Vorname	Name	Matr.-Nr.

Aufgabe 1 Komplexität (3+4+3=10 Punkte)

a) Zeigen oder widerlegen Sie:

$$\frac{n}{\ln n} \in O(\sqrt{n})$$

Lösung:

Es gilt :

$$\lim_{n\to+\infty} \frac{\frac{n}{\ln n}}{\sqrt{n}} = \lim_{n\to+\infty} \frac{\sqrt{n}}{\ln n} \overset{l'hopital}{=} \lim_{n\to+\infty} \frac{\frac{1}{2}\cdot\frac{1}{\sqrt{n}}}{\frac{1}{n}} = \lim_{n\to+\infty} \frac{1}{2}\cdot\frac{n}{\sqrt{n}} = \lim_{n\to+\infty} \frac{\sqrt{n}}{2} = \infty$$

Also ist die Aussage falsch.

b) Zeigen Sie, dass $(n+a)^b \in \Theta(n^b)$, für $a, b \in \mathbb{N}$ gilt.

Lösung:

Es gilt:

$$\begin{aligned}
\lim_{n\to+\infty} \frac{(n+a)^b}{n^b} &= \lim_{n\to+\infty} (\frac{n+a}{n})^b \\
&= \lim_{n\to+\infty} (1+\frac{a}{n})^b \\
&= \overbrace{\lim_{n\to+\infty} (1+\frac{a}{n}) \cdot \lim_{n\to+\infty} (1+\frac{a}{n}) \cdot \cdots\cdots \lim_{n\to+\infty} (1+\frac{a}{n})}^{b-mal} \\
&= \overbrace{1\cdot\cdots\cdot 1}^{b-mal} \\
&= 1
\end{aligned}$$

Vorname	Name	Matr.-Nr.

c) Lösen Sie die Rekursionsgleichung $T(n) = 4T(\frac{n}{2}) + n^3 - n - 1$ mit Hilfe des Mastertheorems.

Lösung:

$E = \log_2 4 = 2$

Da $n^3 - n - 1 \in \Omega(n^{2+\varepsilon})$ gilt, handelt es sich vielleicht um den Fall 3 des Mastertheorems. Überprüfung der Zusatzbedingung:

$$\begin{array}{lll} & b \cdot f(\frac{n}{c}) & \leq \quad d \cdot f(n) \\ \Leftrightarrow & 4 \cdot ((\frac{n}{2})^3 - (\frac{n}{2}) - 1) & \leq \quad d \cdot (n^3 - n - 1) \\ \Leftrightarrow & 4 \cdot (\frac{n^3}{8} - \frac{n}{2} - 1) & \leq \quad d \cdot (n^3 - n - 1) \\ \Leftrightarrow & \frac{n^3}{2} - 2n - 4 & \leq \quad d \cdot (n^3 - n - 1) \end{array}$$

Für $d = \frac{1}{2}$ (beispielsweise) gilt die obige Ungleichung. Damit folgt mit dem Master-Theorem

$$T(n) \in \Theta(n^3).$$

Vorname	Name	Matr.-Nr.

Aufgabe 2 Hashing (3+3+4 = 10 Punkte)

a) Geben Sie die Hashtabelle an, die entsteht, wenn die unten gegebenen Schlüsselwerte unter Verwendung der Hashfunktion

$$h_1(k) = k \bmod 13$$

sukzessive in eine zu Beginn leere *Hashtabelle mit Verkettung (geschlossene Adressierung)* der Länge 13 einfügt werden.

Schlüsselwerte: 5, 35, 3, 16, 27, 39, 26

Lösung:

0	1	2	3	4	5	6	7	8	9	10	11	12
39 26	27		3 16		5				35			

Vorname	Name	Matr.-Nr.

b) Geben Sie die Hashtabelle an, die entsteht, wenn die unten gegebenen Schlüsselwerte unter Verwendung der Hashfunktion

$$h_2(k) = (k \bmod 7) + 1$$

mit *quadratischer Sondierung* und $c_1 = c_2 = 2$, sukzessive in eine zu Beginn leere *Hashtabelle mit offener Adressierung* der Länge 13 einfügt werden.

Nutzen Sie für jeden einzufügenden Schlüsselwert eine Zeile der folgenden Tabelle und markieren Sie fehlgeschlagene Sondierungspositonen mit einem $\times$, die erfolgreiche Sondierungsposition mit dem entsprechenden Schlüssel. Eingefügte Schlüsselwerte müssen in den folgenden Zeilen nicht wiederholt werden.

