Entwicklerdokumentation zur praktischen Arbeit der IHK-Prüfung 2023

Julian Schönau

10.05.2023

Inhalte:

KAPITEL 1

MinimalRechner.py

MinimalRechner.**__init__**(*pfad: str*) → None

Konstruktor zur Erzeugung einer MinimalRechner-Klasseninstanz.

Deklariert Klassenvariablen zur späteren Verwendung und initialisiert den IOManager mit den übergebenen Parametern.

Parameter
pfad (*str*) – Pfad zur Eingabedatei

Rückgabe
None

Rückgabetyp
None

class src.MinimalRechner.**MinimalRechner**(*pfad: str*)

MinimalRechner-Klasse

Als „Verwaltungsinstanz“ gestalten sich die Aufgaben der MinimalRechner-Klasse manigfaltig:

- sie stellt einen Sammelbehaelter fuer alle anderen **Komponenten** dar.
- sie ueberwacht die ordnungsgemaesse Abarbeitung des Algorithmus.
- sie liefert die Schnittstelle zum Lesen und Schreiben der Ein- und Ausgabedatei.
- sie speichert die Ergebnisse der Berechnung zum spaeteren Abruf.

Die Parameter, die in einer Instanz dieser Klasse gespeichert werden, beziehen sich demnach auf Folgendes:

- die Schnittstelle zu den Ein- und Ausgabefunktionen des IOManagers (**manager**)
- die Zugverbindungen, die aus der Eingabedatei eingelesen werden (**zugverbindungen**)
- die errechnete Minimalloesung nach Anwenden des Algorithmus (**minimalloesung**)

Instanzen dieser Klasse bieten folgende Funktionalitaet an:

- Initialisieren aller Parameter zum bei Erstellen der Klasseninstanz (**__init__**)

- Optimieren der Zugverbindungen durch Anwenden der Reduktionstechniken (**reduziereZugverbindungen**)
- Bestimmen der minimalen Anzahl an sowie Positionen der Servicestationen durch Anwenden des Algorithmus (**berechneMinimalloesung**)
- Validieren des im Algorithmus errechneten Ergebnisses (**validiereMinimalloesung**)

berechneKandidaten(*reduktionstechniken: list[*src.Reduktionstechnik.Reduktionstechnik*] = []*) → list[*src.Knoten.Knoten*]

Funktion zur Berechnung der moeglichen Kandidaten für die Minimalloesung durch Bestimmen der Haltestellen mit maximaler Überdeckung.

Der Algorithmus bestimmt die Kandidaten, indem…

…zunächst die Zugverbindungen durch Anwendung der Reduktionstechniken optimiert werden.

…die totalen Aufkommen aller Haltestellen in den optimierten Zugverbindungen berechnet werden.

…pro Iterationsschritt die Haltestelle - die in den meisten Zugverbindungen auftritt - als Servicestation bestimmt wird.

…alle Zugverbindungen - die durch eine Servicestation abgedeckt sind - entfernt werden.

…die Haltestellen - die als Servicestation bestimmt wurden - entfernt werden.

Rückgabe
iterativ erzeugte Liste der Kandidaten

Rückgabetyp
list[*Knoten*]

berechneMinimalloesung(*zugverbindungen: list[list[src.Knoten.Knoten]] | None = None, ausgabeOrdner: str | None = None, reduktionstechniken: list[src.Reduktionstechnik.Reduktionstechnik] = [<class 'src.Reduktionstechnik.ReduziereDuplikate'>, <class 'src.Reduktionstechnik.ReduziereBahnhoefe'>, <class 'src.Reduktionstechnik.ReduziereZugverbindungen'>]*) → tuple[int, list[*src.Knoten.Knoten*]]

Funktion zur Berechnung der minimalen Anzahl an sowie Positionen der Servicestationen durch Bestimmen der Haltestellen mit maximaler Überdeckung.

Der Algorithmus bestimmt die minimale Lösung, indem…

…die möglichen Kandidaten zunächst durch Anwendung eines Greedy-Algorithmus bestimmt werden.

…die mit einem Kandidaten assoziierten Zugverbindungen - nach Haltestellennahmen kategorisiert - gespeichert werden.

…die minimale Anzahl an Zugverbindungen bestimmt wird, die das gesamte Eisenbahnnetz abdeckt.

…die endgültige Lösung anhand ihrer Mächtigkeit auf Plausibilität getestet wird.

