Theorieaufgabe T1. **/ 16 P**

Hinweise:

1) In dieser Aufgabe sollen Sie **nur die Ergebnisse** angeben. Diese können Sie direkt bei den Aufgaben notieren.
2) Als Ordnung verwenden wir für Buchstaben die alphabetische Reihenfolge, für Zahlen die aufsteigende Anordnung gemäss ihrer Grösse.

/ 3 P

a) Nachfolgend sehen Sie drei Folgen gewisser Momentaufnahmen von jeweils einem der vier Algorithmen (a) `InsertionSort` (Sortieren durch Einfügen), (b) `SelectionSort` (Sortieren durch Auswahl), (c) `QuickSort` und (d) `BubbleSort`. Geben Sie unter den Folgen jeweils den Namen des zugehörigen Algorithmus an.

5	4	1	3	2
1	4	5	3	2
1	2	5	3	4
1	2	3	5	4
1	2	3	4	5

SelectionSort

5	4	1	3	2
4	1	3	2	5
1	3	2	4	5
1	2	3	4	5

BubbleSort

5	4	1	3	2
4	5	1	3	2
1	4	5	3	2
1	3	4	5	2
1	2	3	4	5

InsertionSort

/ 2 P

b) Gegeben sei eine Menge von Frauen $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$ sowie eine Menge von Männern $\mathcal{M} = \{m_1, m_2, m_3, m_4, m_5\}$. Die folgenden beiden Tabellen zeigen jeweils ihre Präferenzen, die von links nach rechts absteigend sortiert sind.

f_1	m_1	m_2	m_3	m_4	m_5
f_2	m_4	m_2	m_3	m_5	m_1
f_3	m_4	m_1	m_5	m_2	m_3
f_4	m_2	m_4	m_3	m_5	m_1

m_1	f_2	f_4	f_1	f_3
m_2	f_2	f_1	f_4	f_3
m_3	f_3	f_4	f_1	f_2
m_4	f_3	f_1	f_4	f_2
m_5	f_1	f_2	f_3	f_4

Führen Sie den Algorithmus von Gale und Shapley aus und beantworten Sie folgende Fragen.

(i) Welcher Mann bleibt Junggeselle, wenn die Frauen die Heiratsanträge machen?

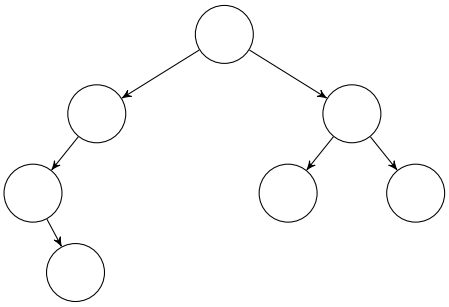
Junggeselle in (i): m_5

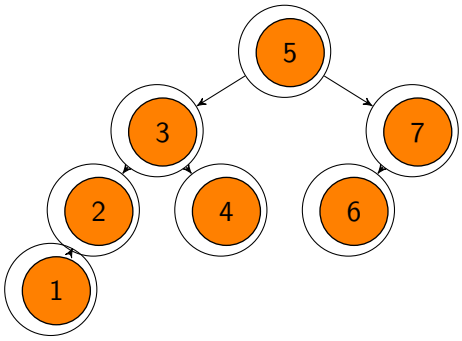
(ii) Welcher Mann bleibt Junggeselle, wenn die Männer die Heiratsanträge machen?

