Karlsruher Institut für Technologie

Institut für Visualisierung und Datenanalyse
Lehrstuhl für Computergrafik
Prof. Dr.-Ing. Carsten Dachsbacher

Nachklausur Algorithmen I
WS 2021/2022

21. Februar 2022

Name:	
Matrikelnummer:	

Beachten Sie:

- Schreiben Sie Ihren vollständigen **Namen** und **Matrikelnummer** in Druckschrift in das Feld auf dem Deckblatt!
- Schreiben Sie Ihre Matrikelnummer oben auf jedes bearbeitete Aufgabenblatt.
- Schreiben Sie Ihre Lösungen auf die Aufgabenblätter. Bei Bedarf können Sie weiteres Papier anfordern.
- Halten Sie sich bei Pseudocode-Aufgaben an die Richtlinien auf der nächsten Seite.
- Wir akzeptieren auch englische Antworten.
- Sie dürfen **ein handbeschriebenes Din A4-Blatt** mitbringen, sonst sind **keine weiteren Hilfsmittel** zugelassen.
- Sie haben **120 Minuten** Bearbeitungszeit.
- Die Klausur umfasst 25 Seiten (13 Blätter) mit 10 Aufgaben.

Aufgabe	1	2	3	4	5	6	7	8	9	10	**Gesamt**
Erreichte Punkte											
Erreichbare Punkte	20	30	11	23	32	36	33	8	8	39	**240**

Note

Pseudocode-Richtlinien

Verwenden Sie in der Klausur folgende Konventionen für alle Aufgaben, bei denen Sie Pseudocode verwenden! In den Fällen, die hiervon nicht abgedeckt sind, halten Sie sich an die Notation aus der Vorlesung / Übung.

`for`-Schleifen: Indizes iterieren

Verwenden Sie eine der folgenden Schreibweisen für `for`-Schleifenköpfe! In allen Beispielen nimmt `i` alle ganzzahligen Werte von 0 bis inklusive 99 an, aber nicht 100:

```
for i ← 0; i < 100; i++
for i in [0, 99]                for i ∈ [0, 99]
for i in [0,...,99]             for i ∈ [0,...,99]
for i from 0 to 99              for i ← 0 to 99
```

`foreach`-Schleifen: Datenstrukturen iterieren

Über Elemente einer Datenstruktur können Sie in undefinierter Reihenfolge wie folgt iterieren:

```
foreach e in E
foreach (u,v) in E
foreach e = (u,v) in E
```

Array-Indizierung

Arrays und Felder werden mit 0 indiziert. Das heißt, für ein Array `a: [int; 5]` mit 5 Einträgen wird mit `a[0]` auf das erste Element von `a` zugegriffen und mit `a[4]` auf das Letzte.

2D-Arrays

Ein $M \times N$-Array `a` (z.B. zur Repräsentation einer $M \times N$-Matrix $A = (a_{ij})$) mit M Zeilen und N Spalten und Einträgen vom Typ `typ` wird folgendermaßen deklariert:

```
a: [[typ; N]; M]
```

Das heißt, jede Zeile ist ein Feld `[typ; N]`.
Die Zeilen werden nacheinander in `a` abgelegt.

Auf den Eintrag in der i-ten Zeile und j-ten Spalte (bzw. den Matrixeintrag a_{ij}) wird zugegriffen mit

```
a[i][j]
```

Matrikelnummer: _____

Aufgabe 1: Laufzeit (20 Punkte)

a) Beweisen oder widerlegen Sie folgende Aussagen!

i) $2^{n+1} \in O(2^n)$ **(4 Punkte)**

ii) $10n\log(n) \in \mathcal{O}(n^2)$ **(4 Punkte)**

iii) $n^{1000} \in \Omega(n^{\log(n)})$ **(6 Punkte)**

b) Geben Sie für die gegebenen Rekurrenzgleichungen möglichst enge asymptotische Schranken an und begründen Sie diese mit dem Master-Theorem! **(6 Punkte)**

i)

$$T_A(n) = \begin{cases} 1 & \text{falls } n = 1, \\ 3T_A(\lfloor \frac{n}{4} \rfloor) + 2n & \text{sonst.} \end{cases}$$

ii)

$$T_B(n) = \begin{cases} 1 & \text{falls } n = 1, \\ 2T_B(\lfloor \frac{n}{2} \rfloor) + 4n & \text{sonst.} \end{cases}$$