Bei Übergabe einer Liste von Zugverbindungen wird diese als Grundlage für die Berechnung der minimalen Lösung verwendet. Durch Angabe eines Ausgabeordners wird die Minimallösung an einen benutzerdefinierten Speicherort geschrieben.

Optionale Parameter:

Parameter

- **zugverbindungen** (*list[list[*Knoten*]]*) – Liste der Zugverbindungen
- **ausgabeOrdner** (*str*) – Pfad zum Ausgabeordner

Rückgabe
Anzahl und Positionen der Servicestationen

Rückgabetyp
tuple[int, list[*Knoten*]]

erzeugeKombinationen(*kandidaten: list[*src.Knoten.Knoten*]*) → list[list[*src.Knoten.Knoten*]]

Hilfsfunktion zur Erzeugung aller Kombinationen einer Liste von Kandidaten.

Erforderliche Parameter:

Parameter
kandidaten (*list[*Knoten*]*) – Liste der Kandidaten

Rückgabe
Liste aller Kombinationen - nach aufsteigender Länge sortiert

Rückgabetyp
list[list[*Knoten*]]

minimaleAbdeckung(*alleVerbindungen: set[frozenset[*src.Knoten.Knoten*]]*, *zuKombinierendeVerbindungen: dict[*src.Knoten.Knoten*, set[frozenset[*src.Knoten.Knoten*]]]*) → list[*src.Knoten.Knoten*]

Hilfsfunktion zur Bestimmung der minimal benötigten Kandidaten, um das Eisenbahnnetz vollständig abzudecken.

Erforderliche Parameter:

Parameter

- **alleVerbindungen** (*set[frozenset[*Knoten*]]*) – Menge aller Zugverbindungen
- **zuKombinierendeVerbindungen** (*dict[*Knoten, *set[frozenset[*Knoten*]]]*) – Menge aller Zugverbindungen, die durch einen Kandidaten abgedeckt werden

Rückgabe
Liste der minimal benötigten Kandidaten

Rückgabetyp
list[*Knoten*]

reduziereZugverbindungen(*reduktionstechniken: list[*src.Reduktionstechnik.Reduktionstechnik*]*) → list[list[*src.Knoten.Knoten*]]

Vorverarbeitungsschritt zur Optimierung der Zugverbindungen nach den gegebenen Reduzierungstechniken.

Rückgabe
Liste der optimierten Zugverbindungen

Rückgabetyp
list[list[*Knoten*]]

validiereErgebnis(*ergebnis: list[*src.Knoten.Knoten*]*) → bool

Hilfsfunktion zur Validierung des Ergebnisses durch Ausschluss unmöglicher Mächtigkeiten.

Erforderliche Parameter:

Parameter
ergebnis (*list[*Knoten*]*) – Liste der Servicestationen

Rückgabe
True, wenn Ergebnis valide ist, sonst False

Rückgabetyp
bool

KAPITEL 2

IOManager.py

IOManager.__init__() → None

Konstruktor zur Erzeugung einer IOManager-Klasseninstanz.

Deklariert einen vorläufigen Dateinamen zur späteren Benennung der Ausgabedatei.

Rückgabe
None

Rückgabetyp
None

class src.IOManager.**IOManager**

IOManager-Klasse

Als Schnittstelle zwischen Ein-/ Ausgabedatei und dem Programm stellt die IOManager-Klasse folgende Funktionalitaet bereit:

- Lesen der Eingabedatei und Umwandeln der einzelnen Textzeilen in eine Liste von Listen von Knoten-Objekte (**leseDatei**)
- Umwandeln einer Zeichenkette in eine Liste von Knoten-Objekten (**verbindungZuKnoten**)
- Schreiben der Ausgabedatei und Umwandeln der Liste von Knoten-Objekten in eine Textzeile (**schreibeAusgabe**)
- Umwandeln einer Liste von Knoten-Objekten in eine Zeichenkette (**knotenlisteZuText**)

Die Parameter, die in einer Instanz dieser Klassen gespeichert werden, beziehen sich auf Folgendes:

- der Name der Eingabedatei (**fname**)

knotenlisteZuText(*menge: list[*src.Knoten.Knoten*]*) → str

Funktion zum Umwandeln einer Liste von Knoten-Objekten in eine semikolonseparierte Zeichenkette.

