מעבדה 4. נושא: מטריצות דלילות

תאריך הגשה: 29.11.2022 בשעה 23:00 (בזוגות)

יש לקרוא היטב לפני תחילת העבודה !

מבוא:

במעבדה זו נלמד סוג נוסף של מטריצות מיוחדות – מטריצות דלילות ונראה כיצד ניתן לממש אותן בעזרת רשימות מקושרות.

תיאור:

מטריצה ריבועית n*n בה רוב האיברים שווים לאפס, נקראת *מטריצה דלילה*. למשל:

0	1	0	4
0	0	0	0
0	0	0	0
0	6	0	0

היא מטריצה דלילה בגודל 4*4 .במטריצה זאת מספר האיברים השונים מאפס שווה ל-3.
מקובל להגדיר מטריצה דלילה ע"י כך שמספר האיברים השונים מאפס **r** הרבה יותר קטן מ **n*n**, כלומר **r<< n*n**
גם מטריצה כזאת ניתן לראות כמטריצה דלילה:

3	1	3	4
3	3	3	3
3	3	3	3
3	6	3	3

במקרה זה רוב האיברים של המטריצה הנ"ל שווים ל – 3. ויש מעט איברים השונים מ – 3.

1). **מימוש סטנדרטי** , באמצעות מערך דו-מימדי, מאפשר ביצוע פעולות בסיבוכיות:

- $get(i,j)$ ב- $O(1)$
- $set(i,j,x)$ ב- $O(1)$.
- אך סיבוכיות המקום הינה גבוהה $O(n * n)$.

נציע מימוש חלופי באמצעות **רשימה מקושרת:**

לכל איבר השונה מהאיבר הנפוץ (0 או 3 בדוגמאות הנ"ל) נחזיק צומת בעל 3 שדות: אינדקס i, אינדקס j, וערך האיבר. כמובן, נוסיף עוד משתנה שיחזיק את הערך של שאר האברים.

לדוגמא, עבור המטריצות בדוגמה לעיל, ניתן להחזיק את הרשימה המקושרת הבאה:

$[1, 2, 1] \rightarrow [1, 4, 4] \rightarrow [4, 2, 6]$

במימוש זה, **<u>הסיבוכיות</u>** תהיה כדלקמן:

–סיבוכיות מקום: $O(r)$

–סיבוכיות $get(i,j)$ היא $O(r)$ (במקרה הגרוע יש לעבור על כל הרשימה)

–סיבוכיות $set(i,j,x)$ היא $O(r)$ (יש לבדוק האם האיבר כבר קיים ברשימה)

המימוש המוצע חוסך הרבה מקום.

1) כתבו מחלקה בשם **SparseMatrix<T>** הממשת את הממשק Matrix<T> הנתון לכם (שימו לב שגם transpose צריך להיות בסיבוכיות זמן קבועה), ומשתמשת במחלקה הגנרית DLinkedList<T> שכתבתם במעבדה קודמת. (העתיקו את המחלקה ממעבדה קודמת לפרוייקט המעבדה הזו).

עם הבנאים:

SparseMatrix (T defaultValue);

SparseMatrix (int size, T defaultValue);

הבנאי הראשון בונה מטריצה שבה האיבר המצוי הוא defaultValue, וגודל המטריצה (מספר השורות) שווה לקבוע הנתון ב-Matrix.

הבנאי השני בונה מטריצה size×size שבה האיבר המצוי הוא defaultValue.

בתוך המחלקה SparseMatrix<T>, הגדירו מחלקה פרטית SparseMatrixEntry כדלהלן:

```
private class SparseMatrixEntry {
        private T value;
        private int row;
        private int col;

        public SparseMatrixEntry(int row, int col, T val) { ... }

        public int getRow() { ... }
        public void setRow(int row) { ... }
        public int getCol() { ...  }
        public void setCol(int col) { … }
        public T getValue() { … }
        public void setValue(T newVal) { … }
}
```

2) נתונה לכם מחלקת בדיקה Junit 4 בשם MatrixTest הבודקת מימוש של Matrix<Integer>.
השלימו את המחלקה MatrixTest הנתונה לכם ע״י הוספת בדיקות.
השתמשו ב-MatrixFactory<T> על מנת לקבל מופע של Matrix<Integer>. אין לקבל מופע של Matrix<Integer> בדרך אחרת.

