/ 15 P

Aufgabe 1.

Hinweise:

1) In dieser Aufgabe sollen Sie **nur die Ergebnisse** angeben. Diese können Sie direkt bei den Aufgaben notieren.
2) Sofern Sie die Notationen, Algorithmen und Datenstrukturen aus der Vorlesung "Datenstrukturen & Algorithmen" verwenden, sind Erklärungen oder Begründungen nicht notwendig. Falls Sie jedoch andere Methoden benutzen, müssen Sie diese **kurz** soweit erklären, dass Ihre Ergebnisse verständlich und nachvollziehbar sind.
3) Als Ordnung verwenden wir für Buchstaben die alphabetische Reihenfolge, für Zahlen die aufsteigende Anordnung gemäss ihrer Grösse.

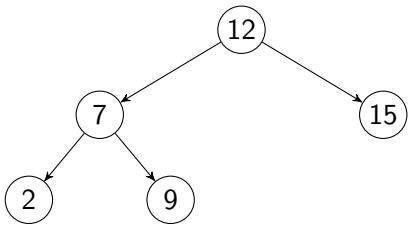
/ 1 P

a) Fügen Sie die Schlüssel 27, 14 und 16 in dieser Reihenfolge in die untenstehende Hashtabelle ein. Benutzen Sie Double Hashing mit der Hashfunktion $h(k) = k \bmod 11$ und benutzen Sie $h'(k) = 1 + (k \mod 9)$ zur Sondierung (nach *links*, d.h. $h'(k)$ soll abgezogen werden).

			3		5			19		
0	1	2	3	4	5	6	7	8	9	10

/ 1 P

b) Sie sollen *einen* ganzzahligen Schlüssel in den folgenden AVL-Baum einfügen, sodass es zu einer *Doppelrotation* kommt. Alle im AVL-Baum gespeicherten Schlüssel müssen paarweise verschieden sein. Geben Sie *alle möglichen* ganzzahligen Kandidaten an.



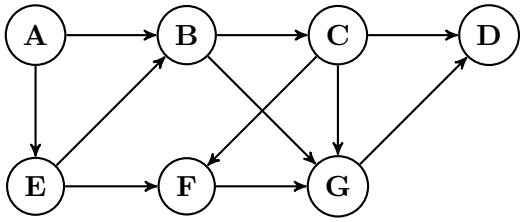
Kandidaten: _____.

/ 1 P

c) Gegeben sei das folgende Array, das nach einem Quicksort-Aufteilungsschritt entstanden ist. Welcher Schlüssel wurde als Pivotelement verwendet? Markieren Sie *alle* möglichen Kandidaten.

2	1	3	5	4	6	9	8	7
1	2	3	4	5	6	7	8	9

/ 1 P d) Geben Sie eine topologische Sortierung des untenstehenden Graphen an.



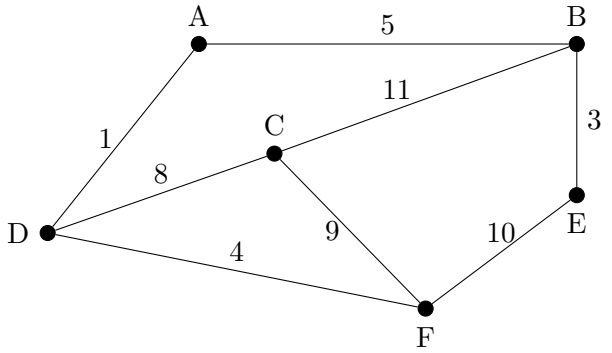
Topologische Sortierung:

____, ____, ____, ____, ____, ____, ____.

/ 1 P e) Zeichnen Sie alle 2, 3-Bäume (B-Bäume mit bis zu 3 Kindern je Knoten) für die Schlüsselmenge $\{1, 2, 3, 4, 5\}$.

