Karlsruher Institut für Technologie

Prof. Dr.-Ing. Anne Koziolek
Architecture-driven Requirements Engineering (ARE)
https://sdqweb.ipd.kit.edu/wiki/Programmieren
programmieren-vorlesung@ipd.kit.edu

Programmieren – Wintersemester 2020/21

Abschlussaufgabe 1 20 Punkte
Version 1.1

Ausgabe: 15.02.2021, ca. 13:00 Uhr
Praktomat: 01.03.2021, 13:00 Uhr
Abgabefrist: 16.03.2021, 06:00 Uhr

Bei Fragen

In unseren *FAQs* finden Sie einen Überblick über häufig gestellte Fragen und die entsprechenden Antworten zum Modul „Programmieren“. Bitte lesen Sie diese sorgfältig durch, noch bevor Sie Fragen haben, und überprüfen Sie diese regelmäßig und eigenverantwortlich auf Änderungen.

https://sdqweb.ipd.kit.edu/wiki/Programmieren/FAQ

Abgabehinweise

Bitte beachten Sie, dass das erfolgreiche Bestehen der öffentlichen Tests für eine erfolgreiche Abgabe dieses Blattes notwendig ist. Der Praktomat wird Ihre Abgabe zurückweisen, falls eine der nachfolgenden Regeln verletzt ist. Eine zurückgewiesene Abgabe wird automatisch mit null Punkten bewertet. Planen Sie entsprechend Zeit für Ihren ersten Abgabeversuch ein.

- Achten Sie auf fehlerfrei kompilierenden Programmcode.
- Verwenden Sie keine Elemente der Java-Bibliotheken, ausgenommen Elemente der Pakete `java.lang`, `java.util`, `java.util.regex`, `java.util.function` und `java.util.stream`.
- Achten Sie darauf, nicht zu lange Zeilen, Methoden und Dateien zu erstellen. Sie müssen bei Ihren Lösungen eine maximale Zeilenbreite von 120 Zeichen einhalten.
- Halten Sie alle Whitespace-Regeln ein.
- Halten Sie alle Regeln zu Variablen-, Methoden- und Paketbenennung ein.
- Wählen Sie geeignete Sichtbarkeiten für Ihre Klassen, Methoden und Attribute.
- Nutzen Sie nicht das `default`-Package.
- `System.exit()` und `Runtime.exit()` dürfen nicht verwendet werden.

- Halten Sie die Regeln zur Javadoc-Dokumentation ein.
- Halten Sie auch alle anderen Checkstyle-Regeln an.

Bearbeitungshinweise

Diese Bearbeitungshinweise sind relevant für die Bewertung Ihrer Abgabe, jedoch wird der Praktomat Ihre Abgabe **nicht** zurückweisen, falls eine der nachfolgenden Regeln verletzt ist.

- Beachten Sie, dass Ihre Abgaben sowohl in Bezug auf objektorientierte Modellierung als auch Funktionalität bewertet werden. Halten Sie die Hinweise zur Modellierung im Ilias-Wiki ein.
- Programmcode muss in englischer Sprache verfasst sein.
- Kommentieren Sie Ihren Code angemessen: So viel wie nötig, so wenig wie möglich.
- Die Kommentare sollen einheitlich in englischer oder deutscher Sprache verfasst werden.
- Wählen Sie aussagekräftige Namen für alle Ihre Bezeichner.

Plagiat

Es werden nur selbstständig angefertigt Lösungen akzeptiert. Das Einreichen fremder Lösungen, seien es auch nur teilweise Lösungen von Dritten, aus Büchern, dem Internet oder anderen Quellen, ist ein Täuschungsversuch und führt zur Bewertung „nicht bestanden“. Ausdrücklich ausgenommen hiervon sind Quelltextschnipsel von den Vorlesungsfolien und aus den Lösungsvorschlägen des Übungsbetriebes in diesem Semester. Alle benutzten Hilfsmittel müssen vollständig und genau angegeben werden und alles, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde, muss deutlich kenntlich gemacht werden. Ebenso stellt die Weitergabe einer Lösung oder von Teilen davon eine Störung des ordnungsgemäßen Ablaufs der Erfolgskontrolle dar. Dieser Ordnungsverstoß kann ebenfalls zum Ausschluss der Erfolgskontrolle führen. Für weitere Ausführungen sei auf die Einverständniserklärung (Disclaimer) verwiesen.

