D&A Aufgabensammlung

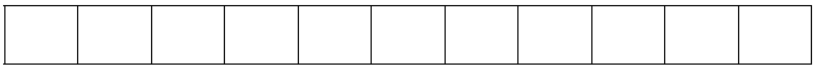
Inhaltsverzeichnis

Kurzaufgaben

Hashing

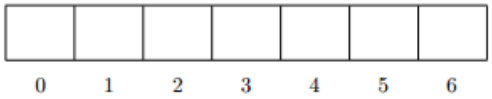
Quadratisches Sondieren

1 P a) Fügen Sie die Schlüssel 4, 16, 20, 6, 12, 9, 5 in dieser Reihenfolge in die untenstehende Hashtabelle ein. Benutzen Sie offenes Hashing mit der Hashfunktion $h(k) = k \bmod 11$. Lösen Sie Kollisionen mittels quadratischem Sondieren auf. Im Falle einer Kollision soll die Sondierung zunächst nach links und erst danach nach rechts erfolgen.



HS15

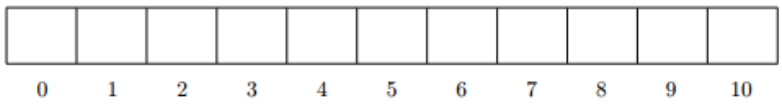
1 P b) Fügen Sie die Schlüssel 8, 10, 15, 9, 17 in dieser Reihenfolge in die untenstehende Hashtabelle ein. Benutzen Sie offenes Hashing mit der Hashfunktion $h(k) = k \bmod 7$ und lösen Sie Kollisionen mittels quadratischem Sondieren auf.



HS14

Double Hashing

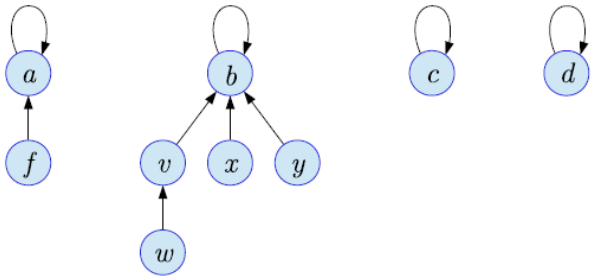
1 P (f) Fügen Sie die Schlüssel 12, 19, 6, 15, 13, 2, 28 in dieser Reihenfolge in die untenstehende Hashtabelle ein. Benutzen Sie Double Hashing mit der Hashfunktion $h(k) = k \bmod 11$, und benutzen Sie $h'(k) = 1 + (k \bmod 9)$ zur Sondierung.



HS13

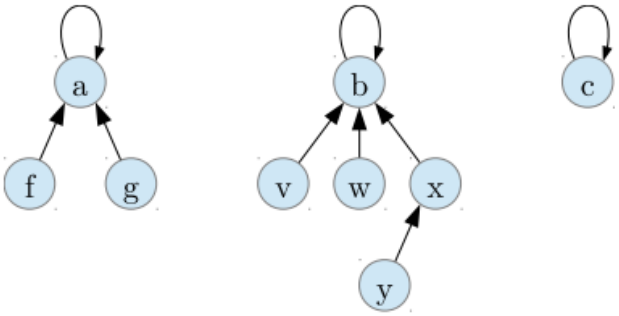
UnionFind

1 P b) Führen Sie auf der folgenden Union-Find-Datenstruktur zunächst UNION(a, c) und danach UNION(FIND$(f), b)$ aus. Benutzen Sie das Verfahren "Vereinigung nach Höhe", und zeichnen Sie die nach diesen zwei Operationen resultierende Datenstruktur.



HS15

1 P (d) Führen Sie auf der folgenden Union-Find-Datenstruktur zuächst UNION(a, c) und danach UNION(FIND$(f), b)$ aus. Benutzen Sie das Verfahren "Vereinigung nach Höhe", und zeichnen Sie die nach diesen zwei Operationen resultierende Datenstruktur.



FS13

Suchbäume

Minimale/maximale Höhe

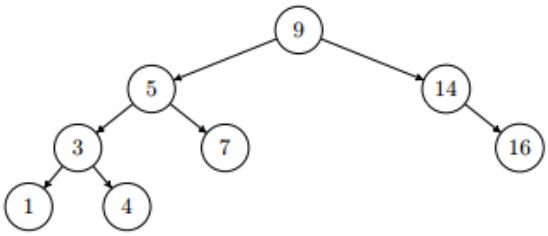
1 P c) Gegeben sei die Schlüsselmenge $\mathcal{K} = \{5, 9, 8, 11, 15, 7, 20\}$. Zeichnen Sie die beiden binären Suchbäume, die genau die Schlüssel aus $\mathcal{K}$ verwalten und die unter allen möglichen Suchbäumen minimale bzw. maximale Höhe haben.

Baum minimaler Höhe:	Baum maximaler Höhe:

HS15

AVL-Tree

1 P d) Fügen Sie in den folgenden AVL-Baum *zuerst* den Schlüssel 2 ein und entfernen Sie *danach* den Schlüssel 14.



Nach Einfügen von 2:	Nach Löschen von 14:

HS14

1 P (b) Zeichnen Sie einen AVL-Baum mit 7 Knoten, bei dem *jeder* innere Knoten einen Balancierungsfaktor ungleich 0 besitzt.

FS13

Preorder/Postorder

1 P f) Zeichnen Sie denjenigen binären Suchbaum, bei dem in Postorder-Reihenfolge die Schlüssel 4, 7, 6, 10, 11, 9 angetroffen werden.

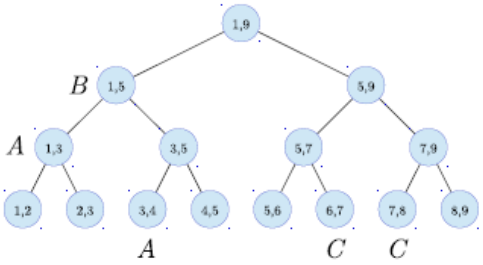
HS14

1 P (f) Zeichnen Sie den binären Suchbaum zur Schlüsselmenge $\{1, \ldots, 8\}$, dessen Preorder-Reihenfolge mit $3, 2, 1, 5, 4$ beginnt, und dessen Postorder-Reihenfolge mit $7, 8, 6, 5, 3$ endet.

FS13

Segmentbaum

1 P k) Gegeben sei der folgende Segmentbaum, der die Intervalle A, B und C speichert. Zeichnen Sie den entstehenden Baum, wenn das Intervall $D = [1, 8]$ eingefügt wird. Markieren Sie ausserdem alle Knoten, die von einer Aufspiessanfrage für $x = 3.7$ besucht werden.



HS15

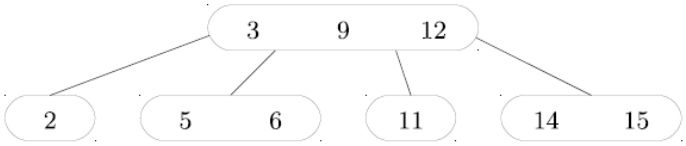
1 P (e) Gegeben sei ein Segmentbaum zur Verwaltung ganzzahliger Intervalle mit Intervallgrenzen aus $\{1, \ldots, n+1\}$ für eine Zweierpotenz $n = 2^k$, $k \in \mathbb{N}$. Geben Sie die Anzahl Knoten an, die eine Aufspiessanfrage für einen Punkt $i \in \{1, \ldots, n+1\}$ maximal besucht.

