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

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 $\{i,\ldots,j\}$ 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.

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)

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.