Schlüsselwerte: 15, 32, 21, 8, 45, 14, 17

Lösung:
Es sollte also die Hashfunktion $h(k, i) = (h_2(k) + 2i + 2i^2) \bmod 13$ benutzt werden.

0	1	2	3	4	5	6	7	8	9	10	11	12
		15										
					32							
	21											
		$\times$				8						
				45								
14	$\times$				$\times$							
				$\times$				17				

Vorname	Name	Matr.-Nr.

5

c) Kuckuck-Hashing ist ein Verfahren um Kollisionen in Hashtabellen aufzulösen. Die Implementierung der Kuckuck-Hashing Einfügeoperation, die als Parameter den einzufügenden Schlüsselwert `a` sowie die Hashtabelle `T` erhält, ist unten angegeben. Die Methoden `h1` und `h2` sind zwei verschiedene Hashfunktionen.

```
1  public boolean kuckuckHashing(int a, int[] T) {
2    if (T[h1(a)] == empty) T[h1(a)] = a;
3    else {
4      int tmp = T[h1(a)];
5      T[h1(a)] = a;
6      while(true) {
7        if (T[h2(tmp)] == empty ) {
8          T[h2(tmp)] = tmp;
9          return true;
10       } else {
11         int tmp2 = T[h2(tmp)];
12         T[h2(tmp)] = tmp;
13         if (h2(tmp) == h2(tmp2)) return false;
14         tmp = tmp2;
15       }
16   }
17 }
```

Geben Sie die Hashtabelle an, die entsteht, wenn die unten gegebenen Schlüsselwerte unter Verwendung der Hashfunktionen

$$h_1(k) = k \bmod 13, \qquad h_2(k) = (k \bmod 7) + 1$$

entsprechend *Kuckuck-Hashing* sukzessive in eine zu Beginn leere Hashtabelle der Länge 13 einfügt werden.

Schlüsselwerte: 5, 35, 3, 16, 27, 39, 26

Lösung:

0	1	2	3	4	5	6	7	8	9	10	11	12
					~~5~~							
									35			
			~~3~~									
			16	3								
	27											
~~39~~												
26					39	5						

Vorname	Name	Matr.-Nr.

Aufgabe 3 Interpolationssuche (7+3=10 Punkte)

Im Folgenden ist der Quellcode der *Interpolationssuche* gegeben. Die Interpolationssuche ist ein Suchalgorithmus, der analog zur *binären Suche*, nur auf sortierte Listen angewandt werden kann.

```
1  public int interpolierteSuche(int key, int daten[]) {
2    int links = 0;                  // linke Teilfeldbegrenzung
3    int rechts = daten.length - 1; // rechte Teilfeldbegrenzung
4    int diff;
5    int pos;                        // aktuelle Teilungsposition
6
7    // solange der key im Bereich liegt (andernfalls ist das gesuchte
8    // Element nicht vorhanden)
9    while( key >= daten[links] && key <= daten[rechts] ){
10     diff = daten[rechts] - daten[links];
11
12     // Berechnung der neuen interpolierten Teilungsposition
13     pos = links + (int)((rechts - links) * (key - daten[links])/ diff);
14
15     if( key > daten[pos] )
16       links = pos + 1;
17     else if( key < daten[pos] )
18       rechts = pos - 1;
19     else                                  // Element gefunden
20       return pos;
21   }
22
23   return -1;                              // Element nicht gefunden
24 }
```

a) Gegeben sei das folgende Array von ganzen Zahlen:

$$daten = [2, 2, 3, 11, 15, 19, 24, 42]$$

Führen Sie die Interpolationssuche für den Schlüsselwert 19 durch. Geben Sie für jeden Schleifendurchlauf die Werte der Parameter `links`, `rechts` und `pos` an.

Lösung:

Durchlauf	links	rechts	pos
1	0	7	2
2	3	7	4
3	5	7	5

b) Für welche Eingabe ergibt sich die Worst-Case Laufzeit, für welche die Best-Case Laufzeit? Geben Sie sowohl die Worst-Case als auch die Best-Case Laufzeit an und begründen Sie Ihre Antwort.

Lösung:

Worst-Case Laufzeit: $O(n)$; z.B. bei Eingabe von $[0, 0, 0, 24, 100]$ und Suche nach 24.

