FernUniversität in Hagen

Lösungsvorschläge zur Hauptklausur „1661 Datenstrukturen I"

8.8.2009

Aufgabe 1

(a)

algebra *playlistverwaltung*

sorts *duration, string, licencetype, pos, track, playlist*

ops

createTrack:	*string* × *duration* × *string* × *string* × *string* × *licence-type*	→ *track*
getTitle:	*track*	→ *string*
getLength:	*track*	→ *duration*
getInterpret:	*track*	→ *string*
getComposer:	*track*	→ *string*
getAuthor:	*track*	→ *string*
getLicence:	*track*	→ *licencetype*
empty:		→ *playlist*
getTrackByPos:	*playlist* × *pos*	→ *track*
getLastPos:	*playlist*	→ *pos*
getTotalLength:	*playlist*	→ *duration*
insertTrackAtPos:	*playlist* × *track* × *pos*	→ *playlist*
deleteTrackAtPos:	*playlist* × *pos*	→ *playlist*
editTrackAtPos:	*playlist* × *pos* × *track*	→ *playlist*
findTrack:	*playlist* × *track* × *pos*	→ *pos*

(b)

sets

$pos = \mathbb{N}_0$

$duration = \mathbb{N}_0$

$licencetype = \{\text{GEMA, free, special, unspecified}\}$,

$track = \{(t, l, i, c, a, lic) \mid t, i, c, a \in string, l \in duration, lic \in licencetype\}$,

$$playlist = \{P \in \boldsymbol{F}((pos \setminus \{0\}) \times track) \mid \forall n \in pos : |\{e \in P \mid e = (n,t)\}| \leq 1, \forall (n,t) \in P : \forall n' \in pos, 0 < n' < n : \exists t' \in track : (n',t') \in P\}$$

Anmerkung: Alternative Lösungen sind natürlich zulässig.

(c)

Anmerkung: In der Aufgabenbeschreibung wurde die Semantik der Operationen für den Fall, dass eine gewählte Position in der Playlist nicht vorkommt, offen gelassen. Für die Angabe der Funktionen legen wir daher willkürlich fest: Von *getTrackbyPos* wird in diesem Falle ein unspezifizierter Track zurückgegeben. Von *insertTrackAtPos* wird der Track ggf. einfach hinter dem letzten in der Playlist enthaltenen Track angehängt. Sowohl *deleteTrack* als auch *moveTrack* liefern in solchen Fällen dagegen die unveränderte Playlist zurück. Andere Festlegungen sind natürlich ebenfalls statthaft.

functions

$createTrack(t, l, i, c, a, lic) = (t, l, i, c, a, lic)$

$getTitle((t, l, i, c, a, lic)) = t$

$getLength((t, l, i, c, a, lic)) = l$

$getInterpret((t, l, i, c, a, lic)) = i$

$getComposer((t, l, i, c, a, lic)) = c$

$getAuthor((t, l, i, c, a, lic)) = a$

$getLicence((t, l, i, c, a, lic)) = lic$

$empty = \{\}$

$$getTrackByPos(L, p) = \begin{cases} t, \text{falls } (p,t) \in L \\ createTrack(\text{"","},0,\text{"","","","},\text{unspecified}), \text{sonst} \end{cases}$$

$$getLastPos(L) = \begin{cases} \max\{p \mid (p,t) \in L\}, \text{ falls L} \neq \{\} \\ 0, \text{ sonst} \end{cases}$$

$$getTotalLength(L) = \sum_{(t,l,i,c,a,lic) \in L} l$$

$$insertTrackAtPos(L, t, p) = \begin{cases} L, \text{falls } p = 0 \\ L \cup (p',t), \text{falls } p > p' \text{ und} \\ \quad p' = \max\{p'' \mid (p'',t'') \in L\} + 1 \\ \{(p',t') \in L \mid p' < p\} \\ \quad \cup \{(p,t)\} \\ \quad \cup \{(p''+1,t'') \mid (p'',t'') \in L, p'' \geq p\}, \text{sonst} \end{cases}$$

$$deleteTrackAtPos(L, p) = \begin{cases} L, \text{falls } (p,t') \notin L \\ \{(p',t') \in L \mid p' < p\} \\ \quad \cup \{(p''-1,t'') \mid (p'',t'') \in L, p'' \geq p\}, \text{sonst} \end{cases}$$

