V$ in einem Graphen bei dem zusätlich eine Kante zwischen dem Start und dem Endknoten existiert, also $\{v_1, v_k\} \in E$.

Musterlösung

a) Ein Graph hat die maximale Anzahl von Kanten wenn jeder Knoten mit jedem Verbunden ist. Bedeutet als jeder Knoten hat den Grad $n-1$. Wir fixen nun einer Reihenfolge der Knoten $v_1, ..., v_n$ und zählen jeweils die "noch nicht gezählten" Kanten. Da hat v_1 gerade $n-1$ viele, v_2 hat noch $n-2$ viele (da die Kante zwischen v_1 und v_2 ja schon gezählt wurde), v_3 hat $n-3$ viele usw. Es gilt also:

$$m \leq \sum_{i=1}^{n}(n-i) = \sum_{i=1}^{n-1} i = \frac{(n-1)\cdot n}{2}$$

Ein anderer Ansatz wäre zu berechnen wie viele 2 elementige Teilmengen es von einer n elementigen Menge gibt. Davon gibt es gerade $\binom{n}{2} = \frac{n!}{2!\cdot(n-2)!} = \frac{n\cdot(n-1)}{2!} = \frac{(n-1)\cdot n}{2}$.

b) Ein zusammenhängender Graph ohne Zyklus hat genau $n-1$ Kanten. Beweis per Induktion.
Induktionsanfang: Für $n = 1$ hat der Graph keine Kante.
Induktionsvoraussetzung: Jeder solcher Graph mit $k \leq n-1$ Knoten hat $k-1$ Kanten.
Induktionsschritt: Wir zeigen nun dass die Voraussetzung auch für einen n-Knoten Graphen G wahr ist. Jeder Graph G mit n Knoten kann zusammengesetzt werden aus einem Knoten v der verbunden ist mit $l \geq 1$ disjunkten Subgraphen $G_1, ..., G_l$ von G. Da G zyklenfrei ist, ist auch jeder dieser Subgraphen zyklenfrei und die einzige Verbindung zwischen 2 Subgraphen ist über den Knoten v. Ohne Beschränkung der Allgemeinheit, sagen wir dass G_i grade n_i Knoten hat (für jeden dieser Subgraphen). Da $n_i \leq n-1$ für alle i, gilt durch die Induktionsvoraussetzung dass G_i gerade $n_i - 1$ Kanten hat. Wir können nun die Anzahl der Kanten m in G wie folgt berechnen:

$$m = deg(v) + \sum_{i=1}^{l}(n_i - 1) = l + \sum_{i=1}^{l} n_i - \sum_{i=1}^{l} 1 = \sum_{i=1}^{l} n_i = n-1$$

Dabei gilt $deg(v) = l$, da v ja gerade mit jedem der l Subgraphen verbunden ist und es gilt $\sum_{i=1}^{l} n_i = n-1$ da dies ja gerade die Summe über alle Knoten in G ohne v repräsentiert.

c) Diese Aufgabe könnte man prinzipiell sowohl mit Tiefen- wie auch Breitensuche lösen. Wir nutzen hier die Breitensuche und gehen davon aus dass G als Adjazenliste vorliegt. Wir führen die Breitensuche "normal" aus, speichern uns aber für jeden Knoten v, von welchem Knoten u aus er zu erst erreicht wurde. Diesen Knoten u nennen wir **parent** von v. Wenn nun v einen markierten Nachbar hat, der nicht sein parent ist, dann existiert ein Zyklus im Graph und wir geben *false* zurück. Diese Prozedur hat die selbe Laufzeit wie die Breitensuche, also $O(n+m)$. Wenn m nun wie in Aufgabe *a*) gerade $O(n^2)$ ist, ist die Laufzeit natürlich zu schlecht. Wir wissen aber von *b*), dass wenn G zyklenfrei ist, er nur $n-1$ Kanten hat. Wir können also die Prozedur nach $n-1$ Schritten abbrechen und *false* zurückgeben sollte es noch unbesuchte Knoten in der Queue geben. Die Laufzeit ist somit $O(n)$.