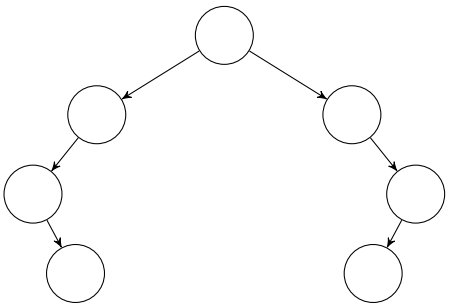
Junggeselle in (ii): m_5

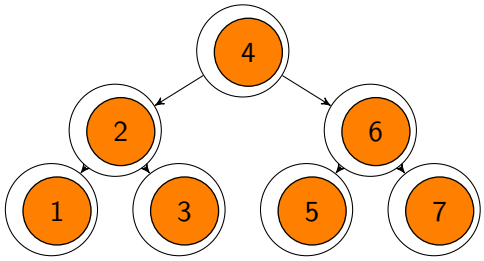
/ 1 P

c) *AVL-Bäume*: Tragen Sie die Schlüssel `1`, `2`, `3`, `4`, `5`, `6`, `7` in die Knoten von *zwei* der vier folgenden binären Suchbäume ein, sodass diese beiden Bäume gültige AVL-Bäume darstellen.



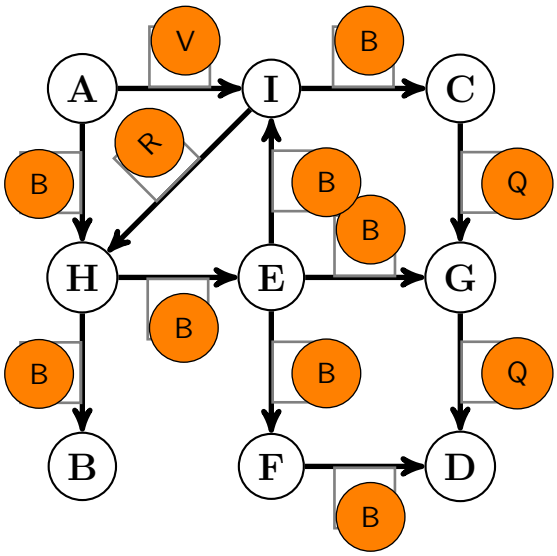






/ 2 P

d) Führen Sie im folgenden Graphen eine Tiefensuche aus. Starten Sie in Knoten A und nehmen Sie an, dass die Traversierung die jeweiligen Nachbarknoten in alphabetischer Reihenfolge besucht. Geben Sie die Tiefenordnung an und schreiben sie zu jeder Kante, ob sie eine Baumkante (B), eine Vorwärtskante (V), eine Rückwärtskante (R) oder eine Querkante (Q) ist.



Tiefenordnung:

___, ___, ___, ___, ___, ___, ___, ___, ___.

A, H, B, E, F, D, G, I, C

/ 1P

e) Wenden Sie auf den nachfolgenden Array von Zahlen den Algorithmus von Blum (Median der Mediane) an. Geben Sie dazu den **ersten** Median der Mediane an.

11	54	3	18	24	21	7	51	63	12	8	14	33	25	9

Erster Median der Mediane:

18

/ 1 P

f) Geben Sie für das folgende Codefragment die asymptotische Laufzeit in Abhängigkeit von $n \in \mathbb{N}$ in **möglichst knapper Θ-Notation** an. Die Funktion f läuft in $\Theta(1)$ Zeit. Sie müssen Ihre Antwort nicht begründen.

```
1 for(int i = 1; i < n; i = 3*i ) {
2         for(int j = i; j ≥ 1; j = j/2 ) {
3                 f();
4         }
5 }
```

Asymptotische Laufzeit:

$\Theta(\log^2 n)$

/ 1 P

g) Geben Sie für das folgende Codefragment die asymptotische Laufzeit in Abhängigkeit von $n \in \mathbb{N}$ in **möglichst knapper Θ-Notation** an. Die Funktion f läuft in $\Theta(1)$ Zeit. Sie müssen Ihre Antwort nicht begründen.

```
1 for(int i = 1; i ≤ n; i = 2*i ) {
2         for(int j = 1; j*j*j ≤ n; j = j+1 ) {
3                 for(int k = 1; k ≤ i*i; k = k+i ) {
4                         f();
5                 }
6         }
7 }
```

Asymptotische Laufzeit:

$\Theta(n^{4/3})$

/ 1P

h) Führen Sie eine Extract-Min Operation einschliesslich Wiederherstellung der Heap-Bedingungen auf folgendem Min-Heap aus. Wie sieht der Heap nach der Operation aus?

