Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Institut für Theoretische Informatik
Peter Widmayer
Tobias Pröger
Thomas Tschager

Beispiellösung zur Prüfung
Datenstrukturen und Algorithmen
D-INFK

5. August 2015

Name, Vorname: _____

Stud.-Nummer: _____

Ich bestätige mit meiner Unterschrift, dass ich diese Prüfung unter regulären Bedingungen ablegen konnte und dass ich die untenstehenden Hinweise gelesen und verstanden habe.

Unterschrift: _____

Hinweise:

- Ausser einem Wörterbuch dürfen Sie keine Hilfsmittel verwenden.
- Bitte schreiben Sie Ihre Studierenden-Nummer auf **jedes** Blatt.
- Melden Sie sich bitte **sofort**, wenn Sie sich während der Prüfung in irgendeiner Weise bei der Arbeit gestört fühlen.
- Bitte verwenden Sie für jede Aufgabe ein neues Blatt. Pro Aufgabe kann nur eine Lösung angegeben werden. Ungültige Lösungsversuche müssen klar durchgestrichen werden.
- Bitte schreiben Sie **lesbar** mit blauer oder schwarzer Tinte. Wir werden nur bewerten, was wir lesen können.
- Sie dürfen alle Algorithmen und Datenstrukturen aus der Vorlesung verwenden, ohne sie noch einmal zu beschreiben. Wenn Sie sie modifizieren, reicht es, die Modifikationen zu beschreiben.
- Die Prüfung dauert 180 Minuten.

Viel Erfolg!

Stud.-Nummer: _____

Aufgabe	1	2	3	4	**Σ**
Mögl. Punkte	18	12	9	11	50
$\sum$ Punkte					

Aufgabe 1.

Hinweise:

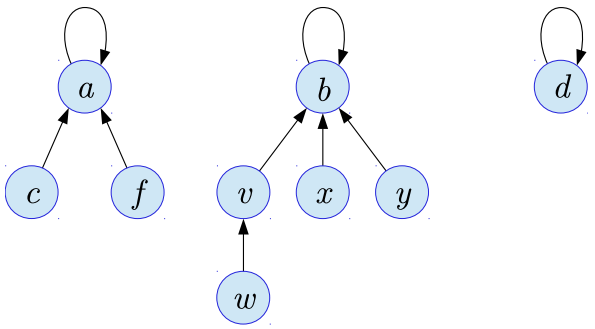
1) In dieser Aufgabe sollen Sie **nur die Ergebnisse** angeben. Diese können Sie direkt bei den Aufgaben notieren.
2) Sofern Sie die Notationen, Algorithmen und Datenstrukturen aus der Vorlesung "Datenstrukturen & Algorithmen" verwenden, sind Erklärungen oder Begründungen nicht notwendig. Falls Sie jedoch 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 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.

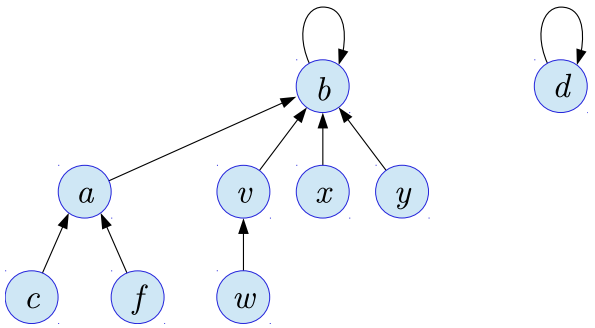
	12			4	16	6	5	9	20	
0	1	2	3	4	5	6	7	8	9	10

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

Lösung: Nach $\textsc{Union}(a, c)$:

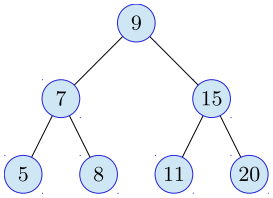


Nach $\textsc{Union}(\textsc{Find}(f), b)$:

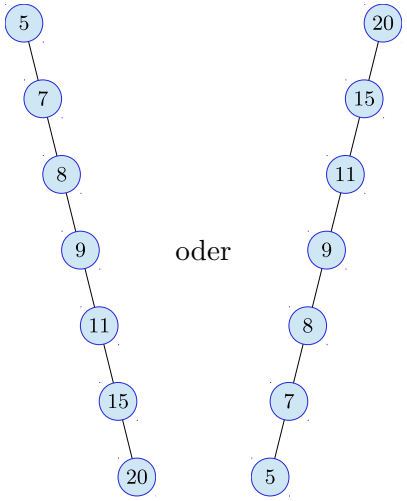


1 P c) Gegeben sei die Schlüsselmenge $\mathcal{K} = \{5, 9, 8, 11, 15, 7, 20\}$. Zeichnen Sie die beiden binären Suchbäume, die genau die Schlüssel aus $\mathcal{K}$ verwalten und die unter allen möglichen Suchbäumen minimale bzw. maximale Höhe haben.

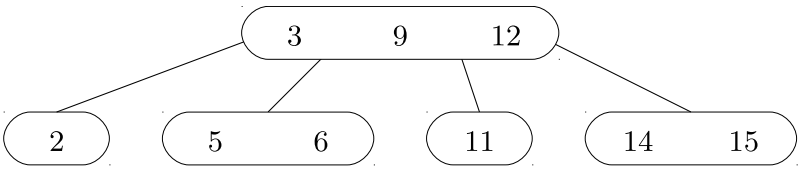
Baum minimaler Höhe:



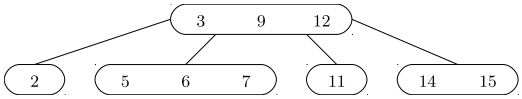
Baum maximaler Höhe:



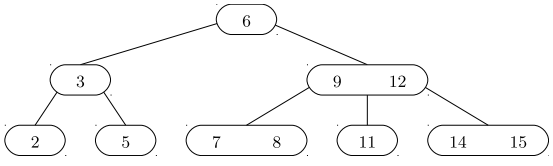
1 P d) Fügen Sie in den untenstehenden 2-3-4-Baum (B-Baum der Ordnung 4) zuerst den Schlüssel 7 und in den entstehenden Baum den Schlüssel 8 ein. Führen Sie auch die zugehörigen Strukturänderungen durch.



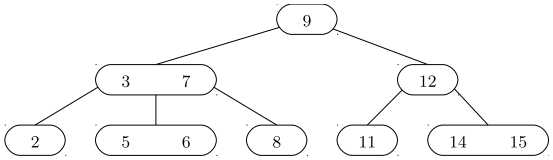
Nach Einfügen von 7:



Nach Einfügen von 8:



oder



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

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

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

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.

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

Lösung: Es sind $\log(n) \in \mathcal{O}(\sqrt{n})$ und $\sqrt{n} \in \mathcal{O}(n/\log(n))$ sowie $\binom{n}{3} \in \Theta(n^3)$. Die einzig mögliche Reihenfolge ist daher

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

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.

Hinweise:

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

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

Lösung: Da wir annehmen dürfen, dass n eine Potenz von 5 ist, gilt $n = 5^k$ für ein $k \in \mathbb{N}$. Wir teleskopieren, um auf eine Formel für $T(n)$ zu kommen:

