Hochschule Fulda
University of Applied Sciences

Prof. Dr. Thomas Wiemann

Algorithmen und Datenstrukturen
Übungsblatt 4

Wintersemester 2023/24

Aufgabe 4.1 (Implementierung eines Sparse-Vektors – Testat)

Bisher haben wir Arrays immer in einer dichten (engl. *dense*) Repräsentation gespeichert. Behandelt man jedoch Daten, die viele Nullen enthalten, kann eine dünn-besetzte (engl. *sparse*) Repräsentation unter Umständen die bessere Wahl bzgl. Speicherverbrauch sein. In dieser Aufgabe implementieren Sie den in der Vorlesung besprochene Sparse-Repräsentation mittels einfach-verketteter Liste. Schreiben Sie dazu eine Java-Klasse `SparseVector` für `double`-Werte, die folgende Funktionalitäten implementiert:

- Einen Standard-Konstruktor, der einen leeren Vektor initialisiert.
- Einen Konstruktor, der einen Vektor der Länge `n` erzeugt.
- Eine Methode `void setElement(int index,, double value), die den Wert value an der Stelle index in den Vektor einfügt.`
- Eine Methode `double getElement(int index)` das den Wert des durch `index` definierten Eintrags wiedergibt (oder `0.0`, falls der Eintrag an dieser Stelle nicht existiert).
- Eine Methode `void removeElement(int index)`, das den Eintrag an der durch `index` definierten Stelle entfernt.
- Eine Methode `int getLength()`, die die Länge des Vektors zurückgibt.
- Eine Methode `bool equals(SparseVector other), die testet, ob der übergebene SparseVector other mit der aktuellen Instanz identisch ist.`
- Eine Methode `void add(SparseVector other), die alle Einträge aus dem Vektor other auf die aktuelle Repräsentation addiert.`

Zur Verwaltung der Einträge soll der Sparse-Vektor intern eine von Ihnen implementierte einfach-verkettete Liste wie in der Vorlesung vorgestellt verwenden. Dazu können Sie eine Hilfsklasse `Node` verwenden, die die folgenden Elemente enthält:

- Ein Feld `value` vom Typ `double`, das den Wert des Knotens hält.
- Ein Feld `index` vom Typ `int`, das den Index des Knotens enthält.
- Ein Feld `next`, das einen Verweis auf den nächsten Eintrag in der Liste enthält.

Beachten Sie, dass die Elemente in der Liste immer nach Index sortiert sein müssen! Zu Implementierung der oben genannten Funktionalitäten bietet es an, sich Hilfsfunktionen, z.B. zum setzen von Nicht-Null-Elementen zu implementieren. Lösen Sie bei fehlerhaften Eingaben geeignete Exceptions aus.

Aufgabe 4.2 (Testen der Implementierung – Testat)

Schreiben Sie eine Reihe von Testfunktionen, die die oben genannten Funktionalitäten testen. Prüfen Sie dabei automatisiert anhand verschiedener Szenarien, ob alle Funktionen des Sparse-Vektors wie gewünscht funktionieren. Erklären Sie Ihrem Übungsleiter Ihre Tests und machen Sie plausibel, dass diese alle genannten Anforderungen abdecken.

Abgabe Ihre Abgabe wird in den Übungen in KW 48 testiert. Die Testate erfolgen in 4er-Gruppen. Zur Abgabe laden Sie dazu pro Person bis zum 27.11.2023, 8:00 Uhr eine Datei `SparseVector.java` sowie eine Datei `CheckSparseVector.java` mit Ihren Abgaben über die bereitgestellte Abgabefunktion in Moodle hoch.