Prof. Dr. Leif Kobbelt
Stefan Dollase, Ira Fesefeldt, Alexandra Heuschling, Gregor Kobsik

Lösung - Übung 9

Tutoraufgabe 1 (Optimaler Suchbaum):

Gegeben sind folgende Knoten mit dazugehörigen Zugriffswahrscheinlichkeiten:

Knoten	l_0	N_1	l_1	N_2	l_2	N_3	l_3	N_4	l_4
Wert	$(-\infty,1)$	1	(1,2)	2	(2,3)	3	(3,4)	4	$(4,\infty)$
Wahrscheinlichkeiten	0.1	0.1	0.1	0.2	0.2	0.1	0.1	0.1	0.0

Konstruieren Sie einen optimalen Suchbaum wie folgt.

a) Füllen Sie untenstehende Tabellen für $W_{i,j}$ und $C_{i,j}$ nach dem Verfahren aus der Vorlesung aus. Geben Sie in $C_{i,j}$ ebenfalls **alle möglichen Wurzeln** des optimalen Suchbaums für $\{i,\dots,j\}$ an.

$W_{i,j}$	0	1	2	3	4
1					
2	–				
3	–	–			
4	–	–	–		
5	–	–	–	–	

$C_{i,j}$ $(R_{i,j})$	0	1	2	3	4
1		( )	( )	( )	( )
2	–		( )	( )	( )
3	–	–		( )	( )
4	–	–	–		( )
5	–	–	–	–	

b) Geben Sie einen optimalen Suchbaum für die Knoten mit den gegebenen Zugriffswahrscheinlichkeiten und der gegebenen Reihenfolge der Knoten graphisch an.

c) Ist der optimale Suchbaum für die Knoten mit den gegebenen Zugriffswahrscheinlichkeiten und der gegebenen Reihenfolge der Knoten eindeutig? Geben Sie dazu eine kurze Begründung an.

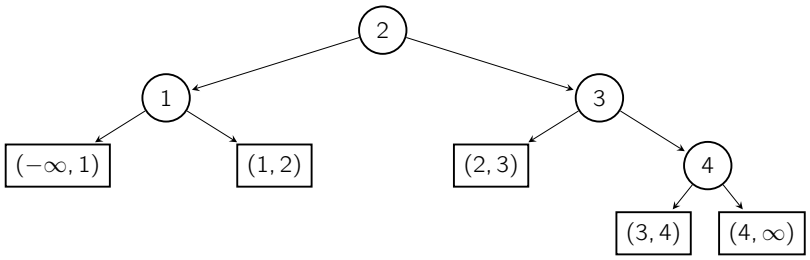
Lösung

a)

$W_{i,j}$	0	1	2	3	4
1	0.10	0.30	0.70	0.90	1.00
2	–	0.10	0.50	0.70	0.80
3	–	–	0.20	0.40	0.50
4	–	–	–	0.10	0.20
5	–	–	–	–	0.00

$C_{i,j}(R_{i,j})$	0	1	2	3	4
1	0.10	0.50 (1)	1.40 (2)	2.10 (2)	2.50 (2)
2	–	0.10	0.80 (2)	1.50 (2)	1.90 (2, 3)
3	–	–	0.20	0.70 (3)	1.00 (3)
4	–	–	–	0.10	0.30 (4)
5	–	–	–	–	0.00

b) Die folgenden Lösungen sind korrekte optimale Suchbäume.



c) Der Optimale Suchbaum ist eindeutig, denn bei der Konstruktion von $C_{i,j}$ war jedes relevante Minimum eindeutig.

Tutoraufgabe 2 (Union Find):

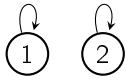
Führen Sie die folgenden Operationen beginnend mit einer anfangs leeren *Union-Find-Struktur* aus und geben Sie die entstehende Union-Find-Struktur nach jeder *MakeSet*, *Union* und *Find* Operation an. Nutzen Sie dabei die beiden Laufzeitverbesserungen: Höhenbalencierung und Pfadkompression. Dabei soll die Union-Operation bei **gleicher Höhe der Wurzeln immer die Wurzel des zweiten Parameters** als neue Wurzel wählen. Es ist nicht notwendig die Höhe der Bäume zu notieren.