$$\begin{aligned} T(n) &= T(n/5) + 4n + 1 = \big(T(n/5^2) + 4n/5 + 1\big) + 4n + 1 \\ &= T(n/5^2) + 4n\Big(\frac{1}{5^0} + \frac{1}{5^1}\Big) + 2 = \big(T(n/5^3) + 4n/5^2 + 1\big) + 4n\Big(\frac{1}{5^0} + \frac{1}{5^1}\Big) + 2 \\ &= T(n/5^3) + 4n\Big(\frac{1}{5^0} + \frac{1}{5^1} + \frac{1}{5^2}\Big) + 3 = \ldots \stackrel{!}{=} T(n/5^k) + 4n\Big(\sum_{i=0}^{k-1} \frac{1}{5^i}\Big) + k \\ &= T(5^k/5^k) + 4 \cdot 5^k\Big(\sum_{i=0}^{k-1} \frac{1}{5^i}\Big) + k = T(1) + 4 \cdot 5 \cdot \sum_{i=0}^{k-1} 5^i + k \\ &= 5 + 4 \cdot 5 \cdot \frac{5^k - 1}{5 - 1} + k = 5 + 5^{k+1} - 5 + k = 5^{k+1} + k = 5n + \log_5(n). \end{aligned}$$

Wir beweisen nun unsere Annahme durch vollständige Induktion über k.

Induktionsverankerung ($k = 0$): Es gilt $T(5^0) = T(1) = 5 = 5^1 + 0 = 5^{k+1} + k$.

Induktionsannahme: Für ein $k \in \mathbb{N}_0$ sei $T(5^k) = 5^{k+1} + k$.

Induktionsschritt ($k \to k + 1$):

$$\begin{aligned} T(5^{k+1}) &= T(5^k) + 4 \cdot 5^{k+1} + 1 \stackrel{\text{Ind.-Ann.}}{=} 5 \cdot 5^k + k + 4 \cdot 5^{k+1} + 1 \\ &= 5^{k+1} + 4 \cdot 5^{k+1} + k + 1 = 5 \cdot 5^{k+1} + (k+1) = 5n + \log_5(n). \end{aligned}$$