Checkstyle

Der Praktomat überprüft Ihre Quelltexte während der Abgabe automatisiert auf die Einhaltung der Checkstyle-Regeln. Es gibt speziell markierte Regeln, bei denen der Praktomat die Abgabe zurückweist, da diese Regel verpflichtend einzuhalten ist. Andere Regelverletzungen können zu Punktabzug führen. Sie können und sollten Ihre Quelltexte bereits während der Entwicklung auf die Regeleinhaltung überprüfen. Das Programmieren-Wiki im Ilias beschreibt, wie Checkstyle verwendet wird.

>_ Terminal-Klasse

Laden Sie für diese Aufgabe die Terminal-Klasse herunter und platzieren Sie diese unbedingt im Paket `edu.kit.informatik`. Die Methode `Terminal.readLine()` liest eine Benutzereingabe von der Konsole und ersetzt `System.in`. Die Methode `Terminal.printLine()` schreibt eine Ausgabe auf die Konsole und ersetzt `System.out`. Verwenden Sie für jegliche Konsoleneingabe oder Konsolenausgabe die Terminal-Klasse. Verwenden Sie in keinem Fall `System.in` oder `System.out`. Laden Sie die Terminal-Klasse niemals zusammen mit Ihrer Abgabe hoch.

Interaktive Benutzerschnittstelle

Da wir automatische Tests Ihrer interaktiven Benutzerschnittstelle durchführen, müssen die Ausgaben exakt den Vorgaben entsprechen. Insbesondere sollen sowohl Klein- und Großbuchstaben als auch die Leerzeichen und Zeilenumbrüche genau übereinstimmen. Setzen Sie nur die in der Aufgabenstellung angegebenen Informationen um. Geben Sie auch keine zusätzlichen Informationen aus. Bei Fehlermeldungen dürfen Sie den englischsprachigen Text frei wählen, er sollte jedoch sinnvoll sein. Jede Fehlermeldung muss aber mit `Error,` beginnen und darf keine Sonderzeichen, wie beispielsweise Zeilenumbrüche oder Umlaute, enthalten.

Abgabemodalitäten

Die Praktomat-Abgabe wird am **Montag, den 1. März 2020 um 13:00 Uhr**, freigeschaltet. Achten Sie unbedingt darauf, Ihre Dateien im Praktomat bei der richtigen Aufgabe vor Ablauf der Abgabefrist hochzuladen. Beginnen Sie frühzeitig mit dem Einreichen, um Ihre Lösung dahingehend zu testen, und verwenden Sie das Forum, um eventuelle Unklarheiten zu klären.

- Geben Sie Ihre Klassen zur Abschlussaufgabe 1 in Einzelarbeit als `*.java`-Dateien ab.

Aufgabe A: Analyse von Fluchtwegenetzen (20 Punkte)

Zum Schutz aller Studierenden und Beschäftigten muss das Karlsruher Institut für Technologie Vorkehrungen treffen, um sicherzustellen, dass sich alle Personen im Gefahrenfall unmittelbar in Sicherheit bringen können und schnell gerettet werden können. Die Gestaltung von Fluchtwegen spielt dabei eine wesentlich Rolle. Ein Fluchtweg ist ein Weg, der im Falle einer notwendigen Flucht schnell und sicher ins Freie oder in einen gesicherten Bereich führt. Für den Schutz von Personen wollen die KIT-Verantwortlichen für jeden Raum individuell analysieren, wie viele Personen pro Minute in Sicherheit gebracht werden können.

Eine Herausforderung besteht hierbei vor allem, wenn sich in einem Raum, wie beispielsweise ein Hörsaal, eine große Anzahl von Personen befindet und diese über Fluchtwege mit begrenzter Kapazität in Sicherheit gebracht werden müssen. In diesem Fall ist für diese Wege die Anzahl der Personen, die sie pro Minute passieren können, begrenzt, und ebenso können sich die Personen jeweils nur in eine Richtung auf ihnen bewegen.