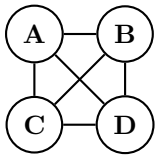
12	14	50	30	15	52	63	61	35	18	99	67

Heap nach dem Löschen:

14 — 15 — 50 — 30 — 18 — 52 — 63 — 61 — 35 — 67 — 99

/ 2P

i) Beantworten Sie nachfolgende Fragen. Jede korrekte Antwort gibt 0,5 Punkte, für jede falsche oder fehlende Antwort werden 0,5 Punkte abgezogen. Insgesamt gibt die Aufgabe mindestens 0 Punkte. Sie müssen Ihre Antworten nicht begründen.

Wie viele Kanten hat ein ungerichteter Graph, wenn die Summe seiner Knotengrade 96 beträgt?	48
Aus wie vielen Zusammenhangskomponenten besteht ein Wald mit 10 Kanten und 17 Knoten?	7
Wie viele Blätter (Knoten mit Knotengrad 1) hat obiger Wald mindestens?	2
Wie viele verschiedene Spannbäume gibt es im abgebildeten vollständigen Graphen mit 4 Knoten? 	16

/ 2 P

j) Gegeben ist die folgende Rekursionsgleichung:

$$T(n) := \begin{cases} 72 + 9 \cdot T(\frac{n}{3}) + 4n & n > 1 \text{ und } n \text{ eine Dreierpotenz} \\ 13 & n = 1 \end{cases}$$

Sie können annehmen, dass n eine Potenz von 3 ist. Zeigen Sie mit vollständiger Induktion, dass

$$T(n) = 24n^2 - 2n - 9$$

die dazugehörige geschlossene Form ist.

Induktionsbeweis:

Induktionsanfang ($n = 1$)*:* Es gilt $T(1) = 13 = 24 \cdot 1^2 - 2 \cdot 1 - 9$

Induktionshypothese (I.H.): Für ein $n \geq 1$ sei $T(n) = 24n^2 - 2n - 9$.

Induktionsschritt ($n \to 3n$)*:*

$$\begin{aligned} T(3n) &= 72 + 9 \cdot T(n) + 4 \cdot (3n) \\ &\overset{I.H.}{=} 72 + 9 \cdot (24n^2 - 2n - 9) + 4 \cdot (3n) \\ &= 72 + 24 \cdot (3n)^2 - 6 \cdot (3n) - 81 + 4 \cdot (3n) \\ &= 24 \cdot (3n)^2 - 2 \cdot (3n) - 9 \end{aligned}$$

Theorieaufgabe T2. **/ 10 P**

Mit der Einführung von Inland-Fernbussen, Spartickets und Zonenabonnements wird es für Sie als studentische(n) Wochenaufenthalter(in) immer unklarer, ob sich ein Generalabonnement noch lohnt. Entscheidend ist, wie teuer Sie jeweils am Freitagabend eine Heimfahrt zu stehen kommt. Selbstverständlich möchten Sie dabei nicht auf Komfort verzichten, und deshalb *so selten wie möglich umsteigen*.

Vorhanden ist eine Liste von m verschiedenen Strecken mit Preisangabe im Format $s_i\ t_i\ p_i$: Die i-te Strecke kann ohne Umsteigen von s_i nach t_i zum Preis von $p_i > 0$ Franken befahren werden. Im Anschluss können Sie auf eine beliebige Strecke j mit $s_j = t_i$ umsteigen.

Gesucht ist *der Preis einer günstigsten Reise* von Zürich zu Ihrem Wohnort, wobei Sie *möglichst wenig oft umsteigen möchten*. Sie dürfen annehmen, dass so eine Reise immer existiert.