1 P 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                         ;
```

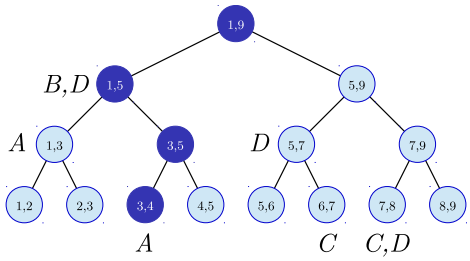
Lösung: Die Schleife in Schritt 1 wird $\Theta(n)$ Mal durchlaufen. Die Schleife in Schritt 2 wird $\Theta(n - i)$ Mal durchlaufen, und da $i \leq n/2$ für etwa $n/4$ Werte von i, ist die Laufzeit von Schritt 2 in $\Theta(n)$. Die Schleife in Schritt 3 wird $\Theta(\log n)$ Mal durchlaufen. Die Gesamtlaufzeit ist daher in $\Theta(n^2 \log n)$.

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

Lösung: Die äussere Schleife wird $\Theta(n^2)$ Mal durchlaufen. Die erste innere Schleife wird $\Theta(\log i) \subseteq \mathcal{O}(\log(n^2)) = \Theta(\log n)$ Mal durchlaufen, die zweite innere Schleife $\Theta(\sqrt{n})$ Mal. Da beiden inneren Schleifen nacheinander ausgeführt werden, beträgt die Gesamtlaufzeit $\Theta(n^2(\log n + \sqrt{n})) = \Theta(n^{5/2})$.

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.



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.

Lösung: Die Anzahl der Blätter beträgt genau 3^h.

Induktionsverankerung ($h = 0$): Im Fall von $h = 0$ ist die Wurzel ein Blatt, und da ein Blatt keine Nachfolger besitzt, beträgt die Anzahl der Blätter genau $1 = 3^0$.

Induktionsannahme: Angenommen, in jedem vollständigen ternären Suchbaum, in dem alle Blätter Tiefe h haben, hat es 3^h viele Blätter.

Induktionsschritt ($h \to h + 1$): Man betrachte einen vollständigen ternären Suchbaum T, in dem alle Blätter Tiefe $h + 1$ haben. Dieser Baum besteht aus einer Wurzel mit genau drei Teilbäumen T_1, T_2, T_3, in denen alle Blätter Tiefe h haben. Nach Induktionsannahme hat jeder der drei Teilbäume T_1, T_2, T_3 genau 3^h viele Blätter. Da die Wurzel kein Blatt ist, hat T also $3 \cdot 3^h = 3^{h+1}$ viele Blätter, was die Aussage beweist.

Aufgabe 2.

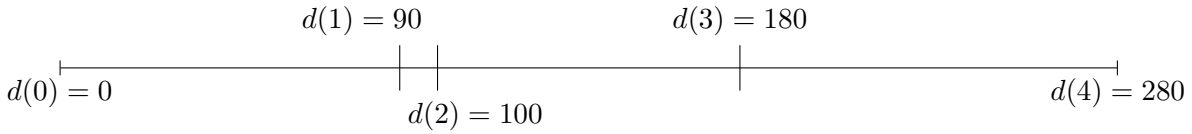
Motivation. Im Rahmen eines Infrastrukturprojekts sollen entlang einer Autobahn Schnellladestationen für Elektroautos gebaut werden. Bei der Planung wird davon ausgegangen, dass ein Elektroauto mit einer vollen Batterieladung 100 km zurücklegen kann. Es werden n mögliche Standorte definiert, aus denen 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 die optimale Teilmenge I mit *beliebiger* Größe gesucht wird.

Beispiel: Es stehen $n = 3$ Standorte zur Auswahl, wobei $d(1) = 90, d(2) = 100$ und $d(3) = 180$. 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.

Lösung:

Definition der DP-Tabelle: Wir verwenden eine eindimensionale Tabelle T mit $n + 2$ Einträgen. Der Eintrag $T[k]$ enthält die minimalen Kosten für eine Auswahl von Ladestationen entlang der Teilstrecke vom Ausgangspunkt bis zum k-ten Standort.

Berechnung eines Eintrags: Der Eintrag $T[0]$ wird mit 0 initialisiert. Alle weiteren Einträge der Tabelle werden mit ∞ initialisiert. Um einen Eintrag $T[i]$ mit $i > 0$ zu berechnen, müssen wir alle Einträge $T[j]$ mit $j < i$ und $d(i) - d(j) \leq 100$ berücksichtigen. Für jeden dieser Einträge berechnen wir $T[i] = \min(T[i], T[j] + c(d(i) - d(j)))$. Die minimalen Kosten über alle Einträge $T[j]$ werden im Eintrag $T[i]$ gespeichert. Für einen Eintrag $T[i]$ mit $i > 0$ gilt also

$$T[i] = \min_{0 \leq j < i} \left(T[j] + c(d(i) - d(j)) \right), \quad \text{sodass } d(i) - d(j) \leq 100.$$