Erforderliche Parameter:

Parameter
menge (*list[*Knoten*]*) – Liste von Knoten

Rückgabe
Eine semikolonseparierte Zeichenkette

Rückgabetyp
str

`leseDatei`(*pfad: str*) → list[list[*src.Knoten.Knoten*]]

Funktion zum Einlesen und Umwandeln der Eingabedaten in geeignete Datenstrukturen.

Liest die Eingabedatei und wandelt die einzelnen Textzeilen in eine Liste von Listen von Knoten-Objekte um, falls diese nur Gross-, Kleinbuchstaben und Semikolons enthalten. Der Name der Eingabedatei wird zur späteren Benennung der Ausgabedatei als Klassenvariable gespeichert.

Erforderliche Parameter:

Parameter
pfad (*str*) – Pfad zur Eingabedatei

Rückgabe
Eine Liste von Zugverbindungen

Rückgabetyp
list[list[*Knoten*]]

`schreibeAusgabe`(*menge: list[*src.Knoten.Knoten*]*, *ausgabeOrdner: str | None = None*) → None

Funktion zur Ausgabe der identifizierten Minimalloesung gemaess Aufgabenstellung.

Schreibt die Ausgabedatei und wandelt die Liste von Knoten-Objekten in eine Textzeile um.

Erforderliche Parameter:

Parameter

- **menge** (*list[Knoten]*) – Liste von Bahnhofsnamen, die als Servicestationen identifiziert wurden
- **ausgabeOrdner** (*str*) – Ordner, in dem die Ausgabedatei gespeichert werden soll

Rückgabe
None

Rückgabetyp
None

`verbindungZuKnoten`(*verbindung: list[str]*) → list[*src.Knoten.Knoten*]

Funktion zum Umwandeln einer Liste von Bahnhofnamen in eine Liste von Knoten-Objekten.

Erforderliche Parameter:

Parameter
verbindung (*list[str]*) – Liste von Bahnhofnamen

Rückgabe
Eine Liste von Zugverbindungen

Rückgabetyp
list[*Knoten*]

KAPITEL 3

Reduktionstechnik.py

class src.Reduktionstechnik.**Reduktionstechnik**

Bases: object

Reduktionstechnik-Klasse

Die Reduktionstechnik-Klasse ist eine Abstrakte Basisklasse.

Implementierungen dieser Klasse beschreiben eine Reduktionstechnik zur Minimierung der Eingabedaten.

abstract reduziere() → list[list[*src.Knoten.Knoten*]]

Abstrakte Funktion zur Reduktion der uebergebenen Liste von Zugverbindungen nach einer beliebigen Technik.

Parameter
k_list (*list[list[Knoten]]*) – Liste von Zugverbindungen

Rückgabe
reduzierte Liste von Zugverbindungen

Rückgabetyp
list[list[*Knoten*]]

class src.Reduktionstechnik.**ReduziereBahnhoefe**

Bases: *Reduktionstechnik*

ReduziereBahnhoefe-Klasse

Die ReduziereBahnhoefe-Klasse ist eine konkrete Implementierung der Reduktionstechnik-Klasse.

Sie minimiert die Eingabedaten durch Entfernen eines Bahnhofes der Bahnhoefe, die ausschliesslich in Paaren auftauchen.

reduziere()

Funktion zur Minimierung der Zugverbindungen durch Entfernen von Bahnhoefen, die ausschliesslich in Paaren auftauchen.

Unter Zuhilfenahme der **Kanten**-Klasse wird modelliert, wie oft eine Verbindung zwischen zwei Bahnhoefen besteht. Dieser Wert wird als Gewicht in der Kante gespeichert. Fuer jede Kante, deren Gewicht

der Anzahl eines der beteiligten Bahnhoefe entspricht, wird der erste Bahnhof nicht in die reduzierte Liste uebernommen.

Parameter
k_lists (*list[list[Knoten]]*) – Liste von Zugverbindungen

Rückgabe
reduzierte Liste von Zugverbindungen

Rückgabetyp
list[list[*Knoten*]]

class src.Reduktionstechnik.**ReduziereDuplikate**

Bases: *Reduktionstechnik*

ReduziereDuplikate-Klasse

Die ReduziereDuplikate-Klasse ist eine konkrete Implementierung der Reduktionstechnik-Klasse.