Matrikelnummer: _____

Aufgabe 2: Listen und Arrays (30 Punkte)

a) Nennen Sie je einen Vorteil und einen Nachteil von Arrays (Feldern) gegenüber verketteten Listen! **(4 Punkte)**

Vorteil von Arrays:

Nachteil von Arrays:

b) Warum kann auch mit einem Endzeiger nicht effizient *vom Ende* einer einfach verketteten Liste gelöscht werden? Warum können Sie das Problem auch nicht lösen, indem Sie die Datenstruktur um einen einzelnen weiteren Zeiger ergänzen? **(6 Punkte)**

c) In dieser Aufgabe sei ein unbeschränktes Array `CompactUArray` so implementiert, dass `popBack` das Array bei halbem Füllstand auf die halbe Länge kürzt. Zeigen Sie mithilfe eines Gegenbeispiels, dass für `pushBack` und `popBack` keine amortisierte konstante Laufzeit garantiert werden kann! **(8 Punkte)**

d) In dieser Aufgabe werden Teilmengen von $\mathbb{N}_0$ als sortierte, einfach verkettete Listen mit Dummy-Header (`v = -1`) repräsentiert. Die Funktion `intersect` soll die Schnittmenge von `L1` und `L2` in Laufzeit $\mathcal{O}(|\texttt{L1}| + |\texttt{L2}|)$ berechnen und als neue einfach verkettete Liste zurückgeben. Implementieren Sie die Funktion in Pseudocode! **(12 Punkte)**

```
Struct SItem {
    next : SItem // Nachfolge-Element
    v : int      // Wert dieses Elements
}

Function intersect(L1 : SItem, L2 : SItem) : SItem
{
   L : SItem
   L.next ← L
   L.v ← -1









   return L
}
```

Matrikelnummer: _____

Aufgabe 3: Hashing (11 Punkte)

a) Wir betrachten in dieser Aufgabe eine Hashtabelle mit 8 Stellen und der Hashfunktion $h(x) = x \mod 8$. Zur Kollisionsauflösung wird lineare Suche (lineares Sondieren) angewendet. Der Ausgangszustand der Hashtabelle ist gegeben als:

Hashtabelle

Index	0	1	2	3	4	5	6	7
Wert	0	1	9		12		6	

Führen Sie nun nacheinander die folgenden Operationen durch und geben Sie den Zustand der Hashtabelle nach Ausführen jeder Operation an! **(7 Punkte)**

Hashtabelle

Index	0	1	2	3	4	5	6	7
`insert(2)`								
`insert(16)`								
`insert(5)`								
`remove(6)`								
`remove(1)`								

Weiterer Platz, falls Sie Ihre Antwort korrigieren wollen. Falls Sie diesen nutzen, kennzeichnen Sie eindeutig, welche Lösung gewertet werden soll!

Hashtabelle

Index	0	1	2	3	4	5	6	7
`insert(2)`								
`insert(16)`								
`insert(5)`								
`remove(6)`								
`remove(1)`								

☐ b) Geben Sie jeweils einen Vorteil und einen Nachteil von Hashing mit linearer Suche gegenüber Hashing mit verketteten Listen an! **(4 Punkte)**

Vorteil von linearer Suche:

Nachteil von linearer Suche:

Matrikelnummer: _____

Aufgabe 4: Graphen (23 Punkte)