Berechnungsreihenfolge: Da ein Tabelleneintrag nur von Einträgen mit kleinerem Index abhängt, können die Einträge $T[k]$ für aufsteigendes k berechnet werden.

Auslesen der Lösung: Die Lösung steht am Ende im Eintrag $T[n + 1]$.

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.

Lösung: Eine Auswahl von Standorten mit minimalen Kosten kann durch Rückverfolgung berechnet werden. Wir setzen zu Beginn $k = n + 1$ und wiederholen die folgende Prozedur solange $k > 0$:

1. Finde $T[k']$ sodass $k' < k, d(k) - d(k') \leq 100$ und $T[k] = T[k'] + c(d(k) - d(k'))$.
2. Gib Standort k' aus, sofern $k' \neq 0$ und setze $k = k'$.

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.

Lösung: Um einen Eintrag $T[i]$ in Teilaufgabe a) zu berechnen, müssen alle Einträge $T[j]$ mit $j < i$ und $d(i) - d(j) \leq 100$ berücksichtigt werden. Da es höchstens n solche Einträge gibt, kann ein Eintrag in Zeit $\mathcal{O}(n)$ berechnet werden. Insgesamt ist die Gesamtlaufzeit für Teilaufgabe a) daher $\mathcal{O}(n^2)$.

In Teilaufgabe b) wird jeder Eintrag genau einmal betrachtet, weshalb die Rückverfolgung in Zeit $\mathcal{O}(n)$ durchgeführt werden. Die Gesamtlaufzeit für beide Teilaufgaben liegt daher in $\mathcal{O}(n^2)$ und ist polynomiell.

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.

Lösung: Der kürzeste Pfad in einem gewichteten Graphen kann mit dem Algorithmus von Dijkstra in Zeit $\mathcal{O}(|E| + |V| \log(|V|))$ berechnet werden, sofern die Kantengewichte positiv sind und eine Priority Queue (z.B. ein Fibonacci-Heap) bei der Implementierung verwendet wird. Außerdem kann angenommen werden, dass der Graph als Adjazenzlisten gegeben ist.

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 Graphen $G = (V', E')$, 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.

Lösung: Die Menge V' besteht aus den Knoten s und t, sowie aus zwei Knoten v_+ und v_- für jeden Knoten $v \in V \backslash \{s, t\}$. Dabei bezeichnet v_-, dass auf dem Weg von s nach zur Station v der Gutschein noch nicht verwendet wurde, und v_+, dass der Gutschein bereits verwendet wurde. Die Menge E' enthält folgende Kanten: für jede Kante $(v, w) \in E$ mit $v, w \notin \{s, t\}$ gibt zwei Kanten (v_-, w_-) und (v_+, w_+) mit Gewicht $c(v, w)$. Für die Kanten $\{s, v\} \in E$ wird eine Kanten $\{s, v_-\}$ (mit Gewicht $c(s, v)$) hinzugefügt und, falls $c(s, v) \geq 50$, eine Kante $\{s, v_+\}$ (mit Gewicht $c(s, v) - 30$). Weiters gibt es eine Kante (v_-, w_+) mit Gewicht $c(v, w) - 30$ in E', sofern $c(v, w) \geq 50$. Wird eine solche Kante gewählt, so wird der Gutschein auf der Teilstrecke von v nach w verwendet. Schliesslich gibt es für jede Kante $\{v, t\} \in E$ zwei Kanten $\{v_+, t\}$ (mit Gewicht $c(v, t)$) und $\{v_-, t\}$ (mit Gewicht $c(v, t) - 30$ falls $c(v, t) \geq 50$ und $c(v, t)$ andernfalls). Da für jeden Knoten in V genau zwei Knoten zu V' hinzugefügt wurden und für jede Kante in E höchstens drei Kanten in E' sind, kann der kürzeste Pfad in der selben asymptotischen Laufzeit wie in Teilaufgabe a) berechnet werden.

Das Zugunternehmen möchte die Fahrkarten aller Reisenden kontrollieren. Dazu sollen Kontrolleure auf verschiedenen Teilstrecken die Fahrkarten überprüfen. Es soll sichergestellt werden, dass

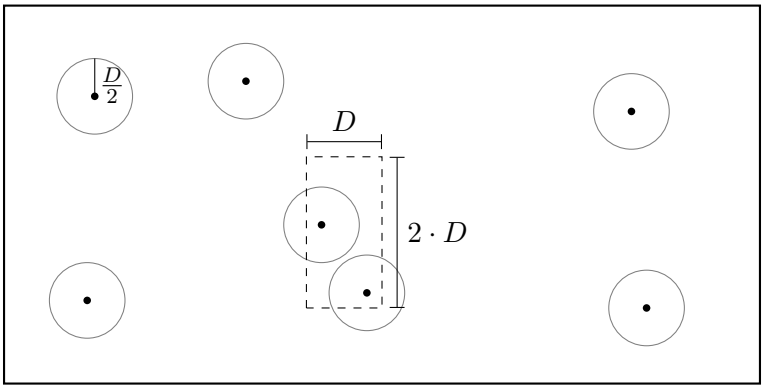