Sie minimiert die Eingabedaten durch Entfernen der Duplikate aus den Zugverbindungen.

reduziere()

Funktion zur Minimierung der Zugverbindungen durch Entfernen multipler Haltestellen.

Fuer jede Zugverbindung wird ueberprueft, ob alle Knoten genau eindeutig sind. Knoten, die mehrfach auftauchen, werden nicht in die reduzierte Liste uebernommen.

Parameter
k_lists (*list[list[Knoten]]*) – Liste von Zugverbindungen

Rückgabe
reduzierte Liste von Zugverbindungen

Rückgabetyp
list[list[*Knoten*]]

class src.Reduktionstechnik.**ReduziereZugverbindungen**

Bases: *Reduktionstechnik*

ReduziereZugverbindungen-Klasse

Die ReduziereZugverbindungen-Klasse ist eine konkrete Implementierung der Reduktionstechnik-Klasse.

Sie minimiert die Eingabedaten durch Entfernen aller Zugverbindungen, deren Teil- oder Gesamtstrecke bereits in anderen Zugverbindungen modelliert wird.

reduziere()

Funktion zur Minimierung der Zugverbindungen durch Entfernen von nicht-minimierten Zugverbindungen.

Durch Konvertieren der einzelnen Zugverbindungen in Strings kann für jede Zugverbindung geprüft werden, ob sie Teil einer anderen Zugverbindung ist. Ist dies der Fall, wird die längere Zugverbindung aus der Liste entfernt.

Parameter
k_lists (*list[list[Knoten]]*) – Liste von Zugverbindungen

Rückgabe
reduzierte Liste von Zugverbindungen

Rückgabetyp
list[list[*Knoten*]]

KAPITEL 4

Kante.py

Kante.__init__(*k1:* Knoten, *k2:* Knoten) → None

Konstruktor zur Erzeugung einer Kante-Klasseninstanz.

Das Gewicht der Kante wird mit 0 initialisiert.

Erforderliche Parameter:

Parameter

- **k1** (Knoten) – Erster Knoten, den diese Kante verbindet
- **k2** (Knoten) – Zweiter Knoten, den diese Kante verbindet

Rückgabe
None

Rückgabetyp
None

Kante.__eq__(*k:* Kante) → bool

Operatorueberladung, die es ermoeglicht, zwei Kanten anhand der Punkte, die sie verbinden, zu vergleichen.

Erforderliche Parameter:

Parameter
k (Kante) – Kante, mit dem diese Kante verglichen werden soll

Rückgabe
True, wenn die Kanten die gleichen Punkte verbinden, sonst False

Rückgabetyp
bool

Kante.__hash__() → int

Operatorueberladung, die es ermoeglicht, die Kante zu hashen und vergleichbar zu machen. Kanten, die dieselben Punkte verbinden, sollen dieselbe Kante sein und intern denselben Hash-Wert besitzen.

Rückgabe
Hash-Wert der Kante

Rückgabetyp
int

class src.Kante.**Kante**(*k1:* Knoten, *k2:* Knoten)

Kante-Klasse

Die Kante-Klasse beschreibt eine Kante im Graphen, die zwei Knoten miteinander verbindet.

Die Parameter, die in einer Instanz dieser Klasse gespeichert werden, beziehen sich auf die beiden Knoten, die miteinander verbunden werden (**k1**, **k2**) sowie die Häufigkeit der Verbindung (**gewicht**) im Eisenbahnnetz.

Instanzen dieser Klasse beiten folgende Funktionalitaet an: * Initialisieren aller Parameter zum Erstellen der Klasseninstanz (**__init__**) * Vergleich zweier Kanten anhand der Knoten, die sie verbinden (**__eq__**) * Hashing des Knotens anhand der Knoten, die sie verbinden (**__hash__**) * Setzen des Vorgaengers/ Nachfolgers (**setzeVorgaenger**, **setzeNachfolger**) * Rueckgabe der Knoten, die diese Kante miteinander verbindet (**gibKnoten**) * Aktualisieren und Abrufen der Häufigkeit der Verbindung (**erhoeheGewicht**, **setzeGewicht**, **gibGewicht**)

erhoeheGewicht() → None

Funktion zur Inkrementierung des Gewichts der Kante um 1.

Rückgabe
None

Rückgabetyp
None

gibGewicht() → int

Funktion zur Rueckgabe des Gewichts der Kante.