Für diese Aufgabe soll ein Programm zur Verwaltung und Analyse von Fluchtwegen konzipiert und implementiert werden. Dazu sollen per Kommandozeile ein oder mehrere Fluchtwegenetze eingegeben werden, über die dann jeweils die maximale Anzahl von Personen ermittelt werden kann, die sich pro Minute in Sicherheit bringen lassen.

A.1 Fluchtwegenetze

Ein Fluchtwegenetz $N = (G, k, s, z)$ besteht aus einem gerichteten Graphen $G = (V, E)$ mit zwei gekennzeichneten Knoten, dem Startknoten $s \in V$ und dem Zielknoten $z \in V$, und einer Kapazitätsfunktion $k \colon E \to \mathbb{K}$, welche jedem Fluchtwegeabschnitt $e \in E$ eine Kapazität zuweist $e \mapsto k(e)$.

Innerhalb des Programms zur Verwaltung und Analyse von Fluchtwegenetzen haben die gerichteten Graphen G mit ihrer Kapazitätsfunktion k aus dem zugehörigen Fluchtwegenetz N immer eine eindeutige Kennung, mit welcher sie identifiziert werden. Diese Kennung besteht immer aus mindestens einem und maximal sechs lateinischen Großbuchstaben `[A-Z]{1,6}`.

A.1.1 Fluchtwegeabschnitte

Ein gerichteter Graph $G = (V, E)$ besteht aus einer Menge von Knoten V und einer Menge von geordneten Knotenpaaren $E \subseteq \{(v, w) \mid (v, w) \in V \times V \land v \neq w\}$ von Fluchtwegeabschnitten. Infolgedessen ist ein Fluchtwegeabschnitt e immer eindeutig durch seine beiden Knoten (v, w) bestimmt. Diese Fluchtwegeabschnitte $(v, w) \in E$ eines gerichteten Graphen sind gerichtete Wegabschnitte und gehen von v nach w, sodass diese nur in eine Richtung durchlaufen werden können und sich Personen auch nur in diese Richtung auf diesem Fluchtwegeabschnitt bewegen können. In einem Fluchtwegenetz darf es keine parallelen gegenläufigen Fluchtwegeabschnitte geben $\forall v\, w \colon (v, w) \in E \to (w, v) \notin E$.

Ein Knoten v ist eine Stellen in einem Fluchtwegenetz $v \in V$, an denen sich Fluchtwegeabschnitte treffen und oder von denen sie abzweigen. Innerhalb eines Fluchtwegenetzes haben Knoten immer

eine eindeutige Kennung, mit der sie identifiziert werden. Die Kennung eines Knotens besteht immer aus mindestens einem und maximal sechs lateinischen Kleinbuchstaben `[a-z]{1,6}`. Innerhalb eines Fluchtwegenetzes darf es nicht zwei Knoten mit der gleichen Kennung geben, jedoch kann es verschiedene Knoten mit der gleichen Kennung in verschiedenen Fluchtwegenetzen geben.

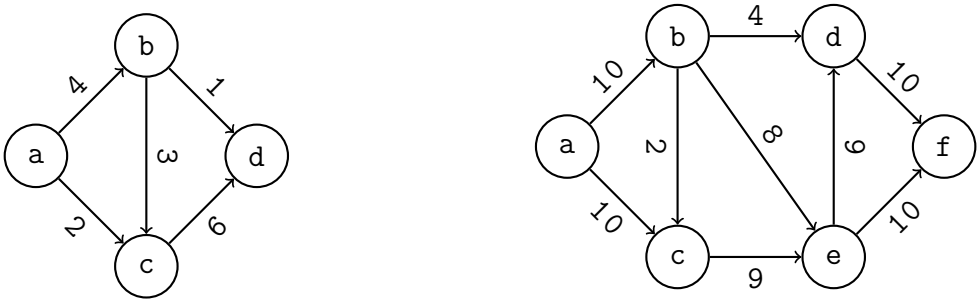
A.1.2 Start- und Zielknoten

Der Startknoten $s \in V$ und der Zielknoten $z \in V$ müssen Teil des Fluchtwegenetzes sein und dürfen nicht identisch sein $s \neq z$. In den Startknoten dürfen keine Fluchtwegeabschnitte hineingehen $\forall v \in V\colon (v, s) \notin E$ und aus dem Zielknoten dürfen keine Fluchtwegeabschnitte herausgehen $\forall v \in V\colon (z, v) \notin E$.