סדר העבודה ופרטים טכניים

- שליפת הפרויקט **DS-Lab04-SparseMatrix** מתוך GITHUB:
https://github.com/ykanizo/DSLab2022-2023Public
השתמשו ב File->Import->Git->CloneURI בתוך התפריט של Eclipse,
או שהעתיקו את הקבצים שיש שם לתיקיית ה-src של הפרוייקט שלכם (שימו לב לא לשים את הקבצים בתוך package אחר מלבד ה-default).

אם אתם עובדים ב VDI, מומלץ לשנות את המיקום המוצע לפרויקט בתיקייה כלשהי בכונן H.

- הוסיפו את המחלקה DLinkedList<T> שמימשתם במעבדה קודמת.
- הוסיפו את המחלקה SparseMatrix<T>.
- השלימו את המחלקה MatrixTest.
- וודאו כי הרצת MatrixTest עוברת בהצלחה.
- יש להגיש את קבצי ה-java.

<u>פורמט קובץ ההגשה ובדיקתו:</u>

פורמט: יש להגיש קובץ ZIP בשם

43_lab04_123456789_987654321.zip

(כמובן, יש להחליף את המספרים עם מספרי ת.ז. של המגישים).

על הקובץ להכיל את כל קבצי ה JAVA שכתבתם. שימו לב: הקובץ לא יכיל את התיקיה שבה הקבצים נמצאים, רק את הקבצים עצמם (אם לא ברור מה ההבדל, ראו סרטון הדגמה מטה).

שימו לב כי קובץ ה-zip שאתם מגישים מכיל את הקובץ עם הטבלאות!

<u>בדיקת קובץ ההגשה:</u> בדקו את הקובץ שיצרתם בתוכנת הבדיקה בקישור:

https://csweb.telhai.ac.il/

ראו סרטון הדגמה של השימוש בתוכנת הבדיקה.

<u>הסבר על תוצאות הבדיקה האוטומטית בתרגיל זה:</u>

במעבדה זו, יש מספר סוגי בדיקות, וזאת במטרה שתוכלו לבדוק את עצמכם היטב לפני ההגשה.
לכל test הוספנו שם ותיאור המסבירים מה הוא בודק. יש 3 סוגים של tests:

1. tests שבודקים את המחלקה SparseMatrix<T> שמימשתם, עבור T=Integer.
עבור טסטים אלו תקבלו 30 נקודות בבודק האוטומטי
2. tests הבודקים את המחלקה MatrixTest שרשמתם, עבור המימוש שלכם ל SparseMatrix<T>.
עבור טסטים אלו תקבלו 15 נקודות.
3. tests הבודקים את המחלקה MatrixTest שרשמתם,
עבור מימוש שגוי של SparseMatrix<T>. המטרה במקרים אלו, שהבדיקות שלכם יתפסו את השגיאה.

הטסטים מסוג 3 מגלים בעיות חוסר שלמות בבדיקות שאתם רשמתם. בעיות אלו מאד קשות לזיהוי ולפיכך הרבה יותר קשה יהיה לתקן בעיות כאלו.
יש שני סטים של בדיקות כאלו (עבור 2 מימושים שגויים של SparseMatrix),
סך הכל 15 נק (7 לסט הראשון ו-8 לסט השני). הנקודות ינתנו במלואן עבור סט של בדיקות אם לפחות אחת מהבדיקות בסט נכשלה.

חשוב !!!
בדיקת ההגשות תבוצע ברובה ע"י תוכנית הבדיקה האוטומטית הנ"ל. תוצאת הבדיקה תהייה בעיקרון זהה לתוצאת הבדיקה הנ"ל שאתם אמורים לערוך בעצמכם . כלומר, אם ביצעתם את הבדיקה באתר החוג, לא תקבלו הפתעות בדיעבד. אחרת, ייתכן שתרגיל שעבדתם עליו קשה ייפסל בגלל פורמט הגשה שגוי וכו'. דבר שהיה ניתן לתקנו בקלות אם הייתם מבצעים את הבדיקה.
היות ואין הפתעות בדיעבד, לא תינתן אפשרות של תיקונים, הגשות חוזרות וכד'.

הגשה שלא מגיעה לשלב הקומפילציה תקבל ציון 0.
הגשה שלא מתקמפלת תקבל ציון נמוך מ- 40 לפי סוג הבעיה.
הגשה שמתקמפלת תקבל ציון 40 ומעלה בהתאם לתוצאות הריצה, ותוצאת הבדיקה הידנית של הקוד (חוץ ממקרה של העתקה).

תכנית הבדיקה האוטומטית מכילה תוכנה חכמה המגלה העתקות. מקרים של העתקות יטופלו בחומרה

עבודה נעימה!