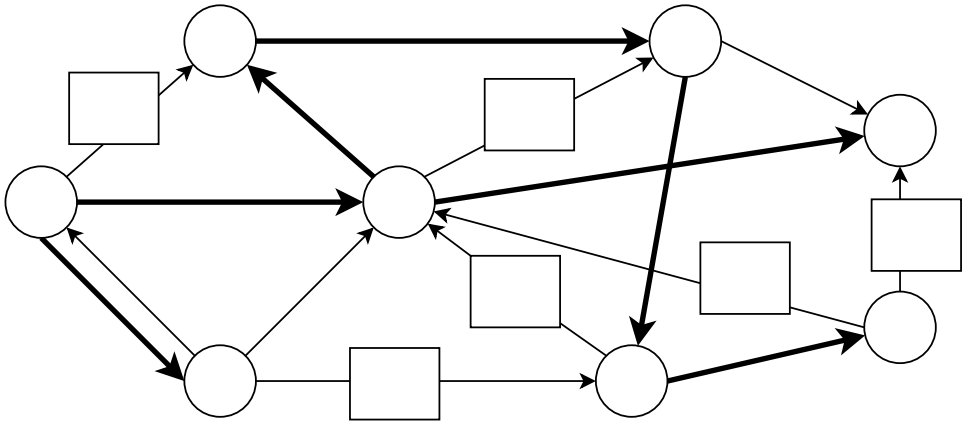
a) Gegeben sei ein Graph $G = (V, E)$ mit n Knoten und m Kanten. Es soll festgestellt werden, ob eine Kante im Graphen enthalten ist. Geben Sie hierzu möglichst enge obere Schranken in $\mathcal{O}$-Notation für die Worst-Case-Laufzeiten an! **(5 Punkte)**

Adjazenzmatrix	Adjazenzliste	Adjazenzfeld

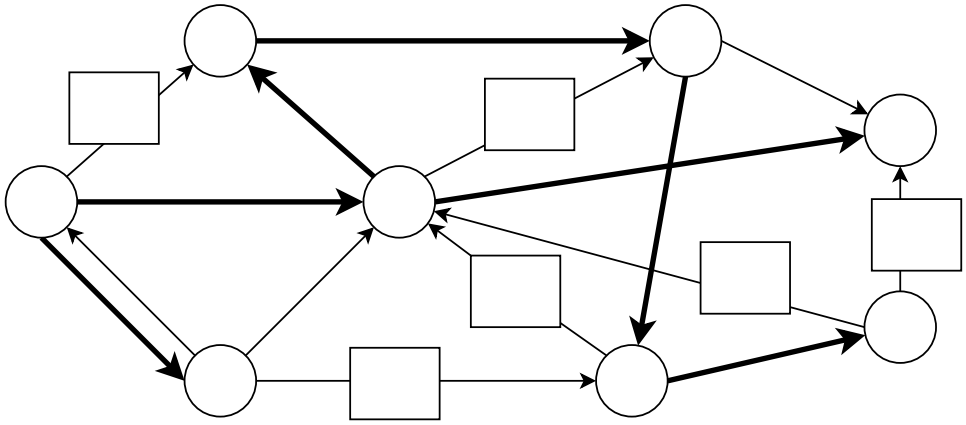
b) Auf dem folgenden Graph wurde eine Tiefensuche ausgeführt. Die Kanten des entstandenen Suchbaums sind breiter dargestellt. Markieren Sie in den Kästen:

- die Vorwärtskanten mit v!
- die Rückwärtskanten mit r!
- die Querkanten mit q!

(6 Punkte)



Ersatzgraph. Markieren Sie, welcher Graph bewertet werden soll!



c) Gegeben sei ein gerichteter gewichteter Graph $G = (V, E)$ mit $V = \{0, \ldots, N-1\}$ und Kantengewichten in $(0, \infty)$. Die Funktion `findpath2` soll ein Tupel (x, w) zurückgeben, sodass x der Zwischenknoten eines kürzesten Pfades (a, x, b) von a nach b mit genau zwei Kanten ist, und w das Gewicht dieses Pfades. Falls kein solcher Pfad existiert, soll das Tupel $(-1, \infty)$ zurückgegeben werden.

Der Graph liegt als Adjazenzmatrix `M` vor. Dabei stellen die Einträge direkt die Kantengewichte dar. Ein Eintrag von 0 bedeutet, dass die entsprechende Kante nicht im Graphen enthalten ist.

☐ Implementieren Sie `findpath2` in Pseudocode mit Worst-Case-Laufzeit $\mathcal{O}(N)$ und zusätzlichem Speicherbedarf $\mathcal{O}(1)$! **(12 Punkte)**

```
Function findpath2(M : [[float; N];N], a : int, b : int) :
(int, float)
{
      x : int
      w : float

      return (x, w)
}
```

Matrikelnummer: _____

Aufgabe 5: Sortieren (32 Punkte)

a) Songs einer Musikdatenbank `D` sollen mit der Funktion `sortTracks` sortiert werden. Die Reihenfolge zweier Songs richtet sich dabei nach ihrem Erscheinungsjahr (Schlüssel `year`), bei gleichem Erscheinungsjahr nach ihrem Albentitel (Schlüssel `title`) und bei zusätzlich gleichem Albentitel nach ihrer Titel-Nummer (Schlüssel `trackNo`).