Um für diese Herausforderung die Obergrenze der maximalen Anzahl von Personen zu finden, welche pro Minute in Sicherheit gebracht werden können, wird angenommen, dass die mögliche Kapazität der Personen in dem Start- und Zielknoten für die Analyse nicht begrenzt ist. Zum Beispiel könnte ein Startknoten der Gerhard-Goos-Hörsaal[1] und ein Zielknoten der „Bunker“[2] unter dem Informatik-Hauptgebäude sein.

A.1.3 Kapazitätsfunktion

Kapazität $\mathbb{K}$ für einen Fluchtwegeabschnitt $e \in E$ ist durch eine Kapazitätsfunktion gegeben. Diese Kapazität ist die maximale Anzahl der Personen, die einen Fluchtwegeabschnitt pro Minute benutzen können. Sie ist stets ein positiver ganzzahliger Wert mit $\mathbb{K} \coloneqq \left\{x \in \mathbb{N} \mid x \not> 0 \wedge x \leq (2^{31} - 1)\right\}$. Die Kapazitätsfunktion ist eine Abbildung $k\colon E \to \mathbb{K}$, welche jedem Fluchtwegeabschnitt in einem Fluchtwegenetz eine Kapazität zuordnet. Die Kapazität eines Fluchtwegeabschnittes wird mit $k(e)$ bezeichnet.



(a) Fluchtwegenetz `ABC` (b) Fluchtwegenetz `XYZ`

Abbildung 0.1: Beispiel für zwei unabhängige Fluchtwegenetze mit ihren Kapazitäten

[1] `https://de.wikipedia.org/wiki/Gerhard_Goos`
[2] `https://blog.eriq.de/2009/08/ein-bombensicherer-arbeitsplatz/`

A.2 Durchfluss

Der Durchfluss ist eine Funktion $f: E \to \mathbb{K}$, welche jedem Fluchtwegeabschnitt $e \in E$ im Fluchtwegenetz einen Durchflusswert $f(e) \in \mathbb{K}$ zuweist. Dabei müssen die folgenden drei Bedingungen erfüllt sein:

Kapazitätskonformität Der Durchfluss entlang eines Fluchtwegeabschnitt kann dessen Kapazität nicht überschreiten $\forall e \in E: f(e) \leq k(e)$. Somit ist die angegebene Kapazität $k(e)$ die Obergrenze für die Anzahl der Personen, welche pro Minute über diesen Fluchtwegeabschnitt $e \in E$ in Sicherheit gebracht werden können.

Flusserhalt Mit Ausnahme des Startknoten $s \in V$ und dem Zielknoten $z \in V$, muss in jeden Knoten genau so viel hinein wie heraus fließen $\forall v \in V: v \neq s \wedge v \neq z \Rightarrow \sum_{w \in V} f(v, w) = 0$. Somit ist der Nettodurchfluss zu einem Knoten 0, bis auf den Startknoten, von dem aus die Menschen in Sicherheit gebracht werden müssen, und den Zielknoten, zu dem die Menschen in Sicherheit gebracht werden.

Wertbeständigkeit Der vom Startknoten ausgehende Durchfluss muss gleich dem am Zielknoten ankommenden Durchfluss sein $\sum_{(s,v) \in E} f(s, v) = \sum_{(w,z) \in E} f(w, z)$.

A.2.1 Restkapazitätennetzwerk

Die Restkapazität k_f eines Fluchtwegeabschnitts e in Bezug auf seinen Durchfluss $f(e)$ ist die Differenz zwischen seiner Kapazität $k(e)$ und seinem Durchfluss $k_f(e) = k(e) - f(e)$. Das Restkapazitätennetzwerk $G_f = (V, E_f)$ stellt die Menge der verfügbaren Kapazität auf der Menge der Fluchtwegeabschnitte im Fluchtwegenetz N dar, sodass dieses die Knotenmenge V, die Fluchtwegeabschnitte $E_f = \{e \in V \times V: k_f(e) > 0\}$ und die Kapazitätsfunktion k_f hat.