Maximale Anzahl: _____

HS13

B-Baum

1 P d) Fügen Sie in den untenstehenden 2-3-4-Baum (B-Baum der Ordnung 4) zuerst den Schlüssel 7 und in den entstehenden Baum den Schlüssel 8 ein. Führen Sie auch die zugehörigen Strukturänderungen durch.



Nach Einfügen von 7:	Nach Einfügen von 8:

HS15

Sortieralgorithmen

Quicksort

1 P e) Führen Sie auf dem folgenden Array einen Aufteilungsschritt (in-situ, d.h. ohne Hilfsarray) des Sortieralgorithmus *Quicksort* durch. Benutzen Sie als Pivot das am rechten Ende stehende Element im Array.

11	5	9	6	1	8	3	4	2	12	7
1	2	3	4	5	6	7	8	9	10	11

1	2	3	4	5	6	7	8	9	10	11

HS15

1 P (a) Führen Sie auf dem folgenden Array einen Aufteilungsschritt (in-situ, d.h. ohne Hilfsarray) des Sortieralgorithmus *Quicksort* durch. Benutzen Sie als Pivot das am rechten Ende stehende Element im Array.

9	5	19	11	7	15	1	0	2	17	4	8
1	2	3	4	5	6	7	8	9	10	11	12

1	2	3	4	5	6	7	8	9	10	11	12

FS13

Sortieren nach Auswahl

1 P a) Führen Sie auf dem folgenden Array zwei Iterationen des Sortieralgorithmus *Sortieren durch Auswahl* aus. Das zu sortierende Array ist durch vorherige Iterationen bereits bis zum Doppelstrich sortiert worden.

1	2	4	6	11	9	20	7	15	12	14	8
1	2	3	4	5	6	7	8	9	10	11	12

1	2	3	4	5	6	7	8	9	10	11	12

1	2	3	4	5	6	7	8	9	10	11	12

HS14

Min/maxheap

1 P (a) Im untenstehenden Array sind die Elemente eines Max-Heaps in der üblichen Form gespeichert. Wie sieht das Array aus, nachdem das Maximum entfernt wurde und die Heap-Bedingung wieder hergestellt wurde?

27	17	20	15	7	9	13	8	2	5	3	1	6
1	2	3	4	5	6	7	8	9	10	11	12	13

HS13

Spannbäume

1 P (b) Gegeben sei ein zusammenhängender, ungerichteter Graph $G = (V, E)$ mit $n = |V|$ Knoten und $m = |E|$ Kanten. Wie viele Knoten und Kanten besitzt ein Spannbaum von G?

Knoten: _____ Kanten: _____

HS13

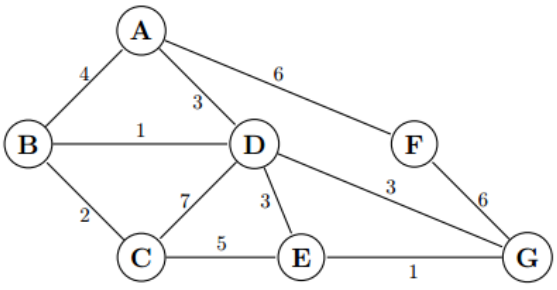
Max/complete matching

1 P (c) Zeichnen Sie einen zusammenhängenden Graphen mit 7 Knoten, der ein maximales Matching der Grösse 2 besitzt.

HS13

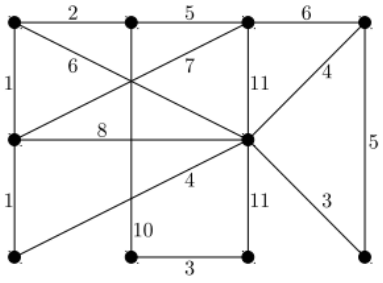
Minimaler Spannbaum

1 P e) 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.



HS14

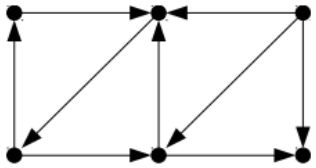
1 P (e) Markieren Sie im untenstehenden gewichteten Graphen die erste Kante, die der Algorithmus von Kruskal *nicht* in den minimalen Spannbaum aufnimmt.



FS13

Topologische Reihenfolge

1 P (g) Markieren Sie in untenstehendem Graphen $G = (V, E)$ eine *kleinstmögliche* Menge S an Kanten, sodass der Graph $G' = (V, E \backslash S)$ topologisch sortiert werden kann.



HS13

O-Notation Reihenfolge

1 P g) 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.

$$n^{3/2},\ \binom{n}{3},\ n!,\ \frac{n^2}{\log n},\ n \log n,\ 3^n,\ (\log n)^3$$

- $\binom{n}{4}$
- $\log^2(n)$
- $n \cdot \sqrt{n}$
- $n!$
- $\log(n^5)$
- 7^{13}
- $\log(n^n)$
- $\sqrt{6^n}$

HS15 HS14

- $\frac{3^n}{n^3}$
- 10^{10}
- $\log(n^n)$
- $n!$
- $n^3 + n$
- $\sqrt{3^n}$

HS13

- $\frac{n^2}{\log n}$
- $\binom{n}{2}$
- $n \log n$
- $n\sqrt{n}$
- $2^{\sqrt{n}}$
- $\log(n^2)$

FS13

Asymtotische Laufzeit (1Punkt pro Aufgabe)

i) 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 = 1; i <= n/2; i += 2)
2         for(int j = n; j >= i; j -= 1)
3                 for(int k = n; k > 2; k /= 2)
4                         ;
```

HS15

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

j) HS15

```
1 for(int i = 1; i <= n; i += 3) {
2         for(int j = n; j > 1; j = j/3)
3                 ;
4         int k = 1;
5         while(k*k <= n)
6                 k = k + 2;
7 }
```

j) HS14

```
1 for(int i = n; i > 0; i -= 1) {
2         for(int j = 0; j < i; j += 1) {
3                 ;
4         }
5 }
```

K) HS14

```
1 for(int i = 1; i <= n; i += 10) {
2         for(int j = 1; j <= n/2; j += 4)
3                 ;
4 }
```

c) HS13

```
1 for(int i = 1; i <= n; i++) {
2         for(int j = 1; j*j <= n; j++)
3                 ;
4         for(int k = n; k ≥ 2; k /= 2)
5                 ;
6 }
```

d) HS13

Rekursion

```
1 int f(int n) {
2         if(n <= 1) { return 1; }
3         else { return f(n/2)+f(n/2); }
4 }
```

e) HS13

```
1 for(int i = 1; i <= n*n; i += 10) {
2         for(int j = 1; j*j <= n; j ++)
3                 ;
4 }
```

c) FS13

```
1 for(int i = n; i > 1; i = i/2) {
2         for(int j = 1; j <= n; j ++)
3                 ;
4 }
```

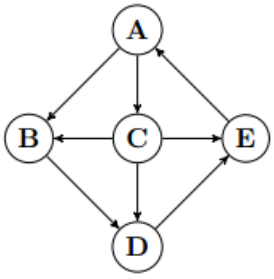
d) FS13