- Geben Sie die Funktion `sortTracks` in Pseudocode an!
- Verwenden Sie hierzu die Funktion `sort(D, k)`, die `D` nach Schlüssel `k` sortiert!
- Welche Eigenschaft muss der in `sort` implementierte Sortieralgorithmus erfüllen?

(8 Punkte)

```
Function sort(D : [Data; N], k: Key) : [Data; N] { ... }

Function sortTracks(D : [Data; N]) : [Data; N] {




}
```

Eigenschaft von `sort`:

b) In dieser Aufgabe sollen Sie indirektes Sortieren eines Arrays `A` der Länge `N` vom Typ `T` implementieren. Dabei wird zuerst eine Permutation in Form eines Arrays von Indizes `I` berechnet, die anschließend auf A angewandt wird.

Hinweis: Sie können die Funktion `swap(a, b)` verwenden, um a und b zu tauschen.

i) Implementieren Sie zunächst den Permutationsschritt auf Basis von Insertionsort in Pseudocode! Permutieren Sie dafür `I`, sodass $i < j \Rightarrow \texttt{A}[\texttt{I}[i]] \leq \texttt{A}[\texttt{I}[j]]$ für alle $i, j \in [0, N)$ gilt. **(10 Punkte)**

```
Function sortIndirect(A : [T; N]) : [int; N] {
   I ← [_; N] // Index-Array
   // Initialisiere Index-Array mit Identität
   for i ← 0; i < N; i++ do
   | I[i] ← i
   end

   return I
}
```

ii) Implementieren Sie nun die Anwendung der Permutation `I` auf `A` in Pseudocode! Das bedeutet $\texttt{A}_{\text{out}}[i] = \texttt{A}_{\text{in}}[\texttt{I}[i]]$ für alle $i \in [0, N)$, wobei $\texttt{A}_{\text{in}}$ und $\texttt{A}_{\text{out}}$ der Zustand von `A` vor bzw. nach Ausführung des Algorithmus ist. Der Algorithmus muss dabei in-place sein. **(14 Punkte)**

```
Function permute(A : [T; N], I : [int; N]) {

}
```

Aufgabe 6: Prioritätslisten (36 Punkte)

a) Gegeben ist das folgende Array. Gilt dafür die Heap-Invariante für binäre Min-Heaps? Begründen Sie Ihre Aussage! **(4 Punkte)**

1	2	3	4	5	6	7	8	9	10	11	12	13	14
4	10	24	15	30	14	20	13	35	40	70	72	25	65

b) Erfüllt Heap-Sort aus der Vorlesung die in-place Eigenschaft? Begründen Sie Ihre Antwort! **(4 Punkte)**

Matrikelnummer: _____

c) Im Folgenden betrachten wir 3-äre Heaps. Sie unterscheiden sich von den impliziten binären Heaps nur dadurch, dass jeder Knoten des Heaps maximal drei Kinder hat.

i) Gegeben sei ein 3-ärer Min-Heap. Führen Sie die angegebenen Operationen nacheinander auf dem Heap aus und tragen Sie ihre Lösung in die gegebenen Felder ein! **(8 Punkte)**

Index	1	2	3	4	5	6	7	8	9	10	11	12
Ausgangsheap	1	4	8	16	5	18	21	10	20	25	19	
`insert(11)`												
`deleteMin()`												

Weiterer Platz, falls Sie Ihre Antwort korrigieren wollen. Falls Sie diesen nutzen, kennzeichnen Sie eindeutig, welche Lösung gewertet werden soll!

Index	1	2	3	4	5	6	7	8	9	10	11	12
Ausgangsheap	1	4	8	16	5	18	21	10	20	25	19	
`insert(11)`												
`deleteMin()`												

ii) Definieren Sie die Funktionen `parent(i)` und `child(i,j)` für einen 3-ären Heap! Die Funktion `parent(i)` gibt dabei den Elternknoten des Knotens `i` an, und `child(i, j)` ($1 \leq$ `j` ≤ 3) gibt das `j`-te Kind des Knotens `i` an. **(8 Punkte)**

Hinweis: Das Array des Heaps ist wie oben 1-indiziert.