$$editTrackAtPos(L, p, t) = \begin{cases} L, \text{falls } (p,t') \notin L \\ (L \setminus \{(p,t')\}) \cup \{(p,t)\}, \text{sonst} \end{cases}$$

$$findTrack(L, p, t) = \begin{cases} 0, \text{falls } (p,t') \notin L \\ \min(P), \text{falls } P = \{p' \mid p' > p, \\ \quad (p',t') \in L, \\ \quad t = (a,b,c,d,e,f), \\ \quad t' = (a',b',c',d',e',f'), \\ \quad (a = "" \vee a' = "" \vee a = a'), \\ \quad (b = 0 \vee b' = 0 \vee b = b'), \\ \quad (c = "" \vee c' = "" \vee c = c'), \\ \quad (d = "" \vee d' = "" \vee d = d'), \\ \quad (e = "" \vee e' = "" \vee e = e'), \\ \quad (f = \text{unspecified} \\ \quad \vee f' = \text{unspecified} \vee f = f')\} \neq \{\}, \\ 0, \text{sonst} \end{cases}$$

end *playlistverwaltung*

Aufgabe 2

Für die Divisionsmethode gilt: $h(k) = k \bmod m$. Bei einer Tabellengröße von m=10 entspricht das Sortierkriterium der letzten Ziffer des Namenswertes. Für die Grundform des quadratischen Sondierens ergibt sich die Formel: $h_i(k) = (h(k) + i^2) \bmod 10$. Die Namen werden auf die folgenden Behälternummern abgebildet:

Name	k	$h(k)$	$h_1(k)$	$h_2(k)$	$h_3(k)$	$h_4(k)$
Anne	29	9				
Christian	29	9	0			
Frank	25	5				
Heidrun	22	2				
Jianqiu	20	0	1			
Mahmoud	22	2	3			
Markus	32	2	3	6		
Simone	41	1	2	5	0	7
Thomas	43	3	4			

Der Inhalt der Tabelle nach Einfügen aller Namen ist

0	Christian	2	Heidrun	4	Thomas	6	Markus	8	
1	Jianqiu	3	Mahmoud	5	Frank	7	Simone	9	Anne

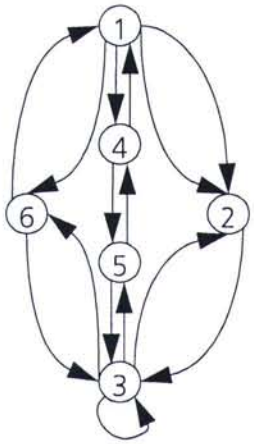
Aufgabe 3

(a) G1 und G2 können nicht als ungerichtete Graphen aufgefasst werden.

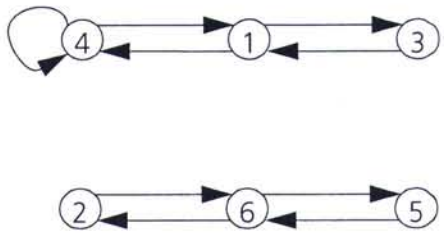
(b) G2 enthält die Zyklen <1, 3 ,4, 1>, <1, 4, 1>, <1, 3, 1>, <4, 4>, <2, 6, 2> und <5, 6, 5>

(c) Die starken Komponenten sind:

Der komplette Graph G1 ist eine starke Komponente:



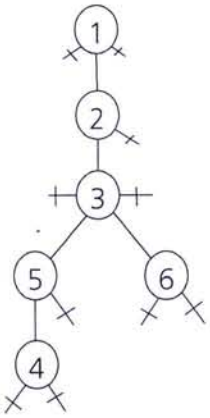
G2 hat 2 starke Komponenten:



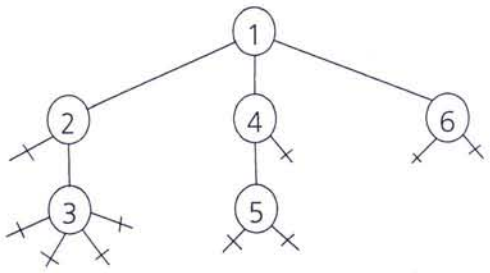
(d)

Für G1 ergeben sich ausgehend von Knoten 1 folgende Spannbäume:

Tiefendurchlauf:

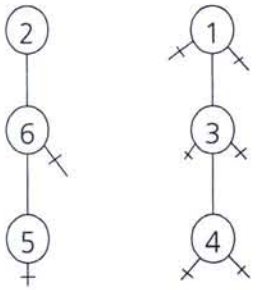


Breitendurchlauf:

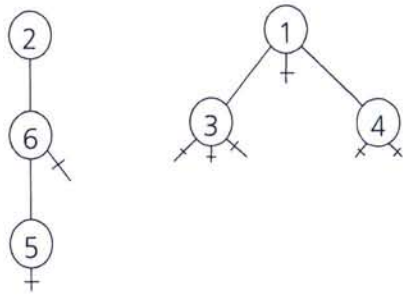


Für G2 ergeben sich ausgehend von Knoten 2 folgende spannende Wälder:

Tiefendurchlauf:



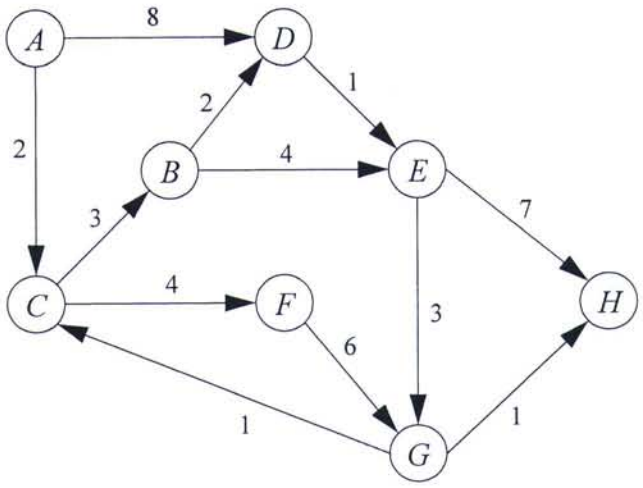
Breitendurchlauf:



Aufgabe 4

(a)

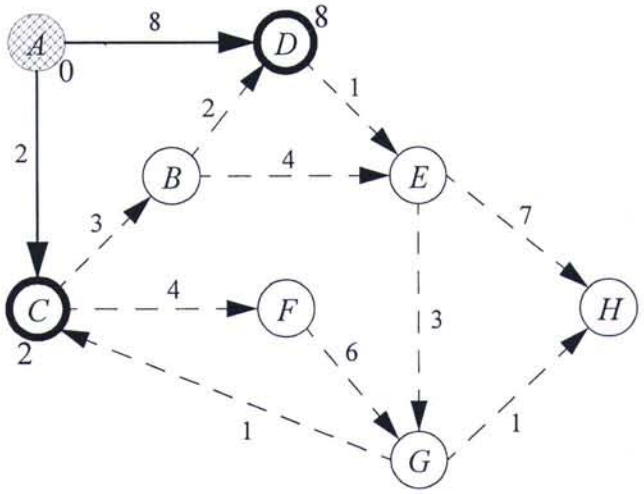
Aus der Kostenmatrix ergibt sich der folgende Graph:



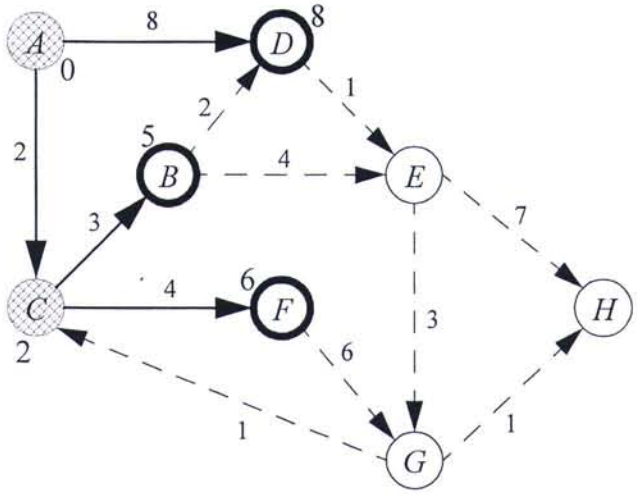
(b)

Im folgenden werden gelb gefärbte Knoten durch einen breiten Rand und grün gefärbte Knoten durch eine Schraffierung markiert. Noch nicht verwendete Kanten werden gestrichelt, gelbe Kanten durch eine durchgehende Linie und rote Kanten durch eine breite, durchgehende Linie gekennzeichnet. Die jeweils aktuellen Abstände zu Knoten *A* werden als zusätzliche Markierungen an den Knoten geschrieben.