```
1 for(int i = n; i > 1; i = i/2) {
2         for(int j = 1; j <= n; j ++)
3                 ;
4 }
```

e) FS13

Breitensuche/Tiefensuche

1 P c) Geben Sie jeweils eine Reihenfolge an, in der die Knoten des folgenden Graphen von einer Breitensuche (BFS) bzw. Tiefensuche (DFS) mit Startknoten A besucht werden, wenn die Nachbarn eines Knotens in alphabetischer Reihenfolge abgearbeitet werden.



BFS: _____

DFS: _____

HS14

Amortisierte Kosten

4 P (e) Wir betrachten einen Dezimalzähler, der mit 0 initialisiert wird und nur eine einzige Operation ERHÖHE unterstützt, die den Wert des Zählers um 1 erhöht. Die Kosten für diese Operation entsprechen genau der Anzahl der Stellen, die verändert werden. Hat der Zähler z.B. den Wert 18 und wird ERHÖHE aufgerufen, dann ist der neue Wert des Zählers 19, und die Kosten für die Operation betragen 1 (nur die 8 wurde verändert). Hat der Zähler den Wert 19999 und wird ERHÖHE aufgerufen, dann ist der Wert des Zählers 20000, und die Operation hatte Kosten 5.

Zeigen Sie nun mittels amortisierter Analyse, dass die Kosten der Operation ERHÖHE amortisiert $\mathcal{O}(1)$ sind. Definieren Sie dazu eine geeignete Kontostandsfunktion Φ_i, und geben Sie jeweils die realen und die amortisierten Kosten an, wenn die letzten $k \geq 0$ Stellen des Zählers den Wert 9 haben.

FS13

Wahr/Falsch

2 P f) Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind. Jede korrekte Antwort gibt 0,5 Punkte, für jede falsche Antwort werden 0,5 Punkte abgezogen. Eine fehlende Antwort gibt 0 Punkte. Insgesamt gibt die Aufgabe mindestens 0 Punkte. Sie müssen Ihre Antworten nicht begründen.

Mergesort kann als stabiles Sortierverfahren implementiert werden.	☐ WAHR	☐ FALSCH
In einem AVL-Baum dürfen sich die Anzahlen der Knoten im linken und im rechten Teilbaum maximal um 1 unterscheiden.	☐ WAHR	☐ FALSCH
In einem Splaybaum mit n Schlüsseln dauert die Suche nach einem Schlüssel im schlimmsten Fall Zeit $\mathcal{O}(\log n)$.	☐ WAHR	☐ FALSCH
Sei $G = (V, E)$ ein gewichteter Graph. Wenn der minimale Spannbaum von G eindeutig bestimmt ist, dann hat G keine zwei Kanten mit dem gleichen Gewicht.	☐ WAHR	☐ FALSCH

HS15

3 P (d) Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind. Jede korrekte Antwort gibt 0,5 Punkte, für jede falsche Antwort werden 0,5 Punkte abgezogen. Eine fehlende Antwort gibt 0 Punkte. Insgesamt gibt die Aufgabe mindestens 0 Punkte. Sie müssen Ihre Antworten nicht begründen.

Eine Postorder-Traversierung eines binären Suchbaums erzeugt eine absteigend sortierte Liste der gespeicherten Schlüssel.	☐ WAHR	☐ FALSCH
Es gibt einen AVL-Baum, bei dem mehr als die Hälfte seiner inneren Knoten nicht perfekt balanciert sind (d.h., einen Balancierungsfaktor ungleich 0 haben).	☐ WAHR	☐ FALSCH
Zu jedem AVL-Baum gibt es eine Einfügereihenfolge, die zu genau diesem Baum führt, ohne dass Rotationen stattfinden.	☐ WAHR	☐ FALSCH
Das Einfügen eines neuen Schlüssels in einen AVL-Baum führt selbst im schlimmsten Fall zu nur einer (einfachen oder Doppel-)Rotation.	☐ WAHR	☐ FALSCH
Das Einfügen eines neuen Elements in einen Fibonacci-Heap erfordert im schlimmsten Fall nur konstante Zeit.	☐ WAHR	☐ FALSCH
Sortieren durch Einfügen kann als stabiles Sortierverfahren implementiert werden.	☐ WAHR	☐ FALSCH

HS13

3 P (c) Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind. Jede korrekte Antwort gibt 0,5 Punkte, für jede falsche Antwort werden 0,5 Punkte abgezogen. Eine fehlende Antwort gibt 0 Punkte. Insgesamt gibt die Aufgabe mindestens 0 Punkte. Sie müssen Ihre Antworten nicht begründen.

Eine Inorder-Traversierung eines binären Suchbaums erzeugt eine sortierte Liste der gespeicherten Schlüssel.	☐ WAHR	☐ FALSCH
Hat eine Folge von m Operationen im schlimmsten Fall Gesamtkosten $\mathcal{O}(m)$, dann hat jede einzelne dieser Operationen im schlimmsten Fall Kosten $\mathcal{O}(1)$.	☐ WAHR	☐ FALSCH
Sei $G = (V, E)$ ein Graph. Jeder Teilgraph von G mit $\lvert V\rvert - 1$ Kanten ist ein Spannbaum von G.	☐ WAHR	☐ FALSCH
In einem Fibonacci-Heap mit n Schlüsseln ist die Laufzeit zur Extraktion des Minimums im schlimmsten Fall $\mathcal{O}(\log n)$.	☐ WAHR	☐ FALSCH
Sortieren durch Auswahl ist ein stabiles Sortierverfahren.	☐ WAHR	☐ FALSCH
Auf einem komplett vorsortierten Array ist die Laufzeit von Sortieren durch Auswahl linear.	☐ WAHR	☐ FALSCH

FS13

Rekursionsgleichung/Induktionsbeweis

3 P h) Gegeben ist die folgende Rekursionsgleichung:

$$T(n) := \begin{cases} T(n/5) + 4n + 1 & n > 1 \\ 5 & n = 1 \end{cases}$$

Geben Sie eine geschlossene (d.h. nicht-rekursive) und *möglichst einfache* Formel für $T(n)$ an und beweisen Sie diese mit vollständiger Induktion.

HS15

3 P i) Gegeben ist die folgende Rekursionsgleichung:

$$T(n) := \begin{cases} 15 + 4T(n/4) & n > 1 \\ 1 & n = 1 \end{cases}$$

Geben Sie eine geschlossene (d.h. nicht-rekursive) und *möglichst einfache* Formel für $T(n)$ an und beweisen Sie diese mit vollständiger Induktion.

Hinweise:

(1) Sie können annehmen, dass n eine Potenz von 4 ist.

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

HS14

4 P l) Ein vollständiger ternärer Suchbaum ist ein Suchbaum, bei dem jeder innere Knoten genau drei Nachfolger besitzt, und in dem alle Blätter die gleiche Tiefe h besitzen (die Wurzel hat nach Definition Tiefe 0). Leiten Sie eine rekursive Formel in Abhängigkeit von h für die Anzahl der Blätter in einem vollständigen ternären Baum her, und begründen Sie Ihre Herleitung. Lösen Sie danach die Rekursion auf und beweisen Sie die Korrektheit ihrer Auflösung durch vollständige Induktion über h.