1. MakeSet(1)
2. MakeSet(2)
3. MakeSet(3)
4. MakeSet(4)
5. MakeSet(5)
6. MakeSet(6)
7. MakeSet(7)
8. MakeSet(8)
9. MakeSet(9)
10. Union(1,2)
11. Union(3,4)
12. Union(3,1)
13. Union(5,6)
14. Union(7,8)
15. Union(7,9)
16. Union(9,5)
17. Union(9,2)
18. MakeSet(10)
19. Union(7,10)
20. Find(3)

Lösung

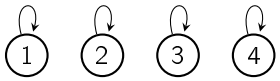
1. MakeSet(1)

2. MakeSet(2)



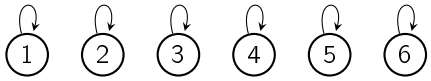
3. MakeSet(3)

4. MakeSet(4)

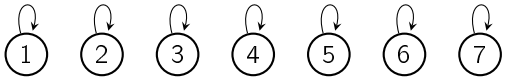


5. MakeSet(5)

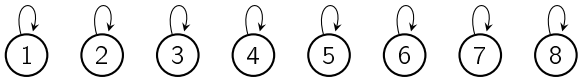
6. MakeSet(6)



7. MakeSet(7)

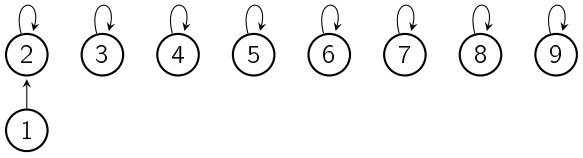


8. MakeSet(8)

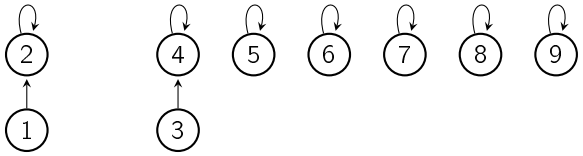


9. MakeSet(9)

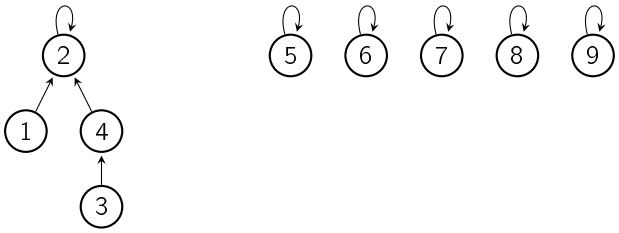
10. Union(1,2)
    - Find(1): unverändert
    - Find(2): unverändert
    - Union(1,2):



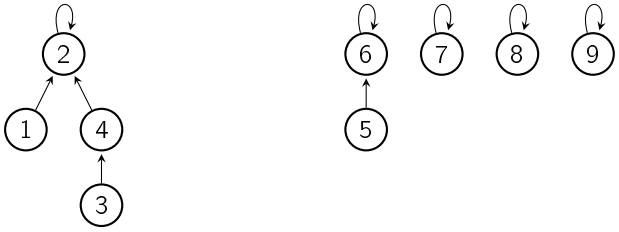
11. Union(3,4)
    - Find(3): unverändert
    - Find(4): unverändert
    - Union(3,4):



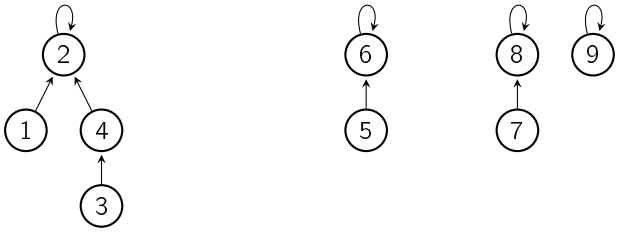
12. Union(3,1)
    - Find(3): unverändert
    - Find(1): unverändert
    - Union(3,1):



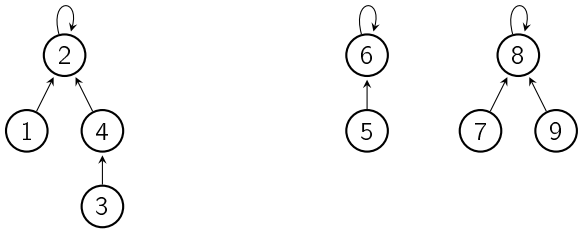
13. Union(5,6)
    - Find(5): unverändert
    - Find(6): unverändert
    - Union(5,6):



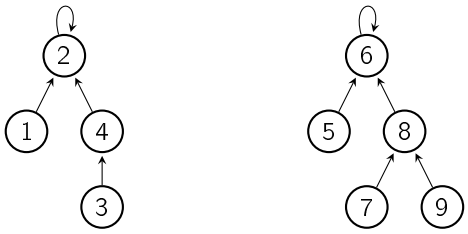
14. Union(7,8)
    - Find(7): unverändert
    - Find(8): unverändert
    - Union(7,8):



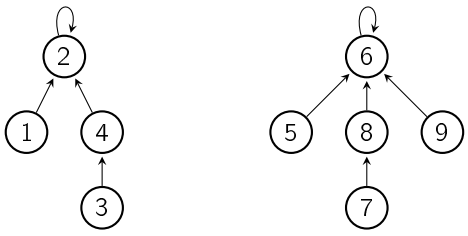
15. Union(7,9)
    - Find(7): unverändert
    - Find(9): unverändert
    - Union(7,9):



