מעבדה 12. נושא: חיפוש לרוחב ולעומק בגרף

תאריך הגשה: 29.06 בשעה 23:00 (בזוגות)

יש לקרוא היטב לפני תחילת העבודה !

מבוא:

במעבדה זו נכיר שני אלגוריתמים בסיסיים לסריקה שיטתית של גרפים, ונוסיף למחלקה Graph שכתבנו במעבדה הקודמת מתודות למימושם.

תיאור:

BFS:

אלגוריתם **חיפוש לרוחב** (Breadth-first search או BFS) הוא אלגוריתם לסריקה שיטתית של גרף, ומהווה בסיס לאלגוריתמים חשובים רבים בתורת הגרפים (את אחד מהם נכיר במעבדה הבאה).

בהינתן גרף $G = (V, E)$ וצומת מקור $s \in V$, האלגוריתם יסרוק את צמתי G הישיגים מ-s, כך שצמתים שמרחקם מ-s קטן ייסרקו לפני צמתים שמרחקם מ-s גדול.

הדגמה: https://www.cs.usfca.edu/~galles/visualization/BFS.html

פסאודו-קוד (מתוך Introduction to Algorithms, פרק 20.2, עמ' 556):

```
BFS(G, s):
    mark s as VISITED
    Q = ∅
    ENQUEUE(Q, s)
    while Q != ∅:
        u = DEQUEUE(Q)
        for each v neighbor of u:
        // explore each edge (u,v)
            if v is not marked as VISITED:
                mark v as VISITED
                ENQUEUE(Q, v)
```

DFS:

אלגוריתם **חיפוש לעומק** (Depth-first search או DFS), כפי ששמו מרמז, סורק את הגרף (רקורסיבית) לעומקו.

"אינטואיטיבית, האלגוריתם מתחיל את החיפוש מצומת שרירותי בגרף ומתקדם לאורך הגרף עד אשר הוא נתקע, לאחר מכן הוא חוזר על עקבותיו עד שהוא יכול לבחור להתקדם לצומת אליו טרם הגיע. דרך פעולת האלגוריתם דומה במידת מה לסריקה שיטתית של מבוך." (ויקיפדיה)

בהינתן גרף $G = (V, E)$, האלגוריתם יסרוק את כל צמתי G.

הדגמה: https://www.cs.usfca.edu/~galles/visualization/DFS.html

פסאודו-קוד (מתוך Introduction to Algorithms, פרק 20.3, עמ' 565):

```
DFS(G):
    for each vertex u∈V:
        if u is not marked as VISITED:
            DFS-Visit(G, u)

DFS-Visit(G, u):
    mark u as VISITED
    for each v neighbor of u:
    // explore each edge (u,v)
        if v is not marked as VISITED:
            DFS-Visit(G, v)
```

הערה: במימוש שלנו, מאחר ומוגדר לנו סדר על הקדקודים (בעקבות השימוש ב-TreeMap, ראו מעבדה קודמת), נתחיל תמיד בקדקוד הראשון.

מימוש:

הוסיפו למחלקה Graph שכתבתן במעבדה הקודמת את המתודות הבאות:

```
public List<V> bfs(V source)
public List<V> dfs()
```

וממשו באמצעותן את שתי הסריקות.
כל מתודה תחזיר רשימת צמתים, מסודרים לפי סדר סריקתם.

- ניתן להשתמש ב-java.util.ArrayList עבור רשימת הצמתים המוחזרת.
- ניתן להשתמש ב-java.util.HashSet לסימון קבוצת הצמתים בהם ביקרנו.
- ניתן להשתמש ב-java.util.LinkedList לצורך מימוש תור.
  שימו לב: LinkedList מממשת את Queue.

קישור לגיטהב (שם יש את הטסט):
https://github.com/michalHorovitz/DSLab2023Public/tree/main/DS-Lab12-Graph-BFS-DFS

עבודה נעימה !!!

סדר העבודה ופרטים טכניים

המשיכו את הפרויקט DS-Lab11-Graph שהגשתם במעבדה הקודמת.

<u>פורמט קובץ ההגשה ובדיקתו:</u>

פורמט: יש להגיש קובץ ZIP בשם

46_lab12_123456789_987654321.zip

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

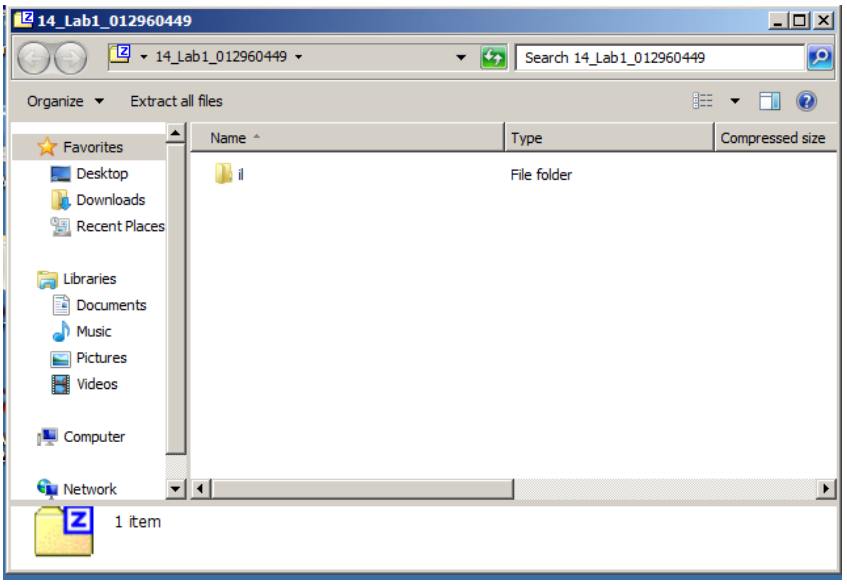
על הקובץ להכיל את כל קבצי ה JAVA שכתבתם כאשר הם נמצאים בתתי תיקיות בתוך התיקייה

il/ac/telhai/ds/

על פי המבנה של הפרוייקט הנתון.

כלומר, השורש של קובץ ההגשה יכיל רק תיקייה בשם il, והוא יכיל את כל קבצי - java על פי התבנית הנתונה בפרוייקט.

להמחשה תמונה של קובץ כזה שנפתח ב - WindowsExplorer



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

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

ראו סרטון הדגמה של השימוש בתוכנת הבדיקה.

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