HS15

4 P (b) Gegeben ist die folgende Rekursionsgleichung:

$$T(n) := \begin{cases} 5T(n/5) + n + 4 & n > 1 \\ 1 & n = 1 \end{cases}$$

Geben Sie eine geschlossene (d.h. nicht-rekursive) und *möglichst einfache* Formel für $T(n)$ an und beweisen Sie diese mit vollständiger Induktion.

Hinweise:

(1) Sie können annehmen, dass n eine Potenz von 5 ist.

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

HS13

3 P (b) Gegeben ist die folgende Rekursionsgleichung:

$$T(n) := \begin{cases} 6 + 7T(n/3) & n > 1 \\ 6 & n = 1 \end{cases}$$

Geben Sie eine geschlossene (d.h. nicht-rekursive) und *möglichst einfache* Formel für $T(n)$ an und beweisen Sie diese mit vollständiger Induktion.

Hinweise:

(1) Sie können annehmen, dass n eine Potenz von 3 ist.

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

FS13

Dynamische Programmierung

Aufgabe 2.

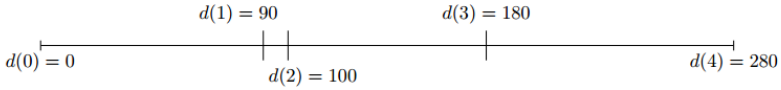
Motivation. Im Rahmen eines Infrastrukturprojekts sollen entlang einer Autobahn Schnellladestationen für Elektroautos gebaut werden. Bei der Planung wird davon ausgegangen, dass ein Elektroauto mit einer vollen Batterieladung 100 km zurücklegen kann. Es werden n mögliche Standorte definiert, aus denen eine beliebig große Teilmenge von Standorten für den Bau der Ladestationen ausgewählt werden sollen. Aus Kostengründen sollen aber nicht zu viele Stationen gebaut werden. Daher sollen die Stationen so gebaut werden, dass die Distanz zwischen zwei aufeinanderfolgenden Ladestationen möglichst nahe an 100 km, aber niemals darüber liegt.

Problemdefinition. Gegeben sind n mögliche Standorte, wobei $d(i)$ die Entfernung des i-ten Standortes vom Ausgangspunkt der Strecke ist. Weiters sei $d(0) = 0$, und $d(n+1)$ gibt die Gesamtlänge der Strecke an. Für eine Distanz x zwischen zwei benachbarten Ladestationen wird eine Kostenfunktion $c(x) = (100 - x)^2$ definiert. Es soll eine Teilmenge $I = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ von Standorten ausgewählt werden, sodass ein Elektroauto höchstens 100 km bis zur nächsten Ladestation (bzw. zum Ziel) zurücklegen muss und die Gesamtkosten aller Teilstrecken

$$\sum_{j=0}^{k} c\left(d(i_{j+1}) - d(i_j)\right)$$

minimiert wird, wobei $i_0 = 0$ und $i_{k+1} = n + 1$ seien. Beachten Sie, dass k nicht Teil der Eingabe ist und eine optimale Teilmenge I mit *beliebiger* Größe gesucht wird.

Beispiel: Es stehen $n = 3$ Standorte zur Auswahl, wobei $d(1) = 90, d(2) = 100$ und $d(3) = 180$ sind. Die Länge der gesamten Strecke ist $d(4) = 280$.



Während die Auswahl $I = \{2, 3\}$ zu Gesamtkosten 400 führt, ist die optimale Auswahl $I^* = \{1, 3\}$ mit Gesamtkosten 200.

7 P a) Geben Sie einen Algorithmus an, der nach dem Prinzip der dynamischen Programmierung arbeitet und die minimalen Kosten einer Auswahl von Standorten berechnet. Gehen Sie in Ihrer Lösung 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 können aus der DP-Tabelle die minimalen Kosten einer Auswahl von Standorten ausgelesen werden?

Hinweis: Der triviale Algorithmus, der einfach alle möglichen Lösungen inspiziert, gibt **keine** Punkte, weil er nicht nach dem Prinzip der dynamischen Programmierung arbeitet.

2 P b) Beschreiben Sie detailliert, wie aus der DP-Tabelle abgelesen werden kann, an welchen Standorten die Schnellladestationen gebaut werden können, um die minimalen Kosten zu erreichen.

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

HS15

Eine Firma hat den Auftrag bekommen, ein Kabel der Länge mindestens L zu produzieren. Da es im Lager viele bereits früher produzierte Kabelstücke gibt, soll das gewünschte Kabel aus diesen Teilstücken zusammengesetzt werden. Konkret gibt es im Lager n Kabelstücke. Kabelstück i hat Länge l_i. Werden zwei Kabelstücke mit Längen l_i und l_j zusammengesetzt, entsteht ein Kabel der Länge $l_i + l_j$. Um das entstehende Kabel nicht unnötig lang zu machen, soll es unter allen Möglichkeiten, ein Kabel der Länge $\geq L$ herzustellen, minimale Länge haben. Sie dürfen annehmen, dass der Lagerbestand ausreichend gross ist, d.h., dass die Summe der Längen aller Kabelstücke mindestens L beträgt.

Beispiel: Es soll ein Kabel der Länge $L = 6$ produziert werden, und im Lager gibt es Kabelstücke der Längen $l_1 = 3$, $l_2 = 4$ und $l_3 = 5$. Die beste Möglichkeit ist die Wahl der Kabelstücke $\{1, 2\}$, denn diese haben zusammen Länge 7. Auch die Wahlen $\{2, 3\}$ und $\{1, 2, 3\}$ führen zu einem Kabel der Länge ≥ 6, aber da ihre Gesamtlängen 9 bzw. 12 betragen, sind sie nicht optimal und daher auch nicht die gesuchte Lösung.

9 P a) Geben Sie einen Algorithmus an, der nach dem Prinzip der dynamischen Programmierung arbeitet und die minimale Länge eines Kabels berechnet, sodass dieses Kabel aus geeigneten Kabelstücken aus $\{1, \ldots, n\}$ zusammengesetzt werden kann und mindestens Länge L hat. Für das Beispiel oben soll also 7 ausgegeben werden. Gehen Sie in Ihrer Lösung 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 der Wert der minimalen Kabellänge ausgelesen werden?

Hinweis: Der triviale Algorithmus, der einfach alle möglichen Lösungen inspiziert, gibt **keine** Punkte, weil er nicht nach dem Prinzip der dynamischen Programmierung arbeitet.

2 P b) Beschreiben Sie detailliert, wie aus der DP-Tabelle abgelesen werden kann, welches Kabelstück in einer optimalen Lösung benutzt wird.