`parent(i)` =

`child(i, j)` =

iii) Implementieren Sie die Funktion `siftDown` für einen 3-ären Heap in Pseudocode, sodass die angegebene `deleteMin` Operation korrekt ist! Der Heap liegt dabei im **1-indizierten** Array `H` der Größe `n`. Sie dürfen neben den Funktionen `child(i,j)` und `parent(i)` noch die Funktion `swap(a, b)` nutzen, welche `a` und `b` tauscht. **(12 Punkte)**

```
Function deleteMin(H : [Element; n]) : Element {
   result ← H[1]
   H[1] ← H[n]
   n ← n - 1
   if n ≠ 0 then
   | siftDown(H, 1)
   end
   return result
}

Function siftDown(H : [Element; n], i : ℕ)
{

}
```

Matrikelnummer: _____

Aufgabe 7: Kürzeste Wege (33 Punkte)

a) Wir suchen kürzeste Wege von einem Startknoten zu den anderen n Knoten eines gerichteten Graphen G mit m Kanten. Geben Sie für die folgenden Graphen-Typen einen Algorithmus aus der Vorlesung an, der eine korrekte Lösung asymptotisch am schnellsten findet! Geben Sie außerdem die asymptotischen Worst-Case Laufzeiten an! **(6 Punkte)**

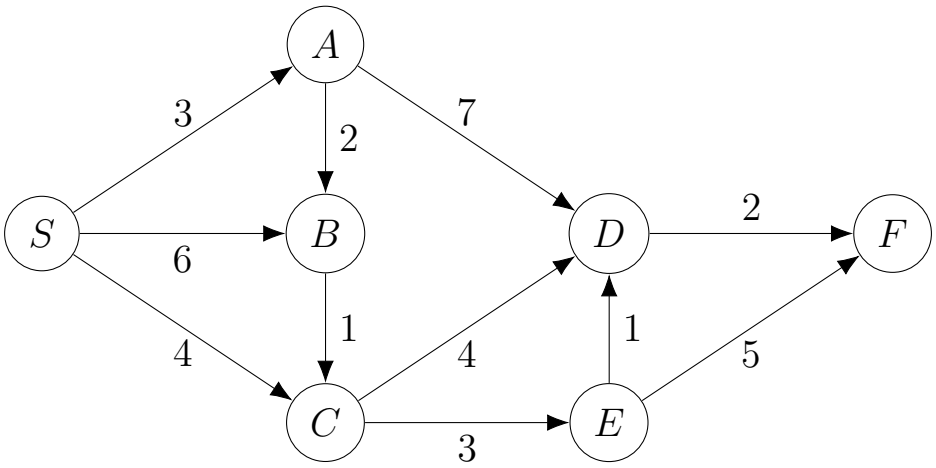
	Algorithmus	Laufzeit
Keine Kantengewichte		
Nicht-Negative Gewichte		
Mit negativen Kreisen		

b) Wir betrachten das Ergebnis des Dijkstra-Algorithmus nach dessen Ausführung auf einem beliebigen ungerichteten, zusammenhängendem Graphen G mit positiven Kantengewichten von einem beliebigen Startknoten S aus.

i) Warum ist die Menge der Kanten im Ergebnis des Dijkstra-Algorithmus stets ein Spannbaum des Graphen G? **(6 Punkte)**

ii) Zeigen oder widerlegen Sie: Die Menge der Kanten im Ergebnis des Dijkstra-Algorithmus ist stets ein *minimaler* Spannbaum des Graphen G! **(6 Punkte)**

c) Führen Sie den Dijkstra-Algorithmus auf dem unten gegebenen gerichteten Graphen aus, um alle kürzesten Wege vom Startknoten S aus zu ermitteln! Geben Sie nach jedem Durchlauf der äußeren Schleife den Vorgänger V und die bisher geringste Distanz D eines jeden Knotens an! Umkreisen Sie außerdem in jedem Durchlauf den Knoten in der Spalte, der im nächsten Schritt gescannt wird, wie beispielhaft in der ersten Spalte gezeigt! **(15 Punkte)**

