

מעבדה 4. נושא: מטריצות דלילות

תאריך הגשה: 23.11.2021 (בזוגות) בשעה 23:00

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 $n \times n$ 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 $n \times n$, כלומר $r \ll n \times n$

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:

- $get(i, j)$ - $O(1)$

- $set(i, j, x)$ - $O(1)$

- אך סיבוכיות המקום הינה גבוהה $O(n * n)$.

רשימה מקושרת: נציע מימוש חלופי באמצעות

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, **הסיבוכיות** 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>** המממשת את הממשק **Matrix<T>** הנתון לכם, 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** גנרית בשם **MatrixTest<T>** הבודקת מימושים של **Matrix<T>**. השלימו את המחלקה **MatrixTest<T>** הנתונה לכם ע"י הוספת בדיקות. השתמשו ב-**MatrixFactory<T>** על מנת לקבל מופע של **Matrix<T>**. אין לקבל מופע של **Matrix<T>** בדרך אחרת. שימו לב כי **MatrixTest** היא מחלקה אבסטרקטית המכילה מתודה אבסטרקטית **getParameterInstance**.

(3) הוסיפו מחלקה **MatrixTestString** היוורשת מ- **MatrixTest<String>**, ומחלקה **MatrixTestInteger** היוורשת מ- **MatrixTest<Integer>**. ממשו את כל הטסטים במחלקת האב **MatrixTest<T>**. במחלקות **MatrixTestString** ו-**MatrixTestInteger** ממשו רק את המתודה **getParameterInstance** על פי הדרישות בתרגיל.

סדר העבודה ופרטים טכניים

- שליפת הפרויקט **DS-Lab04-SparseMatrix** מתוך GITHUB: <https://github.com/michalHorovitz/DSLAb2021-2022Public>
השתמשו ב File->Import->Git->CloneURI בתוך התפריט של Eclipse, או שהעתיקו את הקבצים שיש שם לתיקיית ה-src של הפרויקט 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 בכוון H.
 - הוסיפו את המחלקה `DLinkedList<T>` שמימשות במעבדה קודמת.
 - הוסיפו את המחלקה `SparseMatrix<T>`.
 - השלימו את המחלקה `MatrixTest<T>`.
 - הוסיפו את המחלקות `MatrixTestString`, `MatrixTestInteger`.
 - וודאו כי הרצת `MatrixTestString` `MatrixTestInteger` עוברת בהצלחה.
 - יש להגיש את קבצי ה-java.

פורמט קובץ ההגשה ובדיקתו:

פורמט: יש להגיש קובץ ZIP בשם

37_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!
- בדיקת קובץ ההגשה:** בדקו את הקובץ שיצרתם בתוכנת הבדיקה בקישור:

<https://cs.telhai.ac.il/homework/>

ראו [סרטון הדגמה](#) של השימוש בתוכנת הבדיקה.

הסבר על תוצאות הבדיקה האוטומטית בתרגיל זה:

- במעבדה זו, יש מספר סוגי בדיקות, וזאת במטרה שתוכלו לבדוק את עצמכם היטב לפני ההגשה. לכל test הוספו שם ותיאור המסבירים מה הוא בודק. יש 3 סוגים של tests:
1. tests שבודקים את המחלקה `SparseMatrix<T>` שמימשות, עבור `T=Object`. עבור טסטים אלו תקבלו 30 נקודות בבודק האוטומטי
 2. tests הבודקים את המחלקה `MatrixTestInteger` ו- `MatrixTestString` שרשמתם, עבור המימוש שלכם ל `SparseMatrix<T>`. עבור טסטים אלו תקבלו 15 נקודות (7 נק' עבור `MatrixTestInteger`, ו-8 עבור `MatrixTestString`)
 3. tests הבודקים את המחלקה `MatrixTestInteger` 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 שלשה סטים של בדיקות כאלו (עבור 3 מימושים שגויים של `SparseMatrix`), לכל סט 5 נקודות. הנקודות ינתנו במלואן עבור סט של בדיקות אם לפחות אחת מהבדיקות בסט נכשלה. בעקבות תקלה זמנית בבבודק האוטומטי - אתם תראו ציון חלקי ל-5 נקודות לפי מספר הטסטים שנכשלו, וכן נראה כאילו הטסט הזה מקבל 0 נק', אבל הבודק הידני יעלה ציון זה לפי שלמותם של הטסטים שסיפקתם.

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!

דוגמה לתוצאה טובה של הבדק האוטומטי:

Data Structures Lab, 2021-22, Fall, Michal Horovitz

lab04, lab04, Deadline:2021-11-30 23:59:00

Checking content of /var/www/html/homework/37_lab04_123456789

Compiling...

Testing...

Running Test: TestMatrixInteger

This test checks the correctness of SparseMatrixInteger

OK (4 tests)

Passed 4 test(s)

Grade for this test: 7.0 / 7

Running Test: TestMatrixObject

This test checks the correctness of SparseMatrix

OK (4 tests)

Passed 4 test(s)

Grade for this test: 30.0 / 30

Running Test: TestMatrixOnWrongGetTrans

This test checks the correctness of MatrixTestInteger on a wrong getTrans implementation of SparseMatrix

Tests run: 4, Failures: 1

Failed 1 out of 4 test(s)

Grade for this test: 1.25 / 0

Running Test: TestMatrixOnWrongSet

This test checks the correctness of MatrixTestInteger on a wrong set implementation of SparseMatrix

Tests run: 4, Failures: 3

Failed 3 out of 4 test(s)

Grade for this test: 3.75 / 0

Running Test: TestMatrixOnWrongTrans

This test checks the correctness of MatrixTestInteger on a wrong trans implementation of SparseMatrix

Tests run: 4, Failures: 1

Failed 1 out of 4 test(s)

Grade for this test: 1.25 / 0

Running Test: TestMatrixString

This test checks the correctness of SparseMatrix

OK (4 tests)

Passed 4 test(s)

Grade for this test: 8.0 / 8

Grade for correct format and compilation: 40

OVERALL GRADE: 91.25

This grade DOES NOT ACCURATELY REFLECT the merit of the work which is affected by ADDITIONAL FACTORS such as:

code design, algorithmic efficiency, adequacy of solution, code style, plagiarism, etc...

To get the upload page, press the back button of your browser