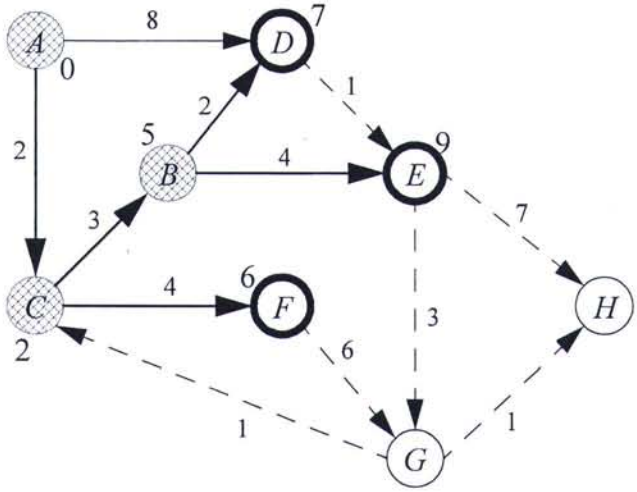
Bei der Initialisierung wird *A* gelb gefärbt und sein Abstand auf 0 gesetzt. Im ersten Schritt wird *A* grün gefärbt und die Abstände zu seinen Nachfolgern gesetzt. Wir erhalten den folgenden Graphen:



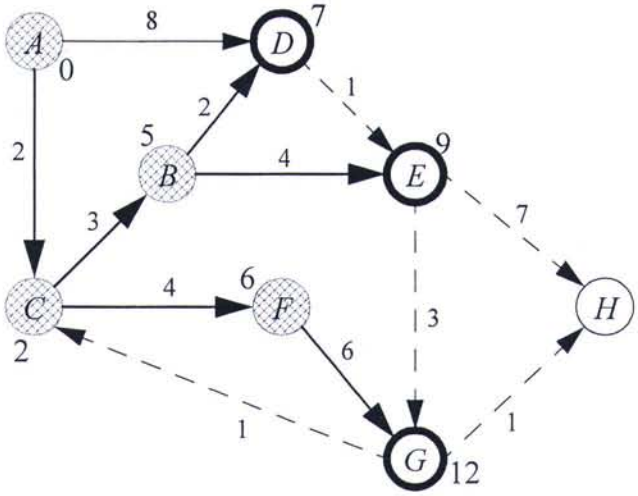
Im nächsten Schritt wird Knoten C gewählt und die Abstände zu seinen Nachfolgern aktualisiert:



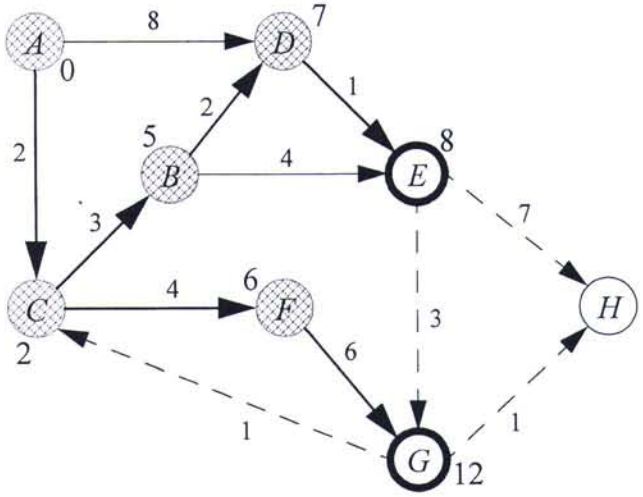
Wir fahren mit der Bearbeitung des Knotens B fort:



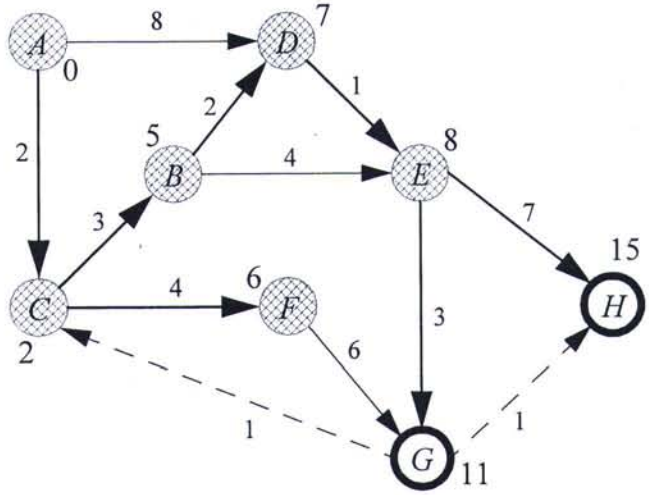
Als nächstes wird Knoten F „grün“ markiert:



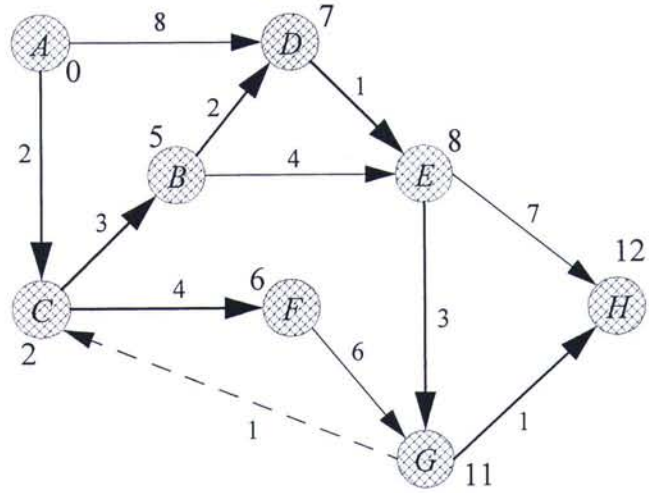
Der nächste zu wählende Knoten ist *D*:



Das Einfügen des Knotens *E* in die Menge der „grün gefärbten" Knoten liefert folgendes Teilergebnis:

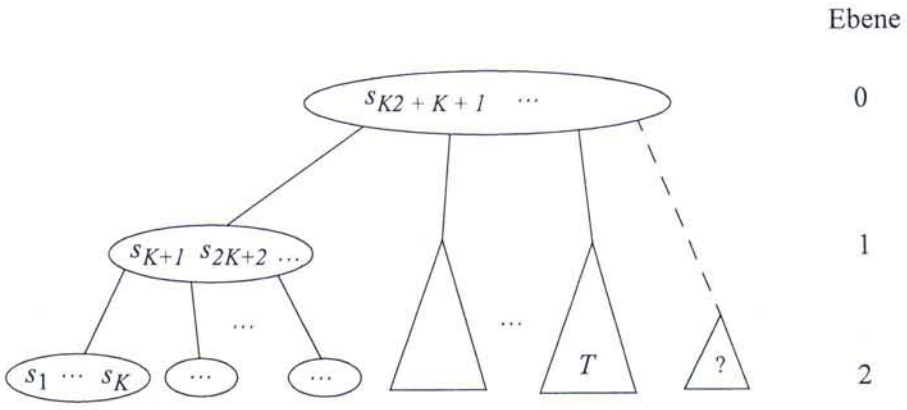


Nun wird der Knoten *G* „grün gefärbt". Dies führt zu einer Aktualisierung der Distanz am Knoten *H*. Dieser wird im letzten Schritt in die Menge der „grünen" Knoten eingefügt, was keine weiteren Änderungen nach sich zieht. Das Ergebnis sieht folgendermaßen aus:



Aufgabe 5

Zunächst berechnet man die Anzahl K der Elemente, die in etwa in einem Knoten gespeichert werden sollen, man erhält $K = 2\,p\,m\,/100$.



Nun durchläuft man die Folge s_i und baut dabei den Baum von links nach rechts und unten nach oben auf. Man beginnt damit, die Elemente einem Blattknoten zuzuordnen. Nachdem K Elemente darin gespeichert sind, hat dieser seinen Füllgrad erreicht und s_{K+1} ist das erste Element für den Vaterknoten der darüber liegenden Ebene. Mit aufsteigendem i werden so Knoten aus immer mehr Ebenen parallel gefüllt. Wenn h die Höhe des resultierenden Baumes ist, müssen aber höchstens h Knoten gleichzeitig verwaltet werden.

Falls N für ein gegebenes p und m nicht groß genug ist, kann es sein, dass sich der am weitesten rechts liegende Teilbaum nicht mehr komplett hochbauen läßt. Die unterfüllten Knoten dieses angefangenen Teilbaumes müssen daher noch mittels *balance* oder *merge* behandelt werden, um die B-Baum Struktur zu erhalten.