Best-Case Laufzeit: $O(1)$, also konstante Laufzeit. Dies geschieht beispielsweise für $[1, 2, 3, 4, 5, 6]$ und jeden möglichen gesuchten Wert.

Vorname	Name	Matr.-Nr.

Aufgabe 4 Algorithmische Geometrie (7+3=10 Punkte)

a) Bestimmen Sie mit Hilfe des Graham-Scan Algorithmus die konvexe Hülle der folgenden Punkte. Die *nötige Sortierung* soll mit Hilfe von *Insertionsort* realisiert werden:

$$(4,3),\ (3,4),\ (3,2),\ (8,2),\ (1,1),\ (5,3)$$

Geben Sie alle Determinanten an die berechnet werden müssen, sowie jeweils die bereits sortierte Liste bzw. den aktuellen Zustand des Stacks.

Lösung:

Der Punkt mit der niedrigsten y-Koordinate ist der Punkt $(1,1)$.

Wir sortieren die restlichen Punkte mit Hilfe von Insertionsort aufsteigend nach Polarwinkel bezüglich Punkt $(1,1)$:

Punkt	Determinanten	sortierte Teilliste
$(4,3)$		$(4,3)$
$(3,4)$	$det\begin{pmatrix} 2 & 3 \\ 3 & 2 \end{pmatrix} = -5$	$(4,3),(3,4)$
$(3,2)$	$det\begin{pmatrix} 2 & 3 \\ 1 & 2 \end{pmatrix} = 1$	$(3,2),(4,3),(3,4)$
$(8,2)$	$det\begin{pmatrix} 7 & 2 \\ 1 & 1 \end{pmatrix} = 5$	$(8,2),(3,2),(4,3),(3,4)$
$(5,3)$	$det\begin{pmatrix} 4 & 7 \\ 2 & 1 \end{pmatrix} = -10,\ det\begin{pmatrix} 4 & 2 \\ 2 & 1 \end{pmatrix} = 0!$	$(8,2),(5,3),(4,3),(3,4)$

Die Determinante für $(5,3)$ $(3,2)$ ist 0, da die beiden Punkte auf einer Linie liegen. Deshalb nehmen wir nur den weiter entfernten Punkt mit in die Liste auf.

Wir nehmen jetzt Punkt für Punkt mit in die Hülle auf und überprüfen, ob diese konvex bleibt.

Hülle/Stack	Determinanten	Bemerkung
$(1,1),(8,2)$		Die ersten zwei Punkte sind auf jeden Fall in der Hülle enthalten.
$(1,1),(8,2),(5,3)$	$det \begin{pmatrix} 7 & -3 \\ 1 & 1 \end{pmatrix} = 10$	ist eine konvexe Ecke
$(1,1),(8,2),(5,3),(4,3)$	$det \begin{pmatrix} -3 & -1 \\ 1 & 0 \end{pmatrix} = 1$	ist eine konvexe Ecke
$(1,1),(8,2),(5,3),(4,3),(3,4)$	$det \begin{pmatrix} -1 & -1 \\ 0 & 1 \end{pmatrix} = -1$	$(4,3)$ ist nicht Teil der konvexen Hülle
$(1,1),(8,2),(5,3),(3,4)$	$det \begin{pmatrix} -3 & -2 \\ 1 & 1 \end{pmatrix} = -1$	$(5,3)$ ist nicht Teil der konvexen Hülle
$(1,1),(8,2),(5,3),(3,4)$	$det \begin{pmatrix} -3 & -2 \\ 1 & 1 \end{pmatrix} = -1$	$(5,3)$ ist nicht Teil der konvexen Hülle
$(1,1),(8,2),(3,4)$	$det \begin{pmatrix} 7 & -5 \\ 1 & 2 \end{pmatrix} = 21$	$(8,2)$ ist Teil der konvexen Hülle

Der letzte Schritt ist nicht nötig, da der zweite Punkt immer ein Teil der Hülle ist.

Es ergibt sich die konvexe Hülle:

$$(1,1),(8,2),(3,4)$$

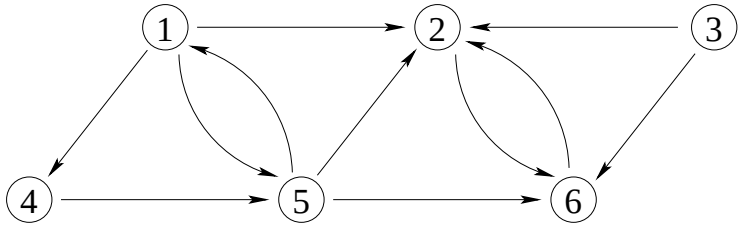
b) Geben Sie die Worst-Case Laufzeitkomplexität des in a) realisierten Verfahrens an. Begründen Sie Ihre Antwort.

