הסבר לטסטים