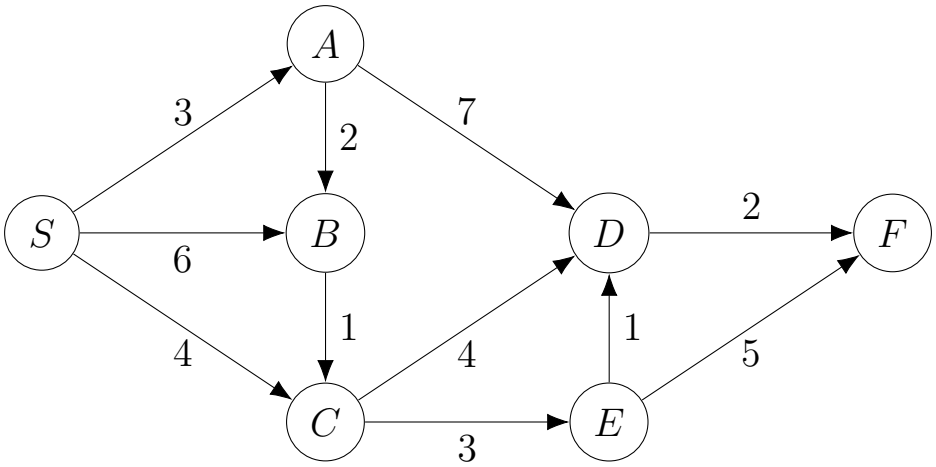


	0		1		2		3		4		5		6	
	V	D	V	D	V	D	V	D	V	D	V	D	V	D
S	-	0												
A	-	∞												
B	-	∞												
C	-	∞												
D	-	∞												
E	-	∞												
F	-	∞												

Ersatztabelle auf der folgenden Seite. Falls Sie Eintragungen in beiden Tabellen vornehmen, kennzeichnen Sie *deutlich*, welche Tabelle gewertet werden soll!

Matrikelnummer: _____

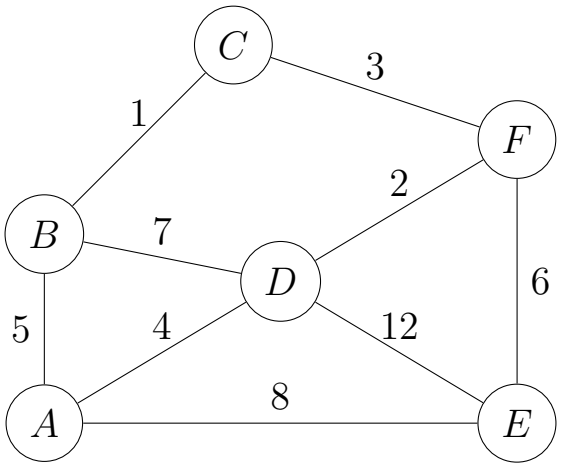
Ersatztabelle. Markieren Sie, welche Tabelle bewertet werden soll!



	0		1		2		3		4		5		6	
	V	*D*	*V*	*D*	*V*	*D*	*V*	*D*	*V*	*D*	*V*	*D*	*V*	*D*
S	-	0												
A	-	∞												
B	-	∞												
C	-	∞												
D	-	∞												
E	-	∞												
F	-	∞												

Aufgabe 8: Minimale Spannbäume (MST) (8 Punkte)

a) Bestimmen Sie den minimalen Spannbaum (MST) des folgenden Graphen mithilfe des Kruskal-Algorithmus! Geben Sie in der Tabelle pro Schritt die betrachtete Kante an, sowie ob diese zum MST hinzugefügt wurde! **(8 Punkte)**



Schritt	1	2	3	4	5	6	7	8	9
Kante									
Im MST ($\checkmark/X$)									