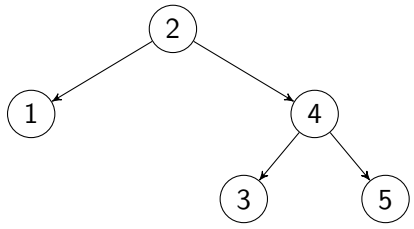
/ 1 P f) Beweisen oder widerlegen Sie: Sei $G = (V, E)$ ein ungerichteter Graph, $s, t \in V$ zwei Knoten und $T = (V, E')$ mit $E' \subseteq E$ ein minimaler Spannbaum von G. Der Pfad zwischen s und t in T ist ein kürzester Pfad in G.

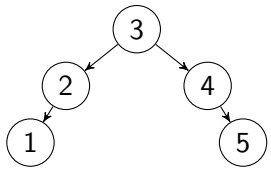
/ 1 P g) Markieren Sie in der untenstehenden Abbildung die ersten *drei* Kanten, die der Algorithmus von Jarník, Prim und Dijkstra *ausgehend von Knoten A* in den minimalen Spannbaum aufnimmt.

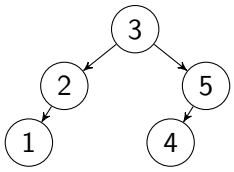


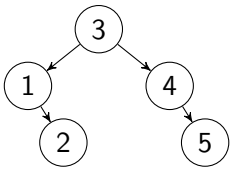
/ 1 P

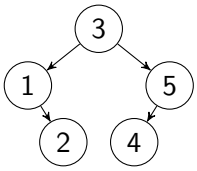
h) Welcher Splay-Baum entsteht, wenn im folgenden Splay-Baum Schlüssel 1 und Schlüssel 3 in dieser Reihenfolge angefragt werden?

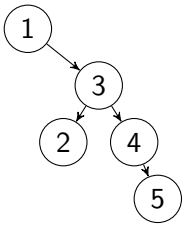


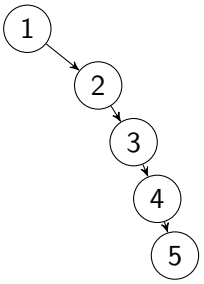
☐ 

☐ 

☐ 

☐ 

☐ 

☐ 

/ 1 P

i) Gegeben seien ein Array $A[1..n]$ und die folgende Java-Implementation des Algorithmus *Sortieren durch Einfügen*. Der Aufruf von `swap(A, i, j)` vertauscht die Elemente $A[i]$ und $A[j]$. Um ein Array $A[1..n]$ aufsteigend zu sortieren wird die Funktion mit den Parametern $l = 1$ und $r = n$ aufgerufen. Vervollständigen Sie die Implementation (Zeilen 2, 3 und 4).

```
1 public void insertionSort(int[] A, int l, int r) {
2     for (int i=_______; i<=_______; i++)
3         for (int j=i-1; j>=l && A[j]>A[_______]; j--)
4             swap(A, _______, _______);
5 }
```

/ 3 P

j) Gegeben ist die folgende Rekursionsgleichung:

$$T(n) := \begin{cases} 3 \cdot T(n/7) + 4 & n > 1 \\ 3 & n = 1 \end{cases}$$

Geben Sie eine geschlossene (nicht-rekursive) und *möglichst einfache* Formel für $T(n)$ an und beweisen Sie diese mit vollständiger Induktion. Sie können annehmen, dass n eine Potenz von 7 ist. Benutzen Sie also $n = 7^k$ oder $k = \log_7(n)$.

Hinweis: Für $q \neq 1$ gilt: $\sum_{i=0}^{k} q^i = \frac{q^{k+1}-1}{q-1}$.