Beispiel. Sie wohnen in Disentis und haben die folgenden Strecken zur Auswahl:
`Zürich Chur 30.-, Zürich Bellinzona 40.-, Zürich Erstfeld 20.-,`
`Chur Disentis 15.-, Bellinzona Disentis 10.-,`
`Erstfeld Andermatt 10.-, Andermatt Disentis 10.-`
Dann ist die gesuchte Reise Zürich—Chur—Disentis zum Preis von 45.- (die Reise via Bellinzona ist teurer; die Reise via Erstfeld–Andermatt ist zwar günstiger, benötigt aber mehr Umsteigevorgänge).

Aufgabe. Modellieren Sie die Fragestellung als graphentheoretisches Problem. Geben Sie an, welche Knoten und Kanten definiert werden und gegebenenfalls welche Gewichte den Kanten zugeordnet werden. Beschreiben Sie insbesondere, wo sich der gesuchte Preis in ihrer Modellierung wiederfindet. Entwerfen Sie einen möglichst effizienten Algorithmus zur Lösung dieses Problems und bestimmen Sie seine Laufzeit in Abhängigkeit von m.

- Graphentheoretische Modellierung inklusive allfälliger Kantengewichte (in Worten).
  Verwenden Sie Zürich als Knoten Z und den Wohnort als Knoten W:

Lösung 1: Modifizierter Graph + Standardalgorithmus

Lösung 1a Gerichteter Graph $G = (V, E, w),\ w\colon E \to \mathbb{R}$:
Jeder in der Liste vorkommende Ort entspricht einem Knoten. Sei $p_{\text{sum}} := \sum_{i=1}^{m} p_i$.
Jede Strecke entspricht einer Kante vom Knoten s_i zum Knoten t_i mit Gewicht $p_{\text{sum}} + p_i$.

Lösung 1b Gerichteter Graph $G = (V, E, w),\ w\colon E \to \mathbb{R}^2$:
Jeder in der Liste vorkommende Ort entspricht einem Knoten.
Jede Strecke entspricht einer Kante vom Knoten s_i zum Knoten t_i mit Gewicht $(1,\ p_i)$.

Lösung 2: Standardgraph + modifizierter Algorithmus
Gerichteter Graph $G = (V, E, w),\ w\colon E \to \mathbb{R}$:
Jeder in der Liste vorkommende Ort entspricht einem Knoten.
Jede Strecke entspricht einer Kante vom Knoten s_i zum Knoten t_i mit Gewicht p_i.

- Der gesuchte Preis entspricht in obiger Modellierung:

  Dem Gesamtgewicht eines kürzesten gewichteten Z-W-Pfads in G, abzüglich eines Gewichts von $p_{\text{sum}} \cdot (\#$ Kanten dieses Pfads). **Lösung 1a**

  Dem zweiten Eintrag des Gesamtgewicht-Tupels eines lexikographisch kürzesten tupel-gewichteten Z-W-Pfads in G. **Lösung 1b**

  Dem Gesamtgewicht eines kürzesten gewichteten Z-W-Pfads unter allen kürzesten *ungewichteten* Z-W-Pfaden in G. **Lösung 2**

- Entwurf eines möglichst effizienten Algorithmus:

  Dijkstra in G von Z aus liefert Distanzen und Vorgänger von jedem Knoten. **Lösung 1a**
  Damit finden wir Gesamtgewicht und Anzahl Kanten eines kürzesten Z-W-Pfades.
  Der gesuchte Preis ist gleich: Gesamtgewicht $- p_{\text{sum}} \cdot (\#$ Kanten des Pfads).

  Wir verwenden Dijkstra in G von Z aus. Für die Priority Queue **Lösung 1b**
  verwenden wir die lexikographische Ordnung; Tupel werden jeweils eintragsweise addiert.
  Der erste Eintrag des Tupels bei Z entspricht dann der minimalen Anzahl Kanten eines jeden Z-W-Pfads, der zweite Eintrag dem gesuchten Preis.