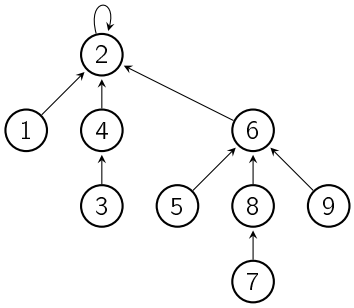
16. Union(9,5)
    - Find(9): unverändert
    - Find(5): unverändert
    - Union(9,5):



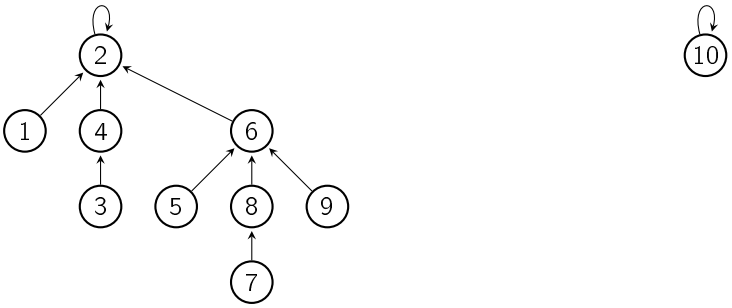
17. Union(9,2)
    - Find(9):



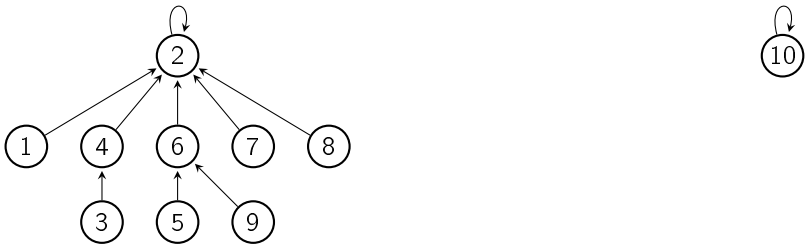
    - Find(2): unverändert
    - Union(9,2):



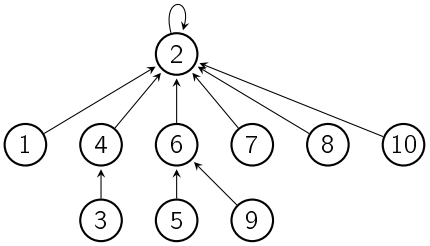
18. MakeSet(10)



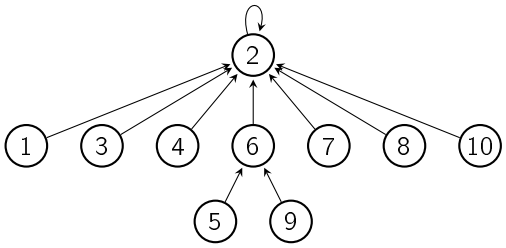
19. Union(7,10)
    - Find(7):



    - Find(10): unverändert
    - Union(7,10):



20. Find(3)



Tutoraufgabe 3 (Prominenz suchen):

Sei ein gerichteter Graph $G = (V, E)$ als Adjazenzmatrix gegeben. Wir nennen einen Knoten $v \in V$ prominent, wenn von allen Knoten $v' \in V \setminus \{v\}$ eine Kante $(v', v) \in E$ nach v existiert, aber es von keinem Knoten $v' \in V$ eine Kante $(v, v') \notin E$ zurück gibt.
Geben Sie einen Algorithmus an, der in $O(|V|)$ Worst-case Laufzeit herausfindet, ob G einen prominenten Knoten besitzt. Begründen Sie die Korrektheit und die Laufzeit Ihres Algorithmus.

Lösung

Da G als Adjazenzmatrix gegeben ist, können wir die Senke durch geschicktes Erkunden der Adjazenzmatrix finden. Angenommen die Adjazenzmatrix ist eine $n \times n$ Matrix und von 1 bis n indiziert.
Wir gehen wie folgt vor: Wir starten bei der Position $(1, n)$ in der Adjazenzmatrix. Ist der Wert an der aktuellen Position (i, j) eine 1 (also ist $(i, j) \in E$), dann erhöhen wir die Zeile um eins. Ist dagegen der Wert an der aktuellen Position (i, j) eine 0 (also ist $(i, j) \notin E$), dann verringern wir die Spalte um eins. Sind wir irgendwann bei einer Position $(k, 1)$ mit Wert 0 angekommen, dann ist entweder Knoten k eine Senke, oder es existiert kein prominenter Knoten. Welcher dieser beiden Fälle zutrifft überprüfen wir anschließend, indem wir die Zeile k auf nicht-null Einträge und die Spalte k auf nicht-eins Einträge prüfen.
In Code sähe dies wie folgt aus:

```
def find_sink(E,n):
    (i,j) = (1,n)
    #bilinear search for sink
    while not (E[i][j] == 0 and j == 1):
        if E[i][j] == 1:
            i += 1
        else:
            j -= 1
    k = i

    #check if k has no out edges
    for i in range(1,n):
        if E[k][i] != 0:
            return False

    #check if k has all in edges
    for i in range(1,n):
        if E[i][k] != 1 and i != k:
            return False

    return k
```