Herleitung (falls benötigt):

Geschlossene und vereinfachte Formel:

$T(n) = T(7^k) =$

Induktionsbeweis:

/ 1 P

k) Geben Sie die asymptotische Laufzeit in Abhängigkeit von $n \in \mathbb{N}$ für den folgenden Algorithmus (so knapp wie möglich) in Θ-Notation an. Sie müssen Ihre Antwort nicht begründen.

```
1 for ( int i = n; i >= 1; i = i / 2 ) {
2         int j = n;
3         while ( j >= 1 ) {
4                 j = j − 20;
5         }
6 }
```

Laufzeit in möglichst knapper Θ-Notation:

/ 1 P

l) Geben Sie die asymptotische Laufzeit in Abhängigkeit von $n \in \mathbb{N}$ für den folgenden Algorithmus (so knapp wie möglich) in Θ-Notation an. Sie müssen Ihre Antwort nicht begründen.

```
1 int f( int n ) {
2         if ( n <= 1 ) return 1;
3         for ( int i = 1; i · i <= n; i = i + 2 ) {
4                 ;
5         }
6         return f( n/4 ) + 1;
7 }
```

Laufzeit in möglichst knapper Θ-Notation:

/ 1 P

m) Geben Sie für die untenstehenden Funktionen eine **Reihenfolge** an, so dass folgendes gilt: Wenn eine Funktion f links von einer Funktion g steht, dann gilt $f \in \mathcal{O}(g)$.

Beispiel: Die drei Funktionen n^3, n^7, n^9 sind bereits in der entsprechenden Reihenfolge, da $n^3 \in \mathcal{O}(n^7)$ und $n^7 \in \mathcal{O}(n^9)$ gilt.

$$4^{n/2},\ \sqrt{n},\ \binom{n}{4},\ 10^{64},\ \log(n!),\ n^n,\ \log(n^{16})$$

Lösung: _____, _____, _____, _____, _____, _____, _____.

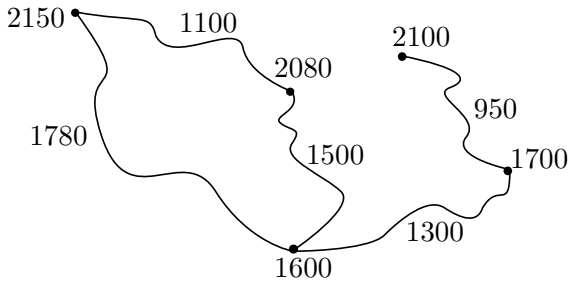
/ 12 P

Aufgabe 2.

In einem Skigebiet gibt es n Stationen, die mit m Pisten verbunden sind. Die Stationen $i \in \{1, \ldots, n\}$ sind nach ihrer Höhe $h_i \in \mathbb{N}$ über dem Meeresspiegel absteigend sortiert, das heisst aus $i < j$ folgt $h_i \geq h_j$. Die Pisten sind als Paare (i, j) für Stationen $i, j \in \{1, \ldots, n\}$ mit $i \neq j$ gegeben, wobei für jede Piste $h_i > h_j$ gilt. Weiters sind die Längen der Pisten $l_1, l_2, \ldots, l_m$ gegeben mit $l_i \in \mathbb{N}$ für alle $i \in \{1, \ldots, m\}$, die maximale Höhe im Skigebiet $H = \max_{i \in \{1,\ldots,n\}} h_i$ und die Gesamtlänge aller Pisten im Skigebiet $L = \sum_{i=1}^{m} l_i$.

Eine Abfahrt ist eine Abfolge von Pisten, die Sie ohne Unterbrechung befahren, d.h. mit Ausnahme der letzten Piste gilt, dass die Endstation einer Piste zugleich die Anfangsstation der nächsten Piste ist. Die Länge einer Abfahrt ist die Summe der Längen der befahrenen Pisten. Berechnen Sie eine längste Abfahrt im Skigebiet.

Beispiel: Es gibt $n = 5$ Stationen (Punkte) und $m = 5$ Pisten (Kurvenstücke). Die Höhen der Stationen und die Längen der Pisten sind im Bild vermerkt. Die Gesamtlänge aller Pisten ist $L = 6630$ Meter und die maximale Meereshöhe $H = 2150$ Meter. Die längste Abfahrt hat eine Länge von 2600 Meter.