alle Reisenden unabhängig von der Wahl der Route von s nach t kontrolliert werden. An wie vielen Teilstrecken müssen mindestens Kontrollen durchgeführt werden, damit jede mögliche Route mindestens eine Teilstrecke benutzt, an der kontrolliert wird?

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?

Lösung: Das Netzwerk N besteht aus der Knotenmenge $V'' = V$ und der Kantenmenge $E'' = E$. Die Kapazitäten aller Kanten werden auf 1 gesetzt. Ein maximaler Fluss kann beispielsweise mit dem Algorithmus von Ford und Fulkerson berechnet werden. Da die Kapazitäten der Kanten 1 sind, ist der maximale Fluss $|V|$ (eine Kante von jedem Knoten zu t). Daher ist die Laufzeit $\mathcal{O}(|V| \cdot |E|)$. Anschließend kann durch Anwendung des Max-Flow-Min-Cut Theorems abgelesen werden, dass der Wert des maximalen Flusses genau der Anzahl der Teilstrecken entspricht, an denen kontrolliert werden muss.

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.

Lösung: Wir können das Rechteck in acht kleine Quadrate mit Seitenlänge $D/2$ unterteilen. Nehmen wir an, dass zwei Punkte in einem Quadrat liegen. Dann entspricht die maximale Distanz dieser beiden Punkte der Länge der Diagonale des Quadrats, d.h. $\sqrt{2} \cdot D/2 < D$. Daher kann in jedem Quadrat höchstens ein Punkt liegen, sofern die paarweisen Distanzen mindestens D sind. Es können also höchstens acht Punkte im Rechteck liegen.

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.

Lösung:

1) Wir verwenden eine vertikale Scanline, die von links nach rechts verläuft. Für jeden Punkt (x_i, y_i) definieren wir zwei Haltepunkte x_i und $x_i + D$.
2) Wir können als Scanline-Datenstruktur einen AVL-Baum verwenden, der Punkte nach y-Koordinate geordnet speichert, d.h. als Schlüssel die y-Koordinaten der Punkte benutzt. Die Punkte verlassen die Datenstruktur in der gleichen Reihenfolge, in der sie eingefügt werden, und können zusätzlich durch eine Liste verkettet werden.
3) Falls die Scanline auf einen Haltepunkt x_i für einen Punkt (x_i, y_i) trifft, d.h. auf den ersten

Haltepunkt, werden alle Punkte in der Datenstruktur betrachtet, deren y-Koordinate im Bereich $[y_i - D, y_i + D]$ liegen. Für jeden Punkt in diesem Interval wird die Distanz zum Punkt (x_i, y_i) berechnet. Falls eine dieser Distanzen kleiner als D ist, wird der Alarm ausgelöst und abgebrochen. Andernfalls wird der Punkt (x_i, y_i) in die Datenstruktur eingefügt.

Trifft die Scanline nun auf den zweiten Haltepunkt eines Punktes (x_i, y_i), d.h. auf den Haltepunkt $x_i + D$, dann wird der Punkt lediglich aus der Datenstruktur gelöscht.

4) Es werden $\mathcal{O}(n)$ Haltepunkte besucht. Ein Punkt kann in Zeit $\mathcal{O}(\log n)$ zur Datenstruktur hinzugefügt, beziehungsweise aus der Datenstruktur gelöscht werden. In der selben Zeit kann auch eine Bereichsabfrage gemacht werden. Aus Teilaufgabe a) folgt, dass außerdem nur eine konstante Anzahl von Distanzen zwischen Punkten berechnet werden muss. Deshalb ist die Laufzeit aller Operationen in $\mathcal{O}(n \log n)$. Das Sortieren der Punkte benötigt ebenfalls eine Laufzeit in $\mathcal{O}(n \log n)$. Somit ist auch die Gesamtlaufzeit in $\mathcal{O}(n \log n)$.

Hinweis: Der Einfachheit halber nehmen wir an, dass für zwei beliebige Flugzeuge i und j stets gilt, dass $x_i \neq x_j$. 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.