2 P c) Geben Sie die Laufzeit des in a) und b) entwickelten Verfahrens an und begründen Sie Ihre Antwort. Ist die Laufzeit polynomiell?

Aufgabe 5. In der Schweiz existieren die folgenden Münzwerte: 5 Rappen, 10 Rappen, 20 Rappen, 50 Rappen, 1 Franken, 2 Franken, 5 Franken. Damit lassen sich alle Geldbeträge darstellen, deren Wert in Rappen durch 5 teilbar ist. In dieser Aufgabe befassen wir uns mit der Anzahl verschiedener Möglichkeiten, einen gegebenen Betrag durch irgendeine Menge von Münzen zu erreichen. Beispielsweise kann der Betrag von 20 Rappen auf vier verschiedene Arten erreicht werden:

1) $5 + 5 + 5 + 5$ Rappen,
2) $5 + 5 + 10$ Rappen,
3) $10 + 10$ Rappen,
4) 20 Rappen.

Beachten Sie, dass es dabei auf die Reihenfolge der Münzen nicht ankommt; $5 + 5 + 10$ Rappen ist also dieselbe Menge von Münzen wie $5 + 10 + 5$ Rappen. Für diese Aufgabe gehen wir von der allgemeinen Annahme aus, dass n Münzen mit den Werten $M_1, \ldots, M_n \in \mathbb{N}$ (in der gleichen Einheit, z.B. Rappen) gegeben sind. O.B.d.A. seien diese aufsteigend geordnet und paarweise verschieden, d.h. $M_1 < M_2 < \cdots < M_n$. Für den Schweizer Franken z.B. hätten wir die Münzwerte $M_1 = 5$, $M_2 = 10$, $M_3 = 20$, $M_4 = 50$, $M_5 = 100$, $M_6 = 200$, $M_7 = 500$ (jeweils in Rappen ausgedrückt).

Hinweis: Die Münzwerte M_i sind nicht notwendigerweise Vielfache von 5 wie im obigen Beispiel, sondern beliebige Zahlen. Es könnten sogar alle M_i Primzahlen (d.h., paarweise teilerfremd) sein.

8 P a) Gegeben sei ein ganzzahliger Betrag $B \in \mathbb{N}$ in der gleichen Einheit wie die Münzen $M_1, \ldots, M_n$. Geben Sie einen möglichst effizienten Algorithmus an, der nach dem Prinzip der dynamischen Programmierung arbeitet und die Anzahl der Möglichkeiten, den Betrag B mit den Münzen $M_1, \ldots, M_n$ darzustellen, berechnet. Gehen Sie in Ihrer Lösung 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 müssen die Einträge berechnet werden?
4) Wie lässt sich die Lösung aus der DP-Tabelle extrahieren?

2 P b) Geben Sie die Laufzeit der Lösung aus Aufgabenteil a) an und begründen Sie Ihre Antwort. Ist die Laufzeit polynomiell?

3 P c) Beschreiben Sie detailliert, wie durch Rückverfolgung in der Lösungstabelle gemäss b) *eine* mögliche Darstellung des Betrags B gefunden werden kann, sofern überhaupt eine existiert. Geben Sie auch die Laufzeit dieses Algorithmus an.

HS13

Aufgabe 3.

Ein *Palindrom* ist eine Zeichenkette, die sich von vorne wie von hinten gleich liest, also z.B. das Wort `RENTNER`. Formal ist ein Palindrom eine Zeichenkette $\langle a_1, \ldots, a_n \rangle$, wobei entweder $n = 1$ gilt, oder aber es sind $a_1 = a_n$ und $\langle a_2, \ldots, a_{n-1} \rangle$ ein Palindrom. Ein Array $A[1 \ldots n]$ speichere eine Zeichenkette der Länge n. Ein Teilarray $A[i \ldots j]$, $1 \leq i \leq j \leq n$, heisst *Palindrom in* A, falls $\langle A[i], \ldots, A[j] \rangle$ ein Palindrom ist.

Beispiel: Das Array [`L`, `A`, `R`, `A`] enthält die Palindrome `A`, `R`, `L` sowie `ARA` (das Palindrom `A` kommt doppelt vor). Das Array [`A`, `N`, `N`, `A`] enthält die Palindrome `A`, `N`, `NN` sowie `ANNA` (die Palindrome `A` und `N` kommen doppelt vor).

9 P (a) Sei A ein Array, das eine Zeichenkette der Länge n speichert. Entwerfen Sie einen Algorithmus nach dem Prinzip der *dynamischen Programmierung*, der alle Paare (i, j) ausgibt, für die $\langle A[i], \ldots, A[j] \rangle$ ein Palindrom ist. Gehen Sie in Ihrer Lösung 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? Unterscheiden Sie bei der Beschreibung Palindrome der Längen 1, 2 sowie längere Palindrome.
3) In welcher Reihenfolge müssen die Einträge berechnet werden?
4) Wie lässt sich die Lösung aus der DP-Tabelle extrahieren?

Beispiel: Für die Eingabe [`L`, `A`, `R`, `A`] werden die Paare $(1, 1)$, $(2, 2)$, $(3, 3)$, $(4, 4)$, $(2, 4)$ ausgegeben. Für die Eingabe [`A`, `N`, `N`, `A`] werden die Paare $(1, 1)$, $(2, 2)$, $(3, 3)$, $(4, 4)$, $(2, 3)$, $(1, 4)$ ausgegeben. Es wird keine spezielle Reihenfolge gefordert, in der die Paare ausgegeben werden müssen.

Beachten Sie, dass die Aufgabe ohne dynamische Programmierung durch triviale Aufzählung aller Palindrome in Zeit $\mathcal{O}(n^3)$ lösbar ist. Gesucht wird hier ein effizienterer Algorithmus.

1 P (b) Geben Sie die Laufzeit Ihrer Lösung an.

3 P (c) Angenommen, der Algorithmus aus (a) hat die DP-Tabelle bereits berechnet. Beschreiben Sie detailliert, wie Sie aus der DP-Tabelle ein längstes Palindrom in A ablesen können. Geben Sie auch die maximal benötigte Laufzeit an.

FS13

Kürzester Weg/Flussprobleme

Aufgabe 3.

Motivation. Sie möchten mit dem Zug von Zürich nach Hamburg fahren und haben mehrere Routen zur Auswahl. Sie haben bereits die Kosten für jede mögliche Teilstrecke ermittelt und suchen nach der kostengünstigsten Route.

Problemdefinition. Gegeben sei eine Menge von Stationen $V = \{s, t, v_1, \ldots, v_n\}$, wobei s Ihr Ausgangspunkt und t Ihr Ziel ist. Die übrigen Stationen v_i bezeichnen alle möglichen Zwischenstopps. Weiters sei eine Menge E von gerichteten Teilstrecken gegeben, wobei genau dann $(v, w) \in E$, wenn es eine Teilstrecke von v zu w gibt. Zudem sind für jede Teilstrecke $(v, w) \in E$ die Kosten $c(v, w) > 0$ gegeben. Sie suchen eine Route von s nach t mit möglichst geringen Gesamtkosten, d.h. einer möglichst geringen Summe der Kosten aller Teilstrecken der Route.