Rückgabe
Gewicht der Kante

Rückgabetyp
int

gibKnoten() → list[*src.Knoten.Knoten*]

Funktion zur Rueckgabe der Knoten, die diese Kante miteinander verbindet.

Rückgabe
Liste der beiden Knoten, die diese Kante miteinander verbindet

Rückgabetyp
list[*Knoten*]

setzeGewicht(*gewicht: int*) → None

Funktion zur Setzung des Gewichts der Kante auf einen beliebigen Wert.

Erforderliche Parameter:

Parameter
gewicht (*int*) – Neuer Wert fuer das Gewicht der Kante

Rückgabe
None

Rückgabetyp
None

KAPITEL 5

Knoten.py

Knoten.**__init__**(*name: str*) → None

Konstruktor zur Erzeugung einer Knoten-Klasseninstanz.

Erforderliche Parameter:

Parameter
name (*str*) – Name des Bahnhofes, den dieser Knoten modelliert

Rückgabe
None

Rückgabetyp
None

Knoten.**__str__**() → str

Operatorueberladung, die es ermoeglicht, den Namen des Knotens als Zeichenkette auszugeben.

Rückgabe
Name des Knotens

Rückgabetyp
str

Knoten.**__repr__**() → str

Operatorueberladung, die den Namen des Knotens zur Repräsentation bei jeder Ausgabe verwendet.

Rückgabe
Name des Knotens

Rückgabetyp
str

Knoten.**__eq__**(*other:* Knoten) → bool

Operatorueberladung, die es ermoeglicht, zwei Knoten anhand ihres Namens zu vergleichen.

Erforderliche Parameter:

Parameter
other (Knoten) – Knoten, mit dem dieser Knoten verglichen werden soll

Rückgabe
True, wenn die Namen der beiden Knoten gleich sind, sonst False

Rückgabetyp
bool

Knoten.__hash__() → int

Operatorueberladung, die es ermoeglicht, den Knoten zu hashen und vergleichbar zu machen. Knoten, die denselben Bahnhof beschreiben, sollen derselbe Knoten sein und intern denselben Hash-Wert besitzen.

Rückgabe
Hash-Wert des Knotens

Rückgabetyp
int

class src.Knoten.**Knoten**(*name: str*)

Knoten-Klasse

Die Knoten-Klasse beschreibt einen Knoten im Graphen und entspricht dabei einem Bahnhof in einer Zugverbindung.

Die Parameter, die in einer Instanz dieser Klasse gespeichert werden, beziehen sich auf den Namen (**name**) des Bahnhofes sowie auf den vorangehenden (**vorgaenger**) und nachfolgenden (**nachfolger**) Knoten.

Instanzen dieser Klasse beiten folgende Funktionalitaet an: * Initialisieren aller Parameter zum Erstellen der Klasseninstanz (**__init__**) * Ausgabe des Knotens in menschen-leserlicher Form bei expliziter Konvertierung in einen String (**__str__**) * Ausgabe des Knotens in menschen-leserlicher Form bei normaler Ausgabe (**__repr__**) * Vergleich zweier Knoten anhand ihres Namens (**__eq__**) * Hashing des Knotens anhand seines Namens (**__hash__**) * Setzen des Vorgaengers/ Nachfolgers (**setzeVorgaenger**, **setzeNachfolger**) * Abrufen des Vorgaengers/ Nachfolgers (**gibVorgaenger**, **gibNachfolger**)

gibNachfolger() → *Knoten*

Funktion zum Abrufen des Nachfolgers des Knotens.

Rückgabe
Nachfolger des Knotens

Rückgabetyp
Knoten

gibVorgaenger() → *Knoten*

Funktion zum Abrufen des Vorgaengers des Knotens.

Rückgabe
Vorgaenger des Knotens

Rückgabetyp
Knoten

setzeNachfolger(*k:* Knoten) → None

Funktion zur Festlegung des Nachfolgers des Knotens.

Erforderliche Parameter:

Parameter
k (Knoten) – Knoten, der als Nachfolger festgelegt werden soll

Rückgabe
None

Rückgabetyp
None

`setzeVorgaenger`(*k:* Knoten) → None

Funktion zur Festlegung des Vorgaengers des Knotens.

Erforderliche Parameter:

Parameter
k (Knoten) – Knoten, der als Vorgaenger festgelegt werden soll

Rückgabe
None

Rückgabetyp
None

Python-Modulindex

S

Stichwortverzeichnis

S

V