/ 7 P

a) Geben Sie einen möglichst effizienten Algorithmus an, der nach dem Prinzip der dynamischen Programmierung arbeitet und die Länge einer längsten Abfahrt berechnet. Definieren Sie die Grösse der DP-Tabelle und die Bedeutung eines Eintrags und gehen Sie auf die folgenden Aspekte ein.

1) Was ist die Bedeutung eines Tabelleneintrags, und welche Grösse hat die DP-Tabelle?
2) Wie berechnet sich ein Tabelleneintrag aus früher berechneten Einträgen?
3) In welcher Reihenfolge können die Einträge berechnet werden?
4) Wie kann aus der DP-Tabelle die Länge einer längsten Abfahrt ausgelesen werden?

Wählen Sie als erstes die Dimension Ihrer DP-Tabelle:

- ☐ **(I) eindimensional** → Benutzen Sie das Schema auf Seite 8.
- ☐ **(II) zweidimensional** → Benutzen Sie das Schema auf Seite 9.
- ☐ **(III) andere Dimension** → Beschreiben Sie Ihr dynamisches Programm auf Seite 10.

Beantworten Sie die Teilaufgaben b) und c) auf Seite 11.

/ 2 P

b) Beschreiben Sie detailliert, wie aus der DP-Tabelle ihres dynamischen Programms abgelesen werden kann, aus welchen Pisten eine mögliche längste Abfahrt zusammengesetzt ist.

/ 3 P

c) Geben Sie die Laufzeit des in a) und b) entwickelten dynamischen Programms an und begründen Sie Ihre Antwort. Ist die Laufzeit polynomiell? Begründen Sie Ihre Antwort.

(I) Für Algorithmen mit einer eindimensionalen DP-Tabelle:

Grösse der DP-Tabelle / Anzahl Einträge:

☐ n	☐ 2^n	☐ $L+1$	☐ andere:
☐ m	☐ 2^m	☐ $H+1$	_____

Bedeutung eines Tabelleneintrags:

Bedeutung des Eintrags an Position i:

$DP[i]$: _____

_____

Initialisierung:

Berechnung eines Eintrags:

$DP[i] =$

Berechnungsreihenfolge:

Auslesen der Länge einer längsten Abfahrt:

(II) Für Algorithmen mit einer zweidimensionalen DP-Tabelle:

Grösse der DP-Tabelle: *Anzahl Zeilen*

☐ n ☐ m ☐ andere: _____

Grösse der DP-Tabelle: *Anzahl Spalten*

☐ n ☐ $L+1$ ☐ andere: _____

☐ m ☐ $H+1$

Bedeutung eines Tabelleneintrags:

Bedeutung des Eintrags in Zeile i und Spalte j:

$DP[i,j]$: _____

_____

Initialisierung:

Berechnung eines Eintrags:

$DP[i,j] =$

Berechnungsreihenfolge:

Auslesen der Länge einer längsten Abfahrt:

(III) Für Algorithmen mit einer DP-Tabelle von Dimension ≥ 3*:*

Teilaufgaben b) und c)

b) Beschreiben Sie detailliert, wie aus der DP-Tabelle ihres dynamischen Programms abgelesen werden kann, aus welchen Pisten eine mögliche längste Abfahrt zusammengesetzt ist.

c) Geben Sie die Laufzeit des in a) und b) entwickelten dynamischen Programms an und begründen Sie Ihre Antwort. Ist die Laufzeit polynomiell? Begründen Sie Ihre Antwort.

/ 7 P

Aufgabe 3.

In einem Krankenhaus sollen Blutkonserven an Patienten verteilt werden. Bei der Zuteilung der Konserven an die Patienten muss die Blutgruppenkompatibilität beachtet werden. In der nebenstehenden Tabelle ist die Kompatibilität der Blutgruppen zusammengefasst. Beispielsweise kann ein Patient mit Blutgruppe A Konserven der Blutgruppen 0 oder A erhalten, während ein Patient mit Blutgruppe 0 nur Konserven der Blutgruppe 0 erhalten kann.