2 P a) Nennen Sie einen möglichst effizienten Algorithmus, der das oben genannten Problem löst. Welche Datenstruktur muss bei der Implementierung verwendet werden, um eine effiziente Laufzeit zu erreichen? Geben Sie die Laufzeit in Abhängigkeit von der Anzahl der Stationen $|V|$ und Teilstrecken $|E|$ an.

Sie haben einen Gutschein im Wert von 30 Franken, den Sie für eine Teilstrecke verwenden können, deren Kosten mindestens 50 Franken betragen. Sie können den Gutschein natürlich nur einmal verwenden.

4 P b) Konstruieren Sie einen gerichteten gewichteten Graphen $G = (V', E', c')$, sodass ein kürzester Weg von s nach t in G der billigsten Route entspricht, die Sie mit dem Gutschein erreichen können. Der Gutschein muss nicht zwingend verwendet werden, denn es könnte eine kostengünstigere Route geben, bei welcher der Gutschein nicht eingesetzt werden kann. Der Graph G soll so konstruiert werden, dass der kürzeste Weg in G in derselben asymptotischen Laufzeit wie in Teilaufgabe a) berechnet werden kann.

Das Zugunternehmen möchte die Fahrkarten aller Reisenden kontrollieren. Dazu sollen Kontrolleure auf verschiedenen Teilstrecken die Fahrkarten überprüfen. Es soll sichergestellt werden, dass alle Reisenden unabhängig von der Wahl der Route von s nach t kontrolliert werden. An wie vielen Teilstrecken müssen mindestens Kontrollen durchgeführt werden, damit jede mögliche Route mindestens eine Teilstrecke benutzt, an der kontrolliert wird?

3 P c) Modellieren Sie das o.g. 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. Nennen Sie einen möglichst effizienten Algorithmus zur Berechnung des maximalen Flusses von s nach t in N. Wie kann aus dem Wert eines maximalen Flusses abgelesen werden, an wie vielen Teilstrecken mindestens kontrolliert werden muss?

HS15

2 P g) Gegeben sei ein gewichteter Graph G. Zeigen oder widerlegen Sie die folgenden Aussagen:

1. Wenn $P = \langle v_1, \ldots, v_k \rangle$ und $Q = \langle w_1, \ldots, w_l \rangle$ kürzeste Pfade mit $v_k = w_1$ sind, dann ist $\langle v_1, \ldots, v_k = w_1, \ldots, w_l \rangle$ ein kürzester Pfad von v_1 nach w_l.
2. Sei P ein kürzester Pfad von u nach v und sei w ein innerer Knoten von P. Der Teilpfad von P von u nach w ist ein kürzester Pfad von u nach w in G. Ebenso ist der Teilpfad von w nach v ein kürzester Pfad in G.

HS14

Aufgabe 3.

In einem Krankenhaus soll bestimmt werden, welcher Arzt an welchem Tag arbeitet. Diese Aufgabe befasst sich mit der Arbeitszeitplanung zu Ferientagen. Sei T die Menge aller Ferientage. Es gibt k Ferienzeiten $\{1, \ldots, k\}$, die aus einem oder mehreren zusammenhängenden Tagen $T_j \subseteq T$ bestehen (für $j \in \{1, \ldots, k\}$). Jeder Tag in T kommt in genau einer Ferienzeit vor, d.h. insbesondere ist $T_i \cap T_j = \emptyset$ für $i \neq j$. Im Krankenhaus arbeiten n Ärzte. Jeder Arzt i gibt eine Menge von Tagen $S_i \subseteq T$ an, an denen er anwesend sein kann. Die Aufgabe besteht nun darin, einen Einsatzplan zu entwerfen, der die Ferientage T so unter den Ärzten verteilt, dass an jedem Tag genau ein Arzt anwesend ist. Ausserdem soll kein Arzt an mehr als $C \in \mathbb{N}$ Ferientagen insgesamt arbeiten müssen. Die Aufgabe besteht nun darin, effizient zu entscheiden, ob für ein gegebenes C ein geeigneter Einsatzplan existiert, und falls ja, effient zu berechnen, wie dieser aussieht.

Beispiel: Sei $C = 2$ gegeben. Die Ärzte und Ferienzeiten seien die folgenden:

i	Arzt	Mögliche Tage S_i
1	Dr. Cuddy	$\{24.12, 25.12, 26.12\}$
2	Dr. Foreman	$\{24.12\}$
3	Dr. House	$\{1.8\}$
4	Dr. Wilson	$\{1.8, 24.12\}$

j	Beschreibung	Zugehörige Tage T_j
1	Weihnachten	$\{24.12, 25.12, 26.12\}$
2	Nationalfeiertag	$\{1.8\}$

Nun existieren verschiedene Möglichkeiten, z.B. könnte Dr. Cuddy am 25. und am 26.12 anwesend sein, Dr. House am 1.8 und Dr. Wilson am 24.12. Wäre stattdessen $C = 1$ gegeben gewesen, dann existierte kein geeigneter Einsatzplan.

4 P a) Modellieren Sie das o.g. 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 die Kanten besitzen sollen. Wie kann aus dem Wert eines maximalen Flusses abgelesen werden, ob ein geeigneter Einsatzplan existiert oder nicht?

2 P b) Nennen Sie einen Algorithmus, der das Problem aus a) möglichst effizient löst, und geben Sie die Laufzeit im schlimmsten Fall in Abhängigkeit von der Anzahl der Ärzte n und der Anzahl der Ferientage $m = |T|$ an. Begründen Sie Ihre Antwort.

3 P c) Nehmen Sie an, das Flussproblem aus a) wurde bereits gelöst, d.h. zu einem maximalen Fluss ϕ kennen Sie den Fluss ϕ_e auf jeder Kante e. Nehmen Sie weiter an, ein geeigneter Einsatzplan wie oben beschrieben existierte tatsächlich. Beschreiben Sie detailliert einen Algorithmus, der aus den ϕ_e einen solchen Einsatzplan berechnet. Welche Laufzeit hat Ihr Verfahren, wenn auf jedes ϕ_e in Zeit $\Theta(1)$ zugegriffen werden kann?

2 P d) Das bisherige Verfahren hat den Nachteil, dass es u.U. Einsatzpläne generiert, bei denen manche Ärzte sehr viel und andere Ärzte sehr wenig eingesetzt werden. Wir wollen daher einen Einsatzplan finden, der die o.g. Bedingungen erfüllt und zusätzlich jedem Arzt für jedes $j \in \{1, \ldots, k\}$ maximal einen Tag in T_j zuweist. Für das obige Beispiel heisst dies, dass ein Arzt zu Weihnachten entweder am 24.12, am 25.12, am 26.12 oder überhaupt keinen Dienst hat. Beschreiben Sie, wie das in a) konstruierte Netz modifiziert werden muss, um diese zusätzliche Anforderung zu erfüllen.

HS14