Lösung: Die Worst-Case-Laufzeit liegt in $\Theta(n^2)$, da das Sortieren der Punkte die Laufzeit dominiert und Insertionsort eine Worst-Case-Laufzeit von $\Theta(n^2)$ besitzt.

Vorname	Name	Matr.-Nr.

Aufgabe 5 Graphen (4+2+4=10 Punkte)

a) Gegeben sei der folgende Graph:



Ermitteln Sie die starken Zusammenhangskomponenten (SCCs) in obigem Graph mit Hilfe des in der Vorlesung vorgestellten Sharir-Algorithmus.

Geben Sie sowohl den Stack, der in der ersten Phase erzeugt wird, als auch die Leiter und die zugehörigen SCCs aus der zweiten Phase an.

Lösung:

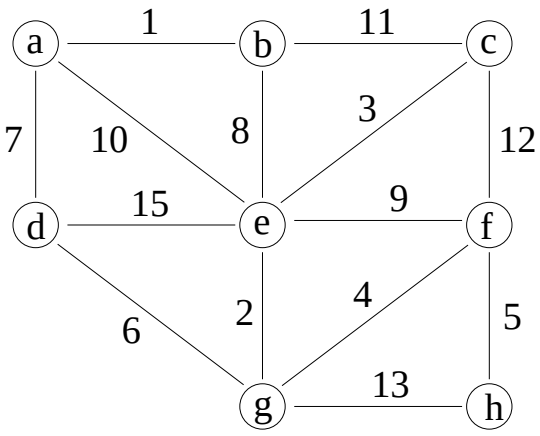
Stack nach erster Phase: $[3, 1, 4, 5, 2, 6]$

Die Leiter sind $3, 1, 2$

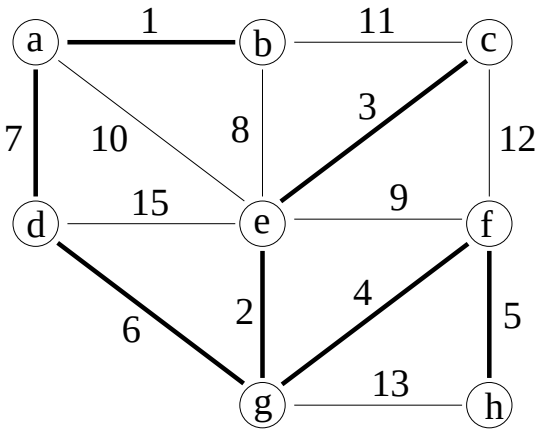
Die zugehörige SCC sind $\{3\}, \{2, 6\}, \{1, 4, 5\}$

Vorname	Name	Matr.-Nr.

b) Bestimmen Sie mittels des Algorithmus von Prim einen minimalen Spannbaum des nachfolgenden Graphen. Geben Sie an, in welcher Reihenfolge die Kanten in den Spannbaum eingefügt werden. Beginnen Sie beim Knoten a.



Lösung:



Die Reihenfolge der Kanten lautet $(a,b),(a,d),(d,g),(g,e),(e,c),(g,f),(f,h)$.
Die Reihenfolge der Knoten lautet a,b,d,g,e,c,f,h.

Vorname	Name	Matr.-Nr.

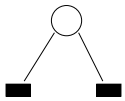
c) Ein Quasi-Rot-Schwarz-Baum ist ein Rot-Schwarz-Baum, dessen Wurzel rot anstatt schwarz ist, ansonsten aber alle Eigenschaften des Rot-Schwarz-Baumes erfüllt.

Sei T_h ein Quasi-Rot-Schwarz Baum mit Schwarz-Höhe h. Zeigen Sie per Induktion, dass T_h höchstens $\frac{1}{2}(4^h) - 1$ innere Knoten besitzt.