  Zuerst löschen wir alle Kanten, welche *nicht* auf einem **Lösung 2a**
  kürzesten ungewichteten Z-W-Pfad in G liegen. Wir erstellen einen
  transponierten Graph $G^T = (V, E^T)$, und verfahren wie folgt:
  1. Breitensuche in G von Z aus liefert die ungewichteten Distanzen $d(Z, s_i)$ und $d(Z, W)$.
  2. Breitensuche in G^T von W aus liefert die ungewichteten Distanzen $d(t_i, W)$.
  3. Wir löschen in G alle Kanten i für welche gilt: $d(Z, s_i) + 1 + d(t_i, W) > d(Z, W)$.

  Auf dem *verbleibenden* Graphen lassen wir Dijkstra von Z aus laufen, und finden im Distanzeintrag für W den gesuchten Preis.

  Wir verwenden einen abgekürzten Bellman-Ford-Algorithmus: **Lösung 2b**
  Anstatt die äussere DP-Schleife $|V| - 1$ mal zu durchlaufen, stoppen wir nach demjenigen Durchgang D, in welchem zum ersten Mal eine Distanz $d(Z, W) < \infty$ berechnet wurde:
  Dann benötigt nämlich jeder Z-w-Pfad mindestens D Kanten, und die berechnete Distanz entspricht dem kürzesten Pfad mit genau D Kanten.

- Laufzeit in Abhängigkeit von m (in möglichst knapper Θ-Notation) inklusive Begründung:

  Totale Laufzeit $\Theta(m \log m)$, bestehend aus: **Lösungen 1a/1b/2a**
  - Aufbau des/der Graphen von Grösse $|V|, |E| \in \mathcal{O}(m)$ in Zeit $\Theta(m)$,
  - 2x BFS in Zeit $\Theta(m)$ (nur Lösung 2a), 1x Dijkstra in Zeit $\Theta(m \log m)$.

  Totale Laufzeit $\mathcal{O}(m^2)$, bestehend aus: **Lösung 2b**
  - Aufbau des/der Graphen von Grösse $|V|, |E| \in \mathcal{O}(m)$ in Zeit $\Theta(m)$,
  - 1x Bellman-Ford in Zeit $\mathcal{O}(D \cdot |E|) \subseteq \mathcal{O}(|V| \cdot |E|) \subseteq \mathcal{O}(m^2)$.

Theorieaufgabe T3. **/ 14 P**

Die folgenden (voneinander unabhängigen) Teilaufgaben drehen sich um *minimale Spanngraphen*. Gegeben sei ein ungerichteter zusammenhängender Graph $G = (V, E, c)$ in Adjazenzlistenform, mit $n := |V|$ Knoten, $m := |E|$ Kanten und Kantenkostenfunktion $c\colon E \to \mathbb{R}$.

Ein *Spanngraph* ist ein zusammenhängender Teilgraph, welcher alle Knoten enthält. Gesucht ist ein *minimaler Spanngraph*, also ein Spanngraph $G' = (V, E')$ mit $E' \subseteq E$, sodass $\sum_{e \in E'} c(e)$ minimal ist.

/ 4 P

a) Nehmen Sie in dieser Teilaufgabe an, dass alle Kantenkosten positiv sind, $\forall e \in E\colon c(e) > 0$. Beweisen Sie, dass in diesem Fall jeder minimale Spanngraph ein Baum ist. Geben Sie einen möglichst effizienten Algorithmus an, welcher einen minimalen Spanngraphen berechnet, und nennen Sie dessen Laufzeit.

Beweis:

Ein minimaler Spanngraph G' ist nach Definition zusammenhängend. Ist G' kein Spannbaum, enthält er dementsprechend mehr als $n - 1$ Kanten und somit einen Kreis.

Entfernen wir eine beliebige Kante dieses Kreises, so bleibt G' zusammenhängend. Da jede Kante positives Gewicht hat, werden durch das Entfernen einer Kante die Kosten strikt kleiner, im Widerspruch zur Minimalität von G'.

Möglichst effizienter Algorithmus:

Algorithmus von Prim (berechnet einen minimalen Spannbaum, nach obigem Beweis also auch einen minimalen Spanngraphen).