Aufgabe 4. Ein Versandhandel bietet die Artikeltypen $\{1, \ldots, n\}$ an. Wir bestellen von jedem Artikeltyp einige Exemplare, und beauftragen für den Transport $m \leq n$ Kuriere. Allerdings transportiert nicht jeder Kurier jeden Artikeltypen (z.B. kann ein Fahrradkurier keinen Fernseher ausliefern). Für jeden Kurier $j \in \{1, \ldots, m\}$ sei $T(j) \subseteq \{1, \ldots, n\}$ die Menge der Artikeltypen, die er transportieren kann. Wir möchten entscheiden, ob die Artikel(exemplare) so auf die Kuriere aufgeteilt werden können, dass alle Artikel befördert werden und jeder Kurier nur einmal fahren muss.

4 P a) Angenommen, von jedem Artikeltyp $i \in \{1, \ldots, n\}$ wird genau ein Exemplar bestellt, und jeder Kurier $j \in \{1, \ldots, m\}$ soll pro Fahrt nur ein einziges Artikelexemplar transportieren. Modellieren Sie das o.g. Problem als Flussproblem. Beschreiben Sie dazu die Konstruktion eines geeigneten Netzwerks $G = (V, E, c)$ mit der Knotenmenge V sowie der Kantenmenge E, und welche Kapazitäten den Kanten zugewiesen werden. Wie kann aus dem Wert eines maximalen Flusses abgelesen werden, ob eine geeignete Verteilung existiert oder nicht?

3 P b) Nun wird von jedem Artikeltyp $i \in \{1, \ldots, n\}$ nicht nur ein, sondern $f(i) \in \mathbb{N}_0$ Exemplare bestellt, und jeder Kurier $j \in \{1, \ldots, m\}$ kann pro Fahrt nicht nur einen, sondern bis zu $k(j)$ Artikelexemplare insgesamt befördern. Beschreiben Sie, wie Sie die Lösung aus a) modifizieren können, um dieses allgemeinere Problem zu lösen. Wie gross muss nun der Wert eines maximalen Flusses sein, damit jeder Kurier nur einmal fahren muss?

Beispiel:

i	Artikeltyp	Bestellte Exemplare $f(i)$
1	Tisch	2
2	Stuhl	2
3	Stift	5
4	Drucker	0

Kurier j	Max. Anz. Artikel $k(j)$ pro Fahrt	Akzeptierte Artikeltypen $T(l)$
1	4	$\{3\}$ (Stift)
2	3	$\{1, 2, 3, 4\}$ (Alles)
3	2	$\{1, 2\}$ (Tisch, Stuhl)

Hier reicht eine Fahrt pro Kurier, z.B. indem 4 Stifte von Kurier 1 transportiert werden, 1 Stift sowie 2 Tische von Kurier 2, und die 2 Stühle von Kurier 3.

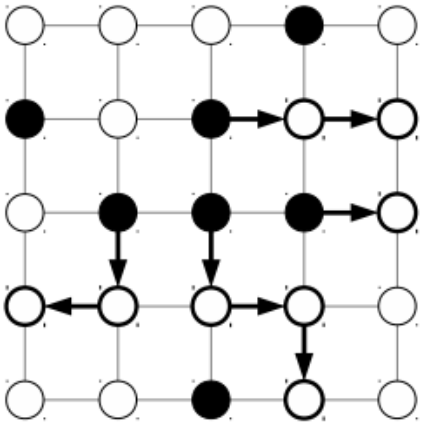
2 P c) Nennen Sie einen Algorithmus, der das Problem aus b) möglichst effizient löst, und geben Sie die Laufzeit im schlimmsten Fall in Abhängigkeit von der Anzahl der Artikeltypen n und der Anzahl der Kuriere m an. Begründen Sie Ihre Antwort.

3 P d) Angenommen, das Flussproblem aus b) wurde bereits gelöst, d.h. zu einem maximalen Fluss ϕ kennen Sie den Fluss ϕ_e auf jeder Kante e, und angenommen, eine geeignete Verteilung wie oben beschrieben existiert tatsächlich. Beschreiben Sie detailliert einen Algorithmus, der aus den ϕ_e eine solche Verteilung berechnet. Konkret soll eine Menge M berechnet werden; diese enthält ein Tripel (j, i, k) genau dann, wenn der Kurier j genau k Exemplare vom Artikel i transportiert. Für das obige Beispiel ist $M = \{(1, 3, 4), (2, 1, 2), (2, 3, 1), (3, 2, 2)\}$. Welche Laufzeit hat Ihr Verfahren, wenn auf jedes ϕ_e in Zeit $\Theta(1)$ zugegriffen werden kann?

HS13

Aufgabe 4:

Gegeben sei ein $n \times n$-Gitter, d.h. ein ungerichteter Graph mit n Zeilen und n Spalten, der Kanten zwischen je zwei horizontal sowie vertikal benachbarten Knoten enthält. Beim *Fluchtwegeproblem* sind $m \leq n^2$ Startknoten $s_1, \ldots, s_m$ gegeben. Wir möchten entscheiden, ob es m kantendisjunkte Wege von den Startpunkten s_i zu je einem Randpunkt des Gitters gibt. Beachten Sie, dass je zwei Wege zwar einen oder mehrere Knoten, aber keine gemeinsamen Kanten besitzen dürfen. Im folgenden Beispiel sind die Startknoten schwarz und die Fluchtwege fett eingezeichnet.



4 P (a) Modellieren Sie das obige Entscheidungsproblem als Flussproblem. Beachten Sie, dass wir bei Flussproblemen *gerichtete* Netzwerke betrachten, das Gitter selbst aber ungerichtet ist. Erläutern Sie daher genau, wie ein geeignetes Netzwerk $N = (V, E, c)$ konstruiert werden kann, d.h. welche Knoten V und welche Kanten E definiert werden, und welche Kapazitäten den Kanten zugewiesen werden müssen. Überlegen Sie, wie Sie aus dem Wert eines maximalen Flusses schlussfolgern können, ob m kantendisjunkte Fluchtwege existieren oder nicht.

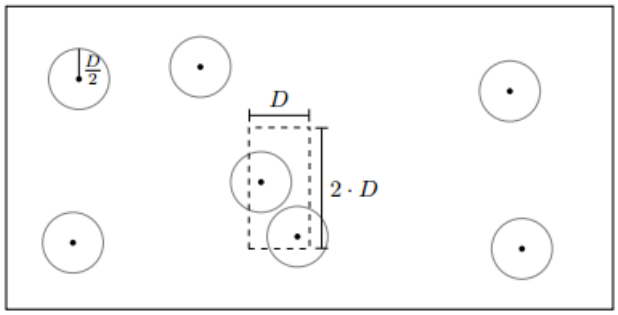
2 P (b) Nennen Sie einen Flussalgorithmus, der das Problem aus (a) möglichst effizient löst. Geben Sie die Laufzeit der Lösung aus (a) in Abhängigkeit von n und m an.

3 P (c) Wir möchten nun entscheiden, ob es eine Menge von m *knotendisjunkten* Fluchtwegen gibt, d.h. eine Menge von Fluchtwegen, von denen keine zwei einen gemeinsamen Knoten besitzen. Beschreiben Sie detailliert, wie Sie Ihre Lösung aus (a) modifizieren müssen, damit durch jeden Knoten maximal ein Fluchtweg läuft. Verändert sich die Laufzeit Ihrer Lösung, und wenn ja, wie?

Scanline

Aufgabe 4.

Eine Flugsicherungsgesellschaft hat den Auftrag, den Luftraum der Schweiz zu überwachen. Dabei soll sichergestellt werden, dass zwei Flugzeuge, die sich auf gleicher Reiseflughöhe bewegen, einen Mindestabstand D zueinander einhalten. Für jedes der n Flugzeuge, die sich auf gleicher Höhe im Luftraum befinden, wird dazu die aktuelle Position (x_i, y_i) übermittelt. Ein Alarm soll dann ausgelöst werden, wenn zwei Flugzeuge den Mindestabstand D zueinander nicht einhalten.



3 P a) Beweisen Sie, dass ein Rechteck mit Seitenlängen $2 \cdot D$ und D nicht mehr als acht Punkte enthalten kann, falls alle paarweisen Distanzen der Punkte mindestens D sind.

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

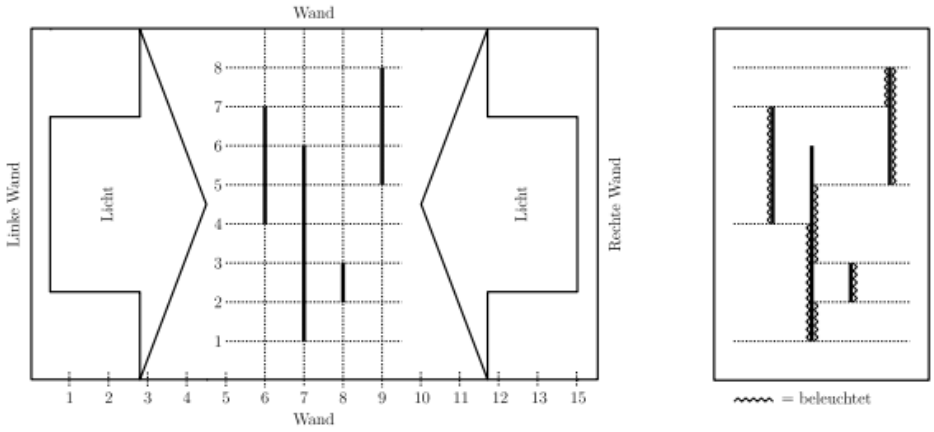
1) In welche Richtung verläuft die Scanline, und was sind die Haltepunkte?