Lösung:

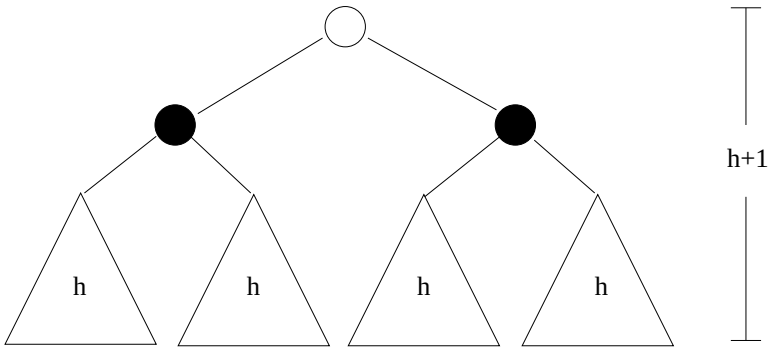
Erinnerung: Die Schwarz-Höhe $bh(x)$ eines Knotens x ist die Anzahl schwarzer Knoten bis zu einem (externen) Blatt, x ausgenommen.

Sei $B(h)$ die maximale Anzahl innerer Knoten von T_h.

- Induktionsanfang: Für $h = 1$ haben wir $T_1 =$  mit 1 inneren Knoten und es gilt:

$$B(h) := \frac{1}{2}(4^h) - 1 = 1$$

- Induktionsvoraussetzung: Die Aussage gilt für h.
- Für $h + 1$:



Es gilt:

$$\begin{aligned} B(h+1) &= 4 \cdot B(h) + 3 \\ &\overset{IV.}{=} 4 \cdot (\frac{1}{2}(4^h) - 1) + 3 \\ &= 2 \cdot 4^h - 4 + 3 \\ &= \frac{1}{2} \cdot 4^{h+1} - 1 \end{aligned}$$

Vorname	Name	Matr.-Nr.

Aufgabe 6 Dynamische Programmierung (5+3+2=10 Punkte)

Eine Musik-CD mit einer Spielzeit von L Minuten soll zusammenstellt werden. Hierzu stehen n Lieder zur Auswahl, die eine Spielzeit von k_1, k_2, ... ,k_n Minuten besitzen. Kein Lied soll mehrfach auf einer zusammengestellten CD sein.

Gesucht ist eine Teilmenge $T \subseteq \{1, \ldots, n\}$, so dass die Summe aller Spielzeiten die Spielzeit der CD nicht überschreitet und die Gesamtspielzeit maximal ist.

a) Geben Sie eine rekursive Gleichung für das Teilproblem $C(i, l)$ an, wobei $C(i, l)$ die maximale Spielzeit für eine CD der Länge $l \leq L$ und den Liedern $1, \ldots, i$, $i \leq n$ ist.

Lösung:

$$C(i,l) = \begin{cases} 0 & \text{für } i = 0, \\ C(i-1,l) & \text{für } l < k_i, \\ max\{C(i-1,l), C(i-1,l-k_i)+k_i\} & \text{sonst.} \end{cases}$$

Vorname	Name	Matr.-Nr.

b) Geben Sie eine Implementierung nach dem Prinzip der dynamischen Programmierung an, die die maximale Spielzeit für eine gegebenen CD mit $L \geq 0$ Minuten und Liedlängen $k_1, k_2, \cdots, k_n$ berechnet.

Lösung:

```
1  int musikauswahl(int k[n], int L){
2      int C[n+1,L+1];
3
4      for (int l = 0; l <= L; l++)
5          C[0,l] = 0;
6
7      for (int i = 1; i <= n; i++){
8          for (l = 0; l <= L; l++){
9              if (l < k[i-1])
10                 C[i,l] = C[i-1,l];
11             else
12                 C[i,l] = max(C[i-1,l], C[i-1,l-k[i-1]] + k[i-1]);
13         }
14     }
15
16     return C[n,L];
17 }
```

c) Geben Sie die Worst-Case Zeit- und Speicherkomplexität des von Ihnen entworfenen Algorithmus an. Begründen Sie Ihre Antwort.

Lösung: Der angegebene Algorithmus enthält zwei Schleifen. Die erste wird L-fach durchlaufen. Bei der zweiten handelt es sich um eine verschachtelte Schleife wobei die äußere n-fach und die innere L-fach durchlaufen wird. Somit kann die erste, einfache Schleife vernachlässigt werden und es ergibt sich eine Gesamtlaufzeit in $\Theta(n \cdot L)$.

Neben einigen Hilfsvariablen, die einen konstanten Platzbedarf verursachen, wird ein Feld der Größe $n \cdot L$ genutzt. Der Platzbedarf liegt somit in $\Theta(n \cdot L)$