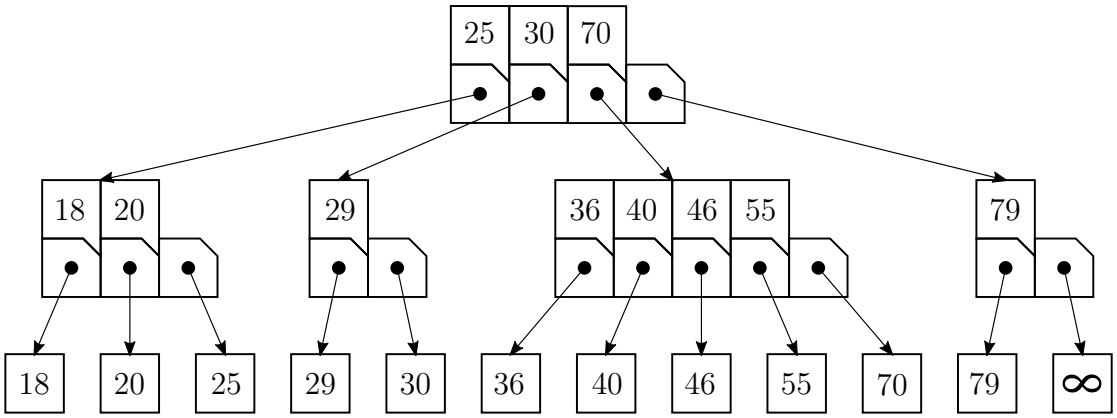
Weiterer Platz, falls Sie Ihre Antwort korrigieren wollen. Falls Sie diesen nutzen, kennzeichnen Sie eindeutig, welche Lösung gewertet werden soll!

Schritt	1	2	3	4	5	6	7	8	9
Kante									
Im MST ($\checkmark/X$)									

Matrikelnummer: _____

Aufgabe 9: Suchbäume (8 Punkte)

a) Führen Sie auf dem folgenden $(2, 5)$-Baum die Operation **remove(79)** aus und geben Sie den resultierenden Baum an! **(8 Punkte)**



Aufgabe 10: Optimierung (39 Punkte)

a) Welches Prinzip muss gelten, damit dynamische Programmierung als Lösungsansatz für ein Problem anwendbar ist? Wann gilt dieses Prinzip? **(5 Punkte)**

Matrikelnummer: _____

b) Sie haben sich ein neues Regal gekauft und möchten jetzt Ihre Bücher dort ablegen. Jedoch passen nicht alle Ihre Bücher in das Regal. Mit einem Algorithmus möchten Sie eine Auswahl treffen. Dazu gelte:

- Ihre Bücher seien durch ein Array `B` der Länge `n` gegeben.
- Ein Buch hat jeweils eine Breite `b` und einen Wert `w`.
- Ein Regal hat eine Breite `l` und Sie speichern die im Regal abgelegten Bücher aus `B` in einem Array `S`.
- Die gesamte Breite aller Bücher in `S` darf nicht größer als die gegebene Regalbreite `l` sein.
- Ein leerer Platz an Stelle `i` ist repräsentiert durch `S[i].b = S[i].w = 0`.

```
Struct Buch {
    b : int // Breite
    w : int // Wert
}
```

```
Struct Regal {
    l : int // Regalbreite
    S : [Buch; n]
}
```

i) Geben Sie einen Algorithmus in Pseudocode an, der ein leeres Regal mit der **maximalen Anzahl** an Büchern aus `B` in Laufzeit $\mathcal{O}(n \log n)$ füllt! **(8 Punkte)**

Hinweis: Sie können die Funktion `sortAufsteigend(A)` bzw. `sortAbsteigend(A)` verwenden, um ein Array von Büchern `A` nach ihrer Breite in Laufzeit $\mathcal{O}(n \log n)$ zu sortieren.

```
Function maxNumber( R : Regal, B: [Buch; n], n: int) {

}
```

ii) Nun ist Ihnen wichtiger, den gesamten Wert der Bücher im Regal zu maximieren. $V(\mathtt{i}, \mathtt{l})$ ist der maximale Gesamtwert, der durch Füllen des Regals mit Büchern aus $\mathtt{B}[0, \ldots, \mathtt{i}-1]$ bis zu einer Gesamtbreite $\mathtt{l}$ erreicht werden kann. Geben Sie eine rekursive Definition für $V(\mathtt{i}, \mathtt{l})$ an! **(10 Punkte)**

$V(\mathtt{i}, \mathtt{l}) =$

$V(0, \mathtt{l}) =$

Matrikelnummer: _____

iii) Sie sollen nun den maximalen Gesamtwert einer Auswahl von Büchern berechnen, die in das Regal passt. Geben Sie einen Algorithmus in Pseudocode an, der den **maximalen Gesamtwert** von Büchern aus `B` für das Regal `R` in Laufzeit $\mathcal{O}(n \cdot l)$ berechnet! **(16 Punkte)**

```
Function maxValue( R : Regal, B: [Buch; n], n: int) : int {

}
```