	Konserve			
Patient	**0**	**A**	**B**	**AB**
AB	✓	✓	✓	✓
A	✓	✓		
B	✓		✓	
0	✓			

Gegeben sind die Anzahl k_a, k_b, k_{ab} und k_0 der verfügbaren Konserven für die Blutgruppen A, B, AB und 0 und die Anzahl b_a, b_b, b_{ab} und b_0 der Patienten mit Blutgruppe A,B, AB und 0, die je eine Konserve benötigen. Entwickeln Sie einen Algorithmus, der berechnet, ob eine Zuteilung existiert, sodass jeder Patient eine Konserve mit einer kompatiblen Blutgruppe erhält.

Beispiel: Die Anzahl der Konserven und der Patienten sind in der Tabelle rechts gegeben. Bei folgender Zuteilung erhält jeder Patient eine kompatible Konserve: Konserven der Blutgruppe 0 werden an Patienten mit Blutgruppe 0 (2 Konserven) und Blutgruppe B (1 Konserve) verteilt, Konserven der Blutgruppe A an Patienten mit Blutgruppe A (3 Konserven) und Blutgruppe AB (1 Konserve), und die Konserven der restlichen Blutgruppen jeweils an Patienten mit derselben Blutgruppe.

Blutgruppe	**0**	**A**	**B**	**AB**
Bedarf	2	3	2	4
Lager	3	4	1	3

/ 4 P a) Modellieren Sie das oben genannte Problem als Flussproblem. Beschreiben Sie dazu die Konstruktion eines geeigneten Netzes $N = (V, E, c)$ mit der Knotenmenge V sowie der Kantenmenge E, und geben Sie an, welche Kapazitäten c die Kanten besitzen sollen. Wie kann aus dem Wert eines maximalen Flusses abgelesen werden, ob eine Zuteilung existiert?

/ 3 P b) Nehmen Sie an, das Flussproblem aus a) wurde bereits gelöst, d.h. zu einem maximalen Fluss φ in N kennen Sie den Fluss φ_e auf jeder Kante $e \in E$. Nehmen Sie weiter an, eine Zuteilung existiert, sodass jeder Patient eine Konserve mit einer kompatiblen Blutgruppe erhält. Beschreiben Sie detailliert einen Algorithmus, der aus den φ_e eine solche Zuteilung berechnet. Welche Laufzeit hat Ihr Algorithmus, wenn auf jedes φ_e in konstanter Zeit zugegriffen werden kann?

Teilaufgabe a)

Definition des Netzes N (wenn möglich in Worten und nicht formal):

Knotenmenge V: _____

_____

Kantenmenge E: _____

_____

Kapazitäten c: _____

_____

Schematische Darstellung / Zeichnung des Netzes:

Eine Zuteilung existiert, falls: _____

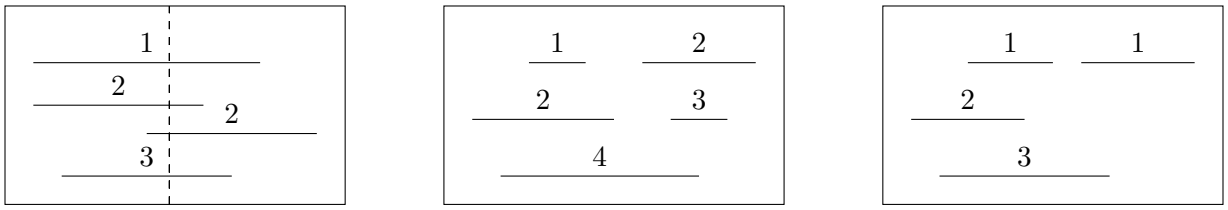
_____

Teilaufgabe b)

b) Nehmen Sie an, das Flussproblem aus a) wurde bereits gelöst, d.h. zu einem maximalen Fluss φ in N kennen Sie den Fluss φ_e auf jeder Kante $e \in E$. Nehmen Sie weiter an, eine Zuteilung existiert, sodass jeder Patient eine Konserve mit einer kompatiblen Blutgruppe erhält. Beschreiben Sie detailliert einen Algorithmus, der aus den φ_e eine solche Zuteilung berechnet. Welche Laufzeit hat Ihr Algorithmus, wenn auf jedes φ_e in konstanter Zeit zugegriffen werden kann?