Laufzeit in Abhängigkeit von n, m (in möglichst knapper Θ-Notation):

Bei Verwenden von Fibonacci-Heaps erhalten wir eine Laufzeit von $\Theta(n \log n + m)$.

/ 5 P

b) In dieser Teilaufgabe gibt es *keine* Einschränkungen der Kantenkosten, es können also auch nicht-positive Kosten $c(e) \leq 0$ auftreten.

Entwerfen Sie einen möglichst effizienten Algorithmus, welcher einen minimalen Spanngraphen berechnet. Analysieren Sie auch dessen Laufzeit.

Algorithmenentwurf:

Wir sortieren die Kanten aufsteigend nach ihren Kosten, $c(e_1) \leq c(e_2) \leq \cdots \leq c(e_m)$.

Wir initialisieren $E' = \emptyset$ und gehen die Kanten nach aufsteigendem Gewicht durch.
Dabei verwenden wir eine modifizierte Variante von Kruskal:
- Ist $c(e_k) \leq 0$, so fügen wir e_k zum Spanngraphen hinzu: $E' \leftarrow E' \cup \{e_k\}$.
- Ist hingegen $c(e_k) > 0$, so verwenden wir die Regel von Kruskal:
— Verbindet e_k zwei Knoten derselben Zusammenhangskomponente, verwerfen wir e_k.
— Sonst fügen wir e_k zum Spanngraphen hinzu: $E' \leftarrow E' \cup \{e_k\}$.

Laufzeit in Abhängigkeit von n, m (in möglichst knapper Θ-Notation, inklusive *Begründung):*

Totale Laufzeit von $\Theta(m \log m)$, zusammengesetzt aus:
- $\Theta(m \log m)$ für das Sortieren der Kanten.
- $\mathcal{O}(m)$ vielen Union-Find Operationen für die Kruskalregel, je amortisiert $\mathcal{O}(\alpha(m, n))$.

/ 5 P

c) Nehmen Sie in dieser Teilaufgabe an, dass alle Kantenkosten positiv sind, das heisst $\forall e \in E\colon c(e) > 0$. Sie haben für den Graphen G bereits einen minimalen Spannbaum T von G berechnet. Nun erhalten Sie eine zusätzliche Kante e^* mit Kosten $c(e^*) > 0$.

Entwerfen Sie einen möglichst effizienten Algorithmus, welcher einen minimalen Spannbaum des um die Kante e^* erweiterten Graphen G berechnet. Nutzen Sie dazu die Kenntnis von T möglichst gut aus. Analysieren Sie auch die Laufzeit Ihres Algorithmus.

Algorithmenentwurf:

Wir fügen die Kante e^* zu T hinzu. Dabei erhalten wir einen zusammenhängenden Graphen mit genau n Knoten und n Kanten und dementsprechend genau einem Kreis. Mittels Breitensuche in $T \cup \{e^*\}$ finden wir diesen Kreis in linearer Zeit. Nun suchen wir die Kante $e_{\max}$ des Kreises mit den höchsten Kosten $c(e_{\max})$; dies kann entweder eine ursprüngliche Kante von T oder auch e^* selbst sein.

Wir entfernen nun die gefundene Kante aus unserem Spanngraphen. Der so resultierende Graph $T \cup \{e^*\} \setminus \{e_{\max}\}$ bleibt zusammenhängend und ist ein minimaler Spanngraph des um die Kante e^* erweiterten Graphen G.

Laufzeit in Abhängigkeit von n, m (in möglichst knapper Θ-Notation, inklusive *Begründung):*

Die Laufzeit beträgt lediglich $\Theta(n)$:

Die Breitensuche in $T \cup \{e^*\}$ kann in $\Theta(n)$ durchgeführt werden. Dabei finden und speichern wir den enthaltenen Kreis. In diesem finden und löschen wir wiederum in Zeit $\mathcal{O}(n)$ die teuerste Kante.