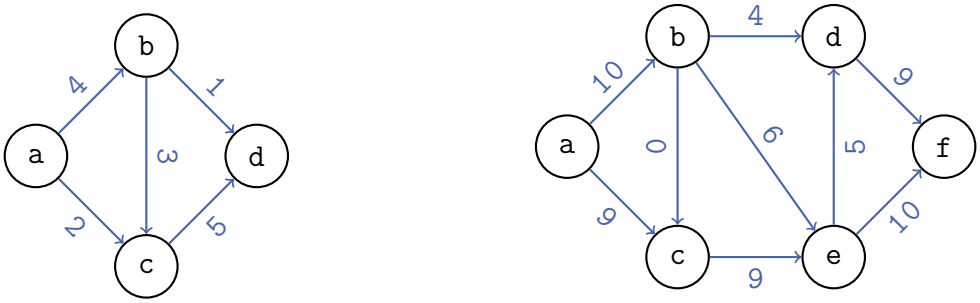
A.2.2 Optimierungspfad

Ein Pfad p der Länge $n \in \mathbb{N}$ vom Knoten v_0 zum Knoten v_n in G ist eine Folge von n Fluchtwegeabschnitten $(e_1, \ldots, e_n)$ von G, so dass $\forall i \in \mathbb{N}: 1 \leq i \leq n \Rightarrow e_i = \{v_{i-1}, v_i\}$. Ein Optimierungspfad ist ein Pfad p im Restkapazitätennetzwerk G_f mit $v_0 = s, v_n = z$ und $k_f(e_i) > 0$. Ein Fluchtwegenetze ist im maximalen Durchfluss, wenn und nur wenn es keinen Optimierungspfad im Restkapazitätennetzwerk gibt.

A.2.3 Maximaler Durchfluss

Das Problem des maximalen Durchflusses besteht darin, unter Berücksichtigung der drei Bedingungen aus Abschnitt A.2 so viele Personen wie möglich vom Startknoten zum Zielknoten zu leiten. Mit anderen Worten, den Durchfluss f für ein Fluchtwegenetze N mit dem maximalen Wert $f_{\max} \in \mathbb{N}$ zu finden. Dabei ist der Wert des Flusses die Anzahl der Personen, die pro Minute von dem Startknoten s zum Zielknoten z geleitet werden $\sum_{(v,z) \in E} f(v, z)$. Wenn es keinen Pfad p im Fluchtwegnetz N vom Startknoten $v_0 = s$ zum Zielknoten $v_n = z$ existiert, ist der maximale Durchfluss $f_{\max}$ per Definition 0.

So ist beispielsweise der maximale Durchfluss $f_{\max}$ für das Fluchtwegenetz `ABC` aus dem Abb. 0.1a mit dem Startknoten `a` und dem Zielknoten ~~`b`~~`d` gleich 6 und für das Fluchtwegenetz `XYZ` aus Abb. 0.1b mit den Startknoten `a` und dem Zielknoten `f` gleich 19.



(a) Maximale Durchfluss von `a` nach `d` (b) Maximale Durchfluss von `a` nach `f`

Abbildung 0.2: Der maximale Durchfluss in den Fluchtwegenetzen aus Abb. 0.1

A.3 Algorithmus

Der Edmonds-Karp-Algorithmus[3] ist eine Implementierung des Ford-Fulkerson-Verfahrens[4], welches den maximalen Durchfluss in einem graphenbasierten Fluchtwegenetzwerk berechnet. Um die Obergrenze der maximalen Anzahl von Personen zu finden, die pro Minute in Sicherheit gebracht werden können, muss der Pseudocode in Algorithmus 1 in Java implementiert werden. Dabei sollte der Optimierungspfad p in Zeile 7 mit einer Breitensuche bestimmt werden. Der Durchfluss durch das Fluchtwegenetz nach jeder Runde im Algorithmus muss die drei Bedingungen in Abschnitt A.2 erfüllen.