Aufgabe 4. **/ 10 P**

Gegeben sei eine Menge von n horizontalen, sich nicht schneidenden Liniensegmenten in der Ebene. Jedes Liniensegment ist durch die Koordinaten seiner beiden Endpunkte (x_l, y) und (x_r, y) gegeben. Eine Nummerierung der Liniensegmente mit Zahlen aus $\{1, \ldots, m\}$ ist *zulässig*, wenn für jede vertikale Gerade gilt, dass die von ihr geschnittenen Liniensegmente von oben nach unten aufsteigend nummeriert sind. Gesucht ist die kleinste Zahl $m \leq n$, für die eine zulässige Nummerierung der Liniensegmente mit Zahlen aus $\{1, \ldots, m\}$ existiert.



Beispiele: Links: Die Nummerierung der Liniensegmente ist nicht zulässig, da die von der dargestellten vertikalen Gerade (gestrichelte Linie) geschnittenen Liniensegmente nicht aufsteigend nummeriert sind. *Mitte:* Die Nummerierung ist zulässig, es gibt jedoch auch eine zulässige Nummerierung mit $m = 3$. *Rechts:* Die Nummerierung ist zulässig und es gibt keine zulässige Nummerierung mit $m < 3$.

/ 8 P a) Entwerfen Sie einen möglichst effizienten Scanline-Algorithmus für das obige Problem. Gehen Sie in Ihrer Lösung auf die folgenden Aspekte ein.

1) In welche Richtung läuft die Scanline, und was sind die Haltepunkte?
2) Welche Objekte muss die Scanline-Datenstruktur verwalten, und was ist eine angemessene Datenstruktur?
3) Was passiert, wenn die Scanline auf einen neuen Haltepunkt trifft?
4) Wie kann die gesuchte, kleinste Zahl m, für die eine Nummerierung der Liniensegmente mit den geforderten Eigenschaften existiert, bestimmt werden?
5) Welche Laufzeit in Abhängigkeit von n hat Ihr Algorithmus? Begründen Sie Ihre Antwort.

/ 2 P b) Beschreiben Sie, wie Sie Ihren Algorithmus erweitern können, damit er zusätzlich eine zulässige Nummerierung der Liniensegmente mit den Zahlen aus $\{1, \ldots, m\}$ berechnet.

Scanline-Richtung (genau eine Möglichkeit ankreuzen):

- Die Scanline ist eine *vertikale Gerade*; sie bewegt sich
  ☐ von links nach rechts / ☐ von rechts nach links.
- Die Scanline ist eine *horizontale Gerade*; sie bewegt sich
  ☐ von oben nach unten / ☐ von unten nach oben.
- Die Scanline ist eine *Halbgerade*; sie rotiert
  ☐ im Uhrzeigersinn / ☐ im Gegenuhrzeigersinn um den Punkt _____.

☐ andere Richtung: _____

_____.

Haltepunkte:

Menge der Haltepunkte: _____

_____

sortiert nach: _____

_____

Scanline-Datenstruktur:

Operationen bei einem Haltepunkt:

Kleinste Zahl m berechnen:

Laufzeit des Algorithmus:

Teilaufgabe b)

b) Beschreiben Sie, wie Sie Ihren Algorithmus erweitern können, damit er zusätzlich eine zulässige Nummerierung der Liniensegmente mit den Zahlen aus $\{1, \ldots, m\}$ berechnet.