2) Welche Objekte muss die Scanline-Datenstruktur verwalten, und was ist eine angemessene Datenstruktur?
3) Was passiert, wenn die Scanline auf einen neuen Haltepunkt trifft?
4) Welche Laufzeit in Abhängigkeit von n hat Ihr Algorithmus? Begründen Sie Ihre Antwort.

Hinweis: Der Einfachheit halber nehmen wir an, dass für zwei beliebige Flugzeuge i und j stets $x_i \neq x_j$ gilt. Beachten Sie, dass ein naiver Algorithmus alle paarweisen Distanzen der Flugzeuge in Zeit $\mathcal{O}(n^2)$ berechnet. Für Lösungen mit quadratischer Laufzeit werden daher keine Punkte in Teilaufgabe b) vergeben.

HS15

Aufgabe 4.

Ein Künstler fertigt einen Grundrissplan seines Kunstwerks an, bei dem vertikale Platten mit Laserlicht von der Seite bestrahlt werden.



Es gibt n Platten mit verschiedenen Positionen und Breiten. Platte i werde dabei durch ein Tripel $P_i = (x_i, y_i, b_i)$ repräsentiert, wobei x_i der Abstand zur linken Wand, y_i der Abstand zur unten gezeichneten Wand und b_i die Breite der Platte angeben. Eine mögliche Eingabe für die Skizze oben wäre beispielsweise $P_1 = (7, 1, 5)$, $P_2 = (9, 5, 3)$, $P_3 = (8, 2, 1)$ sowie $P_4 = (6, 4, 3)$.

Die Platten werden von flächenförmigen Laserlichtquellen (an den Wänden links und rechts) bestrahlt, welche über die gesamte Breite und Höhe des Kunstwerks horizontale Lichtstrahlen aussenden. Licht, welches auf eine Platte trifft, wird vollständig absorbiert. Weil das Licht die Platten erhitzt, muss jede bestrahlte Platte gekühlt werden, und zwar proportional zum Betrag des einfallenden Lichts. Deshalb möchte der Künstler nun *für jede Platte* die gesamte bestrahlte Fläche (als Summe der von links bestrahlten Fläche und der von rechts bestrahlten Fläche) berechnen. Da die Platten vom Fussboden bis zur Decke reichen, ist die Höhe aller Platten gleich und es genügt, anstatt der bestrahlten Fläche die bestrahlte Breite der Platten zu berechnen. Daher liegt wie in der obigen Skizze ein zweidimensionales Problem vor.

> Für jede Platte i soll die gesamte bestrahlte Breite B_i als Summe der von links bestrahlten Breite und der von rechts bestrahlten Breite berechnet werden (siehe Abbildung rechts). Für die obige Eingabe soll also $B_1 = 6$, $B_2 = 4$, $B_3 = 1$ sowie $B_4 = 3$ ausgegeben werden.

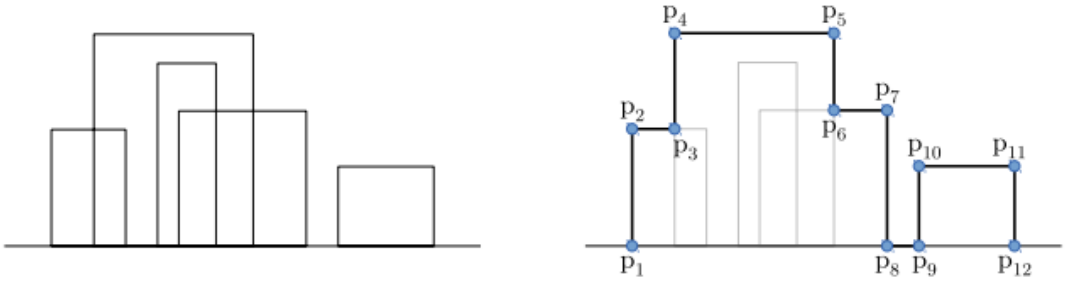
10 P Entwerfen Sie einen möglichst effizienten Scanline-Algorithmus für das obige Problem. Gehen Sie in Ihrer Lösung auf die folgenden Aspekte ein.

1) In welche Richtung verläuft 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 für jede Platte die bestrahlte Breite B_i berechnet werden?
5) Welche Laufzeit in Abhängigkeit von n hat Ihr Algorithmus? Begründen Sie Ihre Antwort.

Hinweis: Der Einfachheit halber nehmen wir an, dass sich keine zwei Platten direkt übereinander befinden oder direkt nebeneinander starten oder enden. Beachten Sie aber, dass wie im obigen Beispiel die Platten der Eingabe nicht notwendigerweise nach x-Koordinate sortiert sind.

HS14

Aufgabe 3. In dieser Aufgabe geht es um die Berechnung des Umrisses von Skylines. Die Skyline einer Stadt (der Umriss) ergibt sich wie der Schatten einer Menge rechteckiger Hochhäuser, die allesamt auf dem Boden aufsitzen und als orthogonale Recktecke wie im Bild links gegeben sind. Der Umriss ist das orthogonale Polygon, das die Vereinigung aller solchen gegebenen Rechtecke beschreibt, wie im Bild rechts zu sehen. Er kann durch die Menge seiner Eckpunkte $p_1, \ldots, p_k$ beschrieben werden.



7 P a) Entwerfen Sie einen möglichst effizienten Scanline-Algorithmus, der als Eingabe eine Menge von n orthogonalen Rechtecken erhält, und die Eckpunkte $p_1, \ldots, p_k$ des Umrisses berechnet. Gehen Sie in Ihrer Lösung auf die folgenden Aspekte ein.

1) In welche Richtung verläuft die Scanline, und was sind die Haltepunkte?

2) Welche Objekte muss die Datenstruktur verwalten, und was ist eine angemessene Datenstruktur?

3) Was passiert, wenn die Scanline auf einen neuen Haltepunkt trifft?

4) Wie lässt sich die Lösung auslesen?

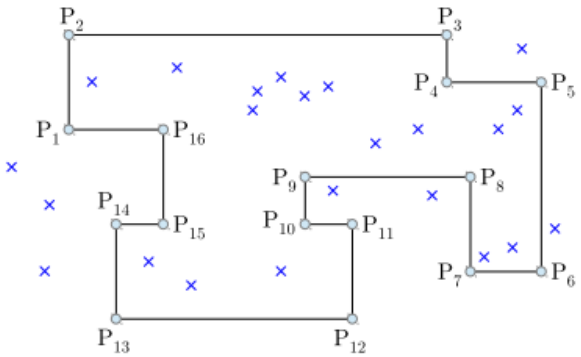
1 P b) Geben Sie die Laufzeit Ihres in a) entwickelten Algorithmus an und begründen Sie Ihre Antwort.

HS13

Aufgabe 5.

Wir betrachten eine Menge von n Kühen sowie ein Gehege aus m geradlinigen orthogonalen Zaunsegmenten. Die Position jeder Kuh i, $1 \leq i \leq n$, kann mittels eines GPS-Empfängers bestimmt werden und ist durch den Punkt $K_i \in \mathbb{Q}^2$ gegeben. Das Gehege ist ein geschlossenes einfaches Polygon mit den Eckpunkten (also den Zaunpfählen) $P_i \in \mathbb{Q}^2$, $1 \leq j \leq m$. Dabei seien P_i und P_{i+1} für $i \in \{1, \ldots, m-1\}$ und zusätzlich P_m und P_1 jeweils durch ein Zaunsegment verbunden. Weiterhin sei P_1 der am weitesten links liegende Punkt (falls es mehr als nur einen solchen Punkt gibt, dann sei P_1 unter diesen der am weitesten unten liegende Punkt). Sie dürfen davon ausgehen, dass sich keine Kuh direkt auf einem Zaunsegment befindet, d.h. kein Punkt K_i liegt auf einer Polygonkante. Wir wollen ermitteln, wie viele Kühe sich innerhalb des Geheges befinden.

9 P (a) Wir nehmen der Einfachheit halber an, dass alle Zaunsegmente parallel zur x- oder zur y-Achse verlaufen. Geben Sie einen Scanline-Algorithmus an, der als Eingabe die Positionen der Kühe K_i sowie die Positionen der Zaunpfähle P_j erhält, und die Anzahl der Kühe berechnet, die sich innerhalb des Geheges befinden.



1 P (b) Geben Sie die Laufzeit des Verfahrens aus (a) an.

FS13