Die Idee hier ist, dass ein prominenter Knoten eine 0 Zeile und eine 1 Spalte besitzt, das heißt eine Matrix dieser Form:

$$\begin{pmatrix} \ddots & \ddots & 1 & \ddots & \ddots \\ \ddots & \ddots & \vdots & \ddots & \ddots \\ \ddots & \ddots & 1 & \ddots & \ddots \\ 0 & \cdots & 0 & \cdots & 0 \\ \ddots & \ddots & 1 & \ddots & \ddots \\ \ddots & \ddots & \vdots & \ddots & \ddots \\ \ddots & \ddots & 1 & \ddots & \ddots \end{pmatrix}$$

Dieses Kreuz lässt sich finden, in dem wir die horizontale 0 Reihe suchen. Daher gehen wir stets nach links wenn wir auf eine null treffen. Damit werden wir auch irgendwann auf die 0 Reihe treffen. Treffen wir dagegen auf eine 1 gehen wir nach unten, da die derzeitige Spalte nicht die 0 Reihe sein kann.
Zwecks Korrektheit müssen wir zuerst einsehen, dass es stets maximal einen prominenten Knoten geben kann:

Gäbe es zwei, seien diese beide i und j müsste es auch eine Kante $(i,j) \in E$ geben, damit j prominent ist. Dann kann aber i nicht mehr prominent sein, da es eine ausgehende Kante gibt.
Nun nehmen wir zuerst an, dass es keine prominente Knoten gibt. Dann wird der Algorithmus bei welchem Knoten auch immer er glaubt einen prominenten Knoten gefunden zu haben, feststellen, dass es nicht dem 0-1-Kreuz entspricht und daher auch kein prominenter Knoten ist.
Nehmen wir nun an, dass es genau einen prominenten Knoten k gibt. Dann muss der Algorithmus irgendwann entweder auf einen Eintrag (i,k) mit $i < k$ oder einen Eintrag (k,j) mit $j > k$ treffen. Dies gilt, da wir in jeder Iteration nur einen Eintrag um eines ändern. Landen wir auf einem Eintrag (i,k) erhöhen wir das i solange bis wir bei (k,k) angekommen sind. Von da werden wir solange den zweiten Eintrag senken bis wir bei $(k,1)$ angekommen sind. Landen wir bei auf einem Eintrag (k,j) senken wir das j solange bis wir bei $(k,1)$ angekommen sind. In allen Fällen landen wir bei $(k,1)$ und geben k als korrekten prominenten Knoten zurück. Für das Suchen des k haben wir maximal $2n$ viele Inkrementierungs-, bzw Dekrementierungsoperationen. Zum überprüfen ob k auch wirklich ein prominenter Knoten ist, müssen wir sowohl einmal die k Zeile, als auch die k Reihe mit jeweils n vielen Vergleichen überprüfen. Insgesamt haben wir damit also $4n$ viele Iterationen, damit also auch eine Laufzeit von $O(n)$.

Aufgabe 4 (Optimaler Suchbaum): 7 + 2 + 1 = 10 Punkte

Gegeben sind folgende Knoten mit dazugehörigen Zugriffswahrscheinlichkeiten:

Knoten	l_0	N_1	l_1	N_2	l_2	N_3	l_3	N_4	l_4	N_5	l_5
Wert	$(-\infty,1)$	1	(1,2)	2	(2,3)	3	(3,4)	4	(4,5)	5	$(5,\infty)$
Wahrscheinlichkeiten	0.1	0.01	0.1	0.01	0.1	0.04	0.2	0.04	0.2	0.05	0.15

Konstruieren Sie einen optimalen Suchbaum wie folgt.

a) Füllen Sie untenstehende Tabellen für $W_{i,j}$ und $C_{i,j}$ nach dem Verfahren aus der Vorlesung aus. Geben Sie in $C_{i,j}$ ebenfalls **alle möglichen Wurzeln** des optimalen Suchbaums für $\{i,\dots,j\}$ an.

$W_{i,j}$	0	1	2	3	4	5
1						
2	–					
3	–	–				
4	–	–	–			
5	–	–	–	–		
6	–	–	–	–	–	

$C_{i,j}$ ($R_{i,j}$)	0	1	2	3	4	5
1		( )	( )	( )	( )	( )
2	–		( )	( )	( )	( )
3	–	–		( )	( )	( )
4	–	–	–		( )	( )
5	–	–	–	–		( )
6	–	–	–	–	–	

b) Geben Sie einen optimalen Suchbaum für die Knoten mit den gegebenen Zugriffswahrscheinlichkeiten und der gegebenen Reihenfolge der Knoten graphisch an.

c) Ist der optimale Suchbaum für die Knoten mit den gegebenen Zugriffswahrscheinlichkeiten und der gegebenen Reihenfolge der Knoten eindeutig? Geben Sie dazu eine kurze Begründung an.

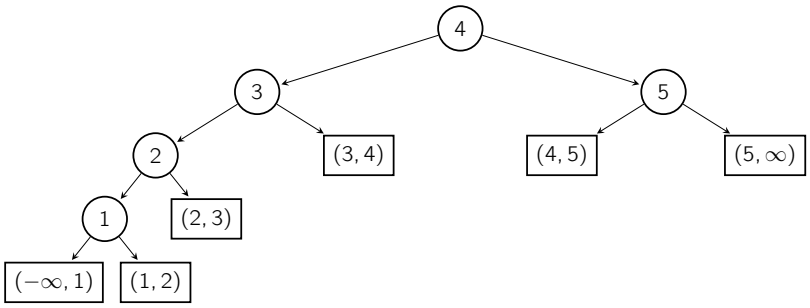
Lösung

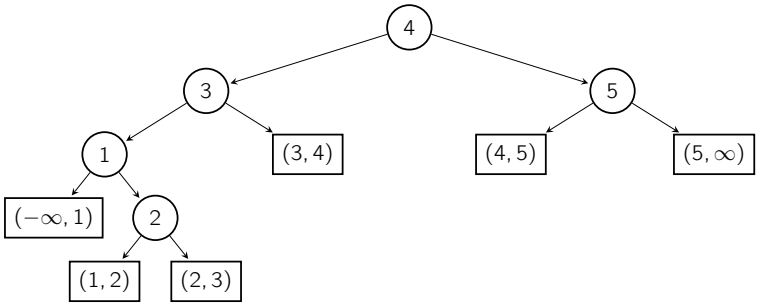
a)

$W_{i,j}$	0	1	2	3	4	5
1	0.10	0.21	0.32	0.56	0.80	1.00
2	–	0.10	0.21	0.45	0.69	0.89
3	–	–	0.10	0.34	0.58	0.78
4	–	–	–	0.20	0.44	0.64
5	–	–	–	–	0.20	0.40
6	–	–	–	–	–	0.15

$C_{i,j}(R_{i,j})$	0	1	2	3	4	5
1	0.10	0.41 (1)	0.83 (1, 2)	1.59 (3)	2.47 (3)	3.34 (4)
2	–	0.10	0.41 (2)	1.06 (3)	1.94 (3)	2.70 (4)
3	–	–	0.10	0.64 (3)	1.42 (4)	2.17 (4)
4	–	–	–	0.20	0.84 (4)	1.59 (4)
5	–	–	–	–	0.20	0.75 (5)
6	–	–	–	–	–	0.15

b) Die folgenden Lösungen sind korrekte optimale Suchbäume.





c) Der optimale Suchbaum ist nicht eindeutig, da wir aus der Tabelle zwei verschiedene optimale Suchbäume konstruieren konnten.

Aufgabe 5 (Union Find): 12 Punkte

Führen Sie die folgenden Operationen beginnend mit einer anfangs leeren *Union-Find-Struktur* aus und geben Sie die entstehende Union-Find-Struktur nach jeder *MakeSet*, *Union* und *Find* Operation an. Nutzen Sie dabei die beiden Laufzeitverbesserungen: Höhenbalencierung und Pfadkompression. Dabei soll die Union-Operation bei **gleicher Höhe der Wurzeln immer die Wurzel des zweiten Parameters** als neue Wurzel wählen. Es ist nicht notwendig die Höhe der Bäume zu notieren.

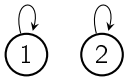
1. MakeSet(1)
2. MakeSet(2)
3. Union(1,2)

4. MakeSet(3)
5. Union(1,3)
6. MakeSet(4)
7. MakeSet(5)
8. Union(4,5)
9. Union(1,4)
10. MakeSet(6)
11. Union(3,6)
12. MakeSet(7)
13. MakeSet(8)
14. Union(7,8)
15. Union(2,7)
16. Find(7)

Lösung

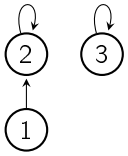
1. MakeSet(1)

2. MakeSet(2)



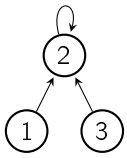
3. Union(1,2)
   - Find(1): unverändert
   - Find(2): unverändert
   - Union(1,2):

4. MakeSet(3)

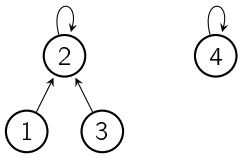


5. Union(1,3)
   - Find(1): unverändert
   - Find(3): unverändert

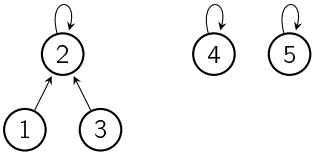
- Union(1,3):



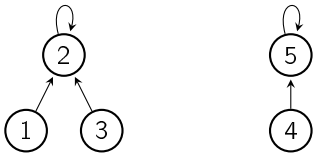
6. MakeSet(4)



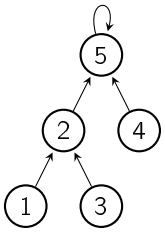
7. MakeSet(5)



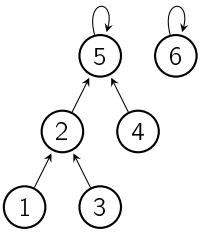
8. Union(4,5)
   - Find(4): unverändert
   - Find(5): unverändert
   - Union(4,5):



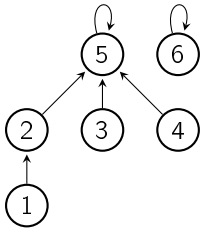
9. Union(1,4)
   - Find(1): unverändert
   - Find(4): unverändert
   - Union(1,4):



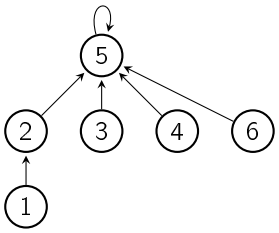
10. MakeSet(6)



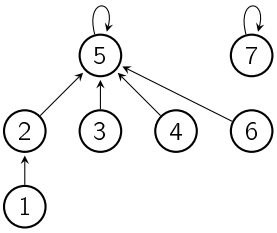
11. Union(3,6)
    - Find(3):



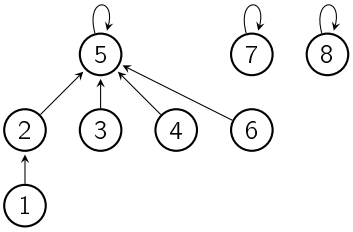
    - Find(6): unverändert
    - Union(3,6):



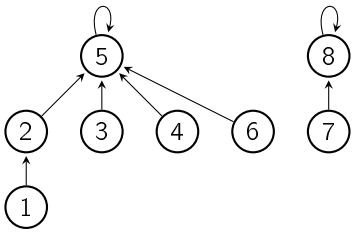
12. MakeSet(7)



13. MakeSet(8)

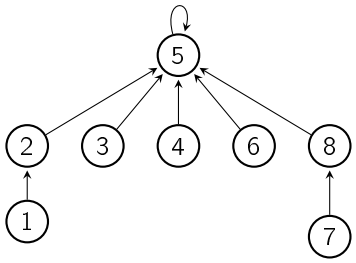


14. Union(7,8)
    - Find(7): unverändert
    - Find(8): unverändert
    - Union(7,8):

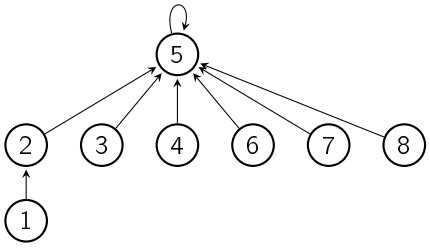


15. Union(2,7)

- Find(2): unverändert
- Find(7): unverändert
- Union(2,7):



16. Find(7)



Aufgabe 6 (Graph Terminology): 2 + 2 + 2 + 2 + (4 * 0.5) = 10 Punkte

a) Sei V eine feste Knotenmenge mit Größe $|V| = n \in \mathbb{N}$. Wie viele Kantenmengen E gibt es, sodass (V, E) ein *gerichteter* Graph ist? Begründen Sie Ihre Antwort kurz.

b) Sei V eine feste Knotenmenge mit Größe $|V| = n \in \mathbb{N}$. Wie viele Kantenmengen E gibt es, sodass (V, E) ein *ungerichteter* Graph ist? Begründen Sie Ihre Antwort kurz.

c) Wie viele einfache Weg der Länge genau $k \in \{0, 1, \ldots, n-1\}$ hat ein vollständiger ungerichteter Graph mit $n \in \mathbb{N}$ Knoten? Begründen Sie Ihre Antwort kurz.

d) Ein einfacher Kreis $v_0 \ldots v_{k-1} v_0$ ist ein Kreis, für den $v_0 \ldots v_{k-1}$ einfach ist. Wie viele einfachen Kreise der Länge mindestens 3 hat ein vollständiger ungerichteter Graph mit $n \in \mathbb{N}$ Knoten? Begründen Sie Ihre Antwort kurz.

e) Sei $G = (V, E)$ ein gerichteter Graph. Wir definieren die Menge $E' = \{(i, j) \mid (j, i) \in E\}$. Betrachten Sie die Graphen $G^T = (V, E')$ und $\hat{G} = (V, \hat{E})$ mit $\hat{E} = E \cup E'$. Beweisen oder widerlegen Sie folgende Aussagen:

i) $\hat{G}$ ist symmetrisch.

ii) Falls $\hat{G}$ stark zusammenhängend ist, dann ist G oder G^T stark zusammenhängend.

iii) Falls G oder G^T stark zusammenhängend ist, dann ist auch $\hat{G}$ stark zusammenhängend.

iv) G ist schwach zusammenhängend genau dann, wenn G^T schwach zusammenhängend ist.

Hinweise:

- Die Länge eines Kreises $v_0 \ldots v_k$ ist k.
- Sie dürfen die Anzahlen auch mit $\sum$ und $\prod$ Termen angeben.

Lösung

Sei $n = |V|$.

a) Es gibt $2^{|M|}$ viele Teilmengen einer endlichen Menge M. Für die Kanten eines gerichteten Graphen $G = (V, E)$ gilt, dass $E \subseteq V \times V = \{(u, v) \mid u, v \in V\}$. Mit $|V \times V| = |V| \cdot |V| = n^2$ folgt, dass es

$$2^{n^2}$$

viele gerichtete Graphen mit n Knoten gibt.

b) Für die Kanten eines ungerichteten Graphen $G = (V, E)$ gilt, dass wenn $(u, v) \in E$ dann auch $(v, u) \in E$. Daher können wir Paare auch als Mengen interpretieren und erhalten dadurch, dass in dieser alternativen Interpretation $E \subseteq \{\{u, v\} \subseteq V \mid u \neq v\} \cup \{\{u\} \mid u \in V\}$ gilt. Es gibt $\binom{m}{k}$ viele k-elementige Teilmengen einer m-elementigen Menge. Daher gibt es $\binom{n}{2}$ viele mögliche Kanten zwischen verschiedenen Knoten und $\binom{n}{1}$ viele mögliche Kanten zwischen gleichen Knoten. Es gibt insgesamt also

$$2^{\binom{n}{2}+\binom{n}{1}} = 2^{\frac{n\cdot(n-1)}{2}+n} = 2^{\frac{n^2}{2}+\frac{n}{2}}$$

viele ungerichtete Graphen mit n Knoten.

Hinweise:

- Man kann bei a) und b) z.B. auch über die Anzahl verschiedener Adjazenzmatrizen argumentieren.

c) Bei einem vollständigen Graphen ist jede Folge von $k + 1$ Knoten ein gültiger Pfad der Länge k. Die Anzahl der Folgen mit $k + 1$ verschiedenen Knoten beträgt

$$\frac{n!}{(n-(k+1))!} = \frac{n!}{(n-k-1)!} = \prod_{i=n-k}^{n} i = \prod_{i=0}^{k}(n-i)$$

Hinweise:

- Beachten Sie, dass für den Fall $k > n$ das obige Produkt immer zu 0 auswertet, da einer der Faktoren 0 ist.

d) In einem vollständigen Graphen gilt:

$$v_0 v_1 \dots v_k \text{ ist ein einfacher Pfad} \iff v_0 v_1 \dots v_k v_0 \text{ ist ein einfacher Kreis}$$

Wir können also jeden einfachen Pfad zu einem einfachen Kreis eindeutig zuordnen.

Maximal kann ein einfacher Kreis Länge n haben.

Aus c) folgt damit: Die Anzahl der Kreise der Länge mindestens 3 beträgt

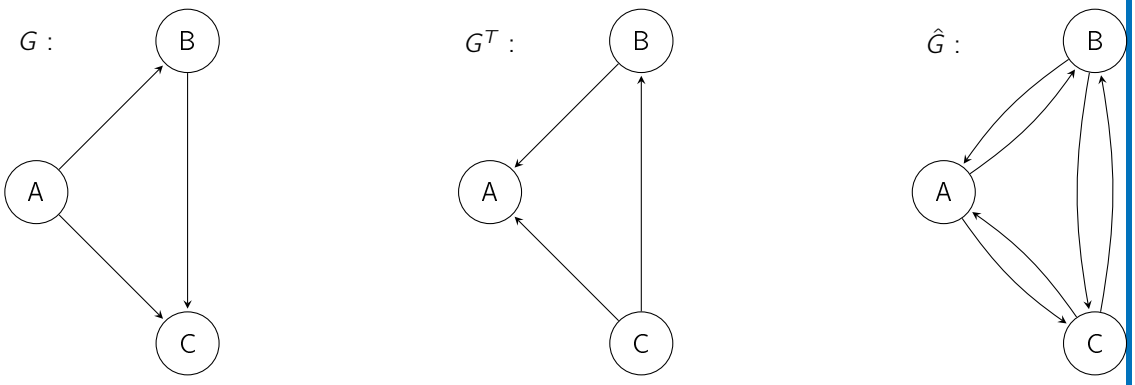
$$\sum_{j=3}^{n} \underbrace{\prod_{i=0}^{j-1}(n-i)}_{\text{Kreise der Länge genau } j}$$

e) i) Die Aussage ist wahr. Falls $(u, v) \in \hat{E}$ dann gibt es zwei Fälle:

- Falls $(u, v) \in E$ ist nach Definition von E' auch $(v, u) \in E'$ und daher $(v, u) \in \hat{E}$.
- Falls $(u, v) \in E'$ ist nach Definition von E' auch $(v, u) \in E$ und daher $(v, u) \in \hat{E}$.

In beiden Fällen folgt also $(v, u) \in \hat{E}$.

ii) Die Aussage ist falsch. Betrachten Sie das folgende Gegenbeispiel. Weder G noch G^T sind stark zusammenhängend, $\hat{G}$ jedoch schon.



iii) Die Aussage ist wahr. Wir nehmen an, dass G stark zusammenhängend ist (der Fall dass G^T stark zusammenhängend ist, ist analog).

Damit $\hat{G}$ stark zusammenhängend ist, muss es von jedem zwei Knoten $u, v \in V$ ein Pfad von u nach v geben. Da G stark zusammenhängend ist, gibt es tatsächlich einen Pfad von u nach v. Da $\hat{E} \subseteq E$ gibt es diesen Pfad dann auch in $\hat{G}$. Damit ist $\hat{G}$ stark zusammenhängend.

iv) Die Aussage ist wahr. " $\Longrightarrow$ ": Sei G schwach zusammenhängend und seien $v, u \in V$ zwei beliebige Knoten. Es gibt eine Folge $v_0 v_1 \dots v_k$ mit $v_0 = v$, $v_k = u$ und für alle $i \in \{0, \dots, k-1\}$ gilt $(v_i, v_{i+1}) \in E$ oder $(v_{i+1}, v_i) \in E$. Nach Definition von E' gilt ebenfalls für alle $i \in \{0, \dots, k-1\}$, dass $(v_{i+1}, v_i) \in E'$ oder $(v_i, v_{i+1}) \in E'$. Es folgt, dass u von v auch in G^T über einen (ungerichteten) Pfad erreichbar ist. G^T ist also schwach zusammenhängend.

" $\Longleftarrow$ ": (analog)

Aufgabe 7 (Zykel finden): 2 + 2 + 2 + 2 = 8 Punkte

Gegeben sei eine einfach verkettete Liste mit n Elementen, deren Länge Sie nicht kennen. Wir betrachten diese Liste im folgenden als gerichteten Graph.

a) Entwerfen Sie einen Algorithmus, mit dem sich testen lässt, ob der Graph einen Zykel enthält.

b) Zeigen Sie die Korrektheit Ihres Algorithmus.

c) Wie ist seine Laufzeit? Begründe Sie Ihre Antwort.

d) Ist dies auch in Zeit $O(n)$ möglich? Begründe Sie Ihre Antwort.

Lösung

Eine Möglichkeit sieht so aus: Mit zwei Zeigern l_1, l_2 durchlaufen wir die Liste. In jedem Schritt wird, falls möglich, l_1 um ein Element weiterverschoben, l_2 um zwei Elemente. Wenn dies nicht möglich ist und das Ende der Liste erreicht ist, liegt offenbar kein Zykel vor. Wenn dieses möglich ist, wird verglichen, ob $l_1 = l_2$ ist, also l_1 und l_2 auf das gleiche Element zeigen. Wenn ja, wurde ein Zykel gefunden.
Zur Analyse: Falls die Liste keinen Zykel enthält, terminiert das Verfahren sobald das Ende der Liste erreicht wurde. Falls die Liste jedoch einen Zykel der Länge k enthält, dann ist nach höchstens n Iterationen die Abbruchbedingung $l_1 = l_2$ erfüllt: Nach höchstens $n - k$ Schritten befinden sich beide Zeiger bereits im Zykel. Im Zykel angekommen reduziert sich die Distanz beider Zeiger nach jeder Iteration um genau eins. Nach höchstens k weiteren Schritten hat sich mindestens einmal die Situation ergeben, dass $l_1 = l_2$, denn irgendwann wird l_2 l_1 überholen. D.h., irgendwann wird die Situation entstehen, dass sie höchstens ein Feld weit auseinanderstehen, also $l_2 = l_1 - 1$. Dann gilt im nächsten Schritt aber $l_2 = l_1$.

Damit ist dann auch die Laufzeit in $O(n)$.
Dieser Algorithmus ist unter dem Namen "Floyd's “tortoise and hare” cycle detection algorithm" bekannt.