Algorithm 1 Edmonds-Karp-Algorithmus

1: **procedure** MAXIMUMFLOW(N) ▷ Eingabe eines Fluchtwegenetz, siehe Abschnitt A.1
2: **for all** $(v, w) \in E$ **do**
3: $f(v, w) \leftarrow 0$ ▷ Der Durchfluss ist zunächst 0
4: **end for**
5: **repeat**
6: $G_f \leftarrow (V, E_f)$ ▷ Bestimmen des Restkapazitätennetzwerks, siehe Abschnitt A.2.1
7: $p \leftarrow$ BREITENSUCHE(G_f, s, z) ▷ Finden eines Optimierungspfads, siehe Abschnitt A.2.2
8: $k_{\min}(p) \leftarrow \min\{k_f(v, w) : (v, w) \in p\}$ ▷ Finden der geringsten Restkapazität
9: **for all** $(v, w) \in p$ **do**
10: $f(v, w) \leftarrow f(v, w) + k_{\min}(p)$ ▷ Durchfluss längs des Pfades wählen
11: $f(w, v) \leftarrow f(w, v) - k_{\min}(p)$ ▷ Durchfluss kann wieder zurückgeführt werden
12: **end for**
13: **until** $p \neq \emptyset$ ▷ Solange es einen Pfad p von s nach z in G_f gibt
14: **return** $f_{\max} = \sum_{(v,z)\in E} f(v, z)$ ▷ Maximaler Durchfluss $f_{\max}$ von s nach z
15: **end procedure**

[3] `https://de.wikipedia.org/wiki/Algorithmus_von_Edmonds_und_Karp`
[4] `https://de.wikipedia.org/wiki/Algorithmus_von_Ford_und_Fulkerson`

A.4 Verwaltung

Das zu implementierende Programm sollte die Verwaltung mit Eingabe und Speicherung sowie die Berechnung des maximalen Durchflusses $f_{\max}$ von Fluchtwegenetzen N ermöglichen. Dazu können zunächst die gerichteten Graphen G mit ihrer Kapazitätsfunktion k aus dem zugehörigen Fluchtwegenetz mit ihrem Bezeichner hinzugefügt werden. Dabei werden diese Graphen G als Menge ihrer einzelnen gewichteten $k(e)$ Fluchtwegeabschnitte e eingetragen, die wiederum durch die Kennungen der Knoten (v, w) bestimmt sind, und die Kapazitätsfunktion ist direkt durch die Kapazität des Fluchtwegeabschnitts gegeben. Das heißt, die Eingabe eines Fluchtwegenetzes besteht aus der textuellen Beschreibung der Fluchtwegeabschnitte mit ihrer zugehörigen Kapazität in der Form `<v><k><w>`. Dabei ist `<v>` der Bezeichner des ersten Knotens v, `<w>` ist der Bezeichner des zweiten Knotens w und `<k>` ist die Kapazität dieses Fluchtwegeabschnitts $k((v, w))$. Die Eingabe eines Graphen enthält einen oder mehrere Fluchtwegeabschnitte in beliebiger Reihenfolge, jeweils durch ein Semikolon getrennt und sind wie folgt aufgebaut: `<v_1><k_1><w_1>;<v_2><k_2><w_2>;...;<v_n><k_n><w_n>`

Das zu implementierende Programm soll den maximalen Durchfluss $f_{\max}$ eines Fluchtwegenetzes N explizit nur auf Aufforderung mit der vorgegebene Methode in Algorithmus 1 berechnen und diesen anschließend speichern. Im zu implementierende Programm werden immer nur nach den gegebene Spezifikationen formal gültige maximale Durchflüsse berechnet und gespeichert. Dabei werden für ein zuvor hinzugefügtes Fluchtwegenetz N, das durch seine Kennung bestimmt wird, die Kennungen der beiden Startknoten s und Zielknoten v angegeben, um die Berechnung durchzuführen. Wurde der maximale Durchfluss für einen Start- und Zielknoten bereits berechnet, wird dieser nicht erneut berechnet, sondern nur das bereits gespeicherte Ergebnis ausgegeben. Nach einer Änderung oder dem Hinzufügen eines Fluchtwegeabschnittes zu einem Fluchtwegenetz werden alle zuvor gespeichertes Ergebnis der maximalen Durchflüsse für dieses Fluchtwegenetz entfernt.

A.5 Interaktive Benutzerinteraktion

Nach dem Start nimmt das Programm zur Analyse von Fluchtwegenetzen über die Konsole Befehle unter Verwendung der Methode `readLine` der Klasse `Terminal` entgegen. Nach der Verarbeitung eines Befehls wartet das System auf weitere Eingaben, bis es durch Eingabe des `quit`-Befehls beendet wird. Die Eingabe und die Ausführung der Befehle dürfen nicht gegen die definierten Spezifikationen verstoßen. Bei einem Verstoß muss immer eine aussagekräftige Fehlermeldung ausgegeben werden, anschließend wartet das System regulär auf weitere Eingaben.

A.5.1 Symbole

`<n>` Die Kennung eines Fluchtwegenetzes, siehe Abschnitt A.1.

`<v>` Die Kennung eines Knotens, siehe Abschnitt A.1.1.

`<k>` Die Kapazität eines Fluchtwegeabschnitts, siehe Abschnitt A.1.3.

`<f>` Der maximale Durchfluss eines Fluchtwegenetzes, siehe Abschnitt A.2.3.

`<x>` Die Anzahl aller Knoten in einem Fluchtwegenetz, siehe Abschnitt A.1.1.

`<e>` Ein Fluchtwegeabschnitt mit seiner zugehörigen Kapazität in der Form `<v_1><k><v_2>`, siehe Abschnitt A.4

`<g>` Der Graph eines Fluchtwegenetzes bestehend aus einem oder mehrere jeweils durch ein Semikolon getrennten Fluchtwegeabschnitte, in der Form `<e_1>;<e_2>;...;<e_n>`, siehe Abschnitt A.4.

A.5.2 Befehle

Hinzufügen eines Fluchtwegenetz Dieser Befehl fügt für ein neues Fluchtwegenetz mit gegebener Kennung `<n>` einen gerichteten Graphen mit zugehöriger Kapazitätsfunktion hinzu. Bezeichner werden nur einmal vergeben und werden nicht überschrieben. Vor dem Hinzufügen muss überprüft werden, ob der Graph `<g>` die gegebenen formalen Anforderungen erfüllt, d.h. es muss mindestens eine Start- und Zielknotenpaarung geben, für welche die Berechnung des maximalen Durchflusses formal möglich ist.

Eingabe `add <n> <g>`

Ausgabe `Added new escape network with identifier <n>.`

Hinzufügen eines Fluchtwegeabschnitt Dieser Befehl fügt ein Fluchtwegeabschnitt mit seiner Kapazität `<e>` dem für das durch seine Kennung bestimmte bereits existierende Fluchtwegnetz `<n>` hinzu oder verändert die Kapazität eines bereits vorhandenen Fluchtwegeabschnitt. Bevor der Graph des Fluchtwegenetzes `<n>` verändert wird, muss ähnlich wie beim Hinzufügen eines neuen Fluchtwegenetzes geprüft werden, ob der daraus resultierende Graph noch den gegebenen formalen Anforderungen entspricht. Nach dem erfolgreichen Hinzufügen oder Abändern, werden die zuvor berechneten maximalen Durchflüsse für das Fluchtwegnetz `<n>` wieder entfernt.

Eingabe `add <n> <e>`

Ausgabe `Added new section <e> to escape network <n>.`

Auflisten der Fluchtwegenetze Dieser Befehl listet alle hinzugefügten Fluchtwegnetze auf, welche zeilenweise in dem angegebenen Format ausgegeben werden. Die Fluchtwegnetze werden zunächst absteigend nach der Anzahl ihrer Knoten `<x>` sortiert, d.h. die Fluchtwegnetze mit den meisten Knoten werden zuerst aufgelistet. Ist die Anzahl der Knoten identisch, werden die entsprechenden Fluchtwegnetze anschließend noch alphabetisch aufsteigend nach ihrer Kennung `<n>` sortiert, d.h. die lexikografisch kleinsten Werte kommen zuerst. Wenn noch kein Fluchtwegnetz hinzugefügt wurde, wird lediglich `EMPTY` ausgegeben.

Eingabe `list`

Ausgabe `<n> <x>`

Ausgabe eines Fluchtwegenetz Dieser Befehl listet alle Fluchtwegeabschnitte `<e>` für das durch seine Kennung bestimmte Fluchtwegnetz `<n>` zeilenweise im angegebenen Format auf. Diese gerichteten Fluchtwegeabschnitte werden zuerst aufsteigend nach der Kennung ihres ausgehenden Knotens v und dann aufsteigend nach der Kennung ihres eingehenden Knotens w sortiert.

Eingabe `print <n>`

Ausgabe `<e>`

Berechnung des maximalen Durchflusses Dieser Befehl startet die Berechnung des maximalen Durchflusses `<f>` für das durch seine Kennung bestimmte Fluchtwegnetz `<n>` mit den beiden angeforderten Startknoten `<v_1>` und Zielknoten `<v_2>` und gibt den Durchfluss dann im angegebenen Format aus. Vor dieser Berechnung muss überprüft werden, ob die Eingabe `<v_1> <v_2>` für das Fluchtwegnetz `<n>` den gegebenen formalen Anforderungen erfüllt. Wenn für einen Start- und Zielknoten in einem Fluchtwegenetzwerk bereits ein Ergebnis berechnet wurde, wird es nicht erneut berechnet und gespeichert, sondern das bereits gespeicherte Ergebnis wird ausgegeben.

Eingabe `flow <n> <v_1> <v_2>`

Ausgabe `<f>`

Auflisten aller Ergebnisse Dieser Befehl listet die zuvor berechneten maximalen Durchflüsse `<f>` für das durch seine Kennung bestimmte Fluchtwegnetz `<n>` im angegebenen Format auf. Diese Flüsse werden zunächst nach ihrem Wert aufsteigend sortiert. Wenn der maximale Durchfluss identisch ist, wird zunächst nach dem Bezeichner des Startknotens `<v_1>` und dann nach dem Bezeichner des Zielknotens `<v_2>` alphabetisch aufsteigend sortiert. Wenn für ein vorhandenes Fluchtwegnetz noch kein maximaler Durchfluss berechnet wurde, wird nur die Zeichenkette `EMPTY` ausgegeben.

Eingabe `list <n>`

Ausgabe `<f> <v_1> <v_2>`

Beenden des Programms Durch den parameterlosen Befehl beendet das Programm jederzeit komplett. Wenn dieser Befehl erfolgreich ist, erfolgt keine Ausgabe.

Eingabe `quit`

A.6 Beispielinteraktion

Die Zeilennummern und die Trennlinie sind kein Bestandteil der Benutzerschnittstelle, sie dienen lediglich zur Orientierung für die gegebene Beispielinteraktion. Die Eingabezeilen werden mit dem Größer-als-Zeichen gefolgt von einem Leerzeichen eingeleitet, diese beiden Zeichen sind ebenfalls kein Bestandteil des eingegebenen Befehls, sondern dienen der Unterscheidung zwischen Ein- und Ausgabezeilen.

Beispielinteraktion

```
> add ABC a4b;a2c;b1d;b3c;c6d
Added new escape network with identifier ABC.
> list
ABC 4
> print ABC
a4b
a2c
b3c
b1d
c6d
> flow ABC a d
6
> list ABC
6 a d
> add ABC a5d
Added new section a5d to escape network ABC.
> print ABC
a4b
a2c
a5d
b3c
b1d
c6d
> list ABC
EMPTY
> add XYZ a10b;a10c;b2c;b4d;b8e;c9e;d10f;e10f;e6d
Added new escape network with identifier XYZ.
> list
XYZ 6
ABC 4
> print XYZ
a10b
a10c
b2c
b4d
b8e
c9e
d10f
e6d
e10f
> flow XYZ a f
19
> list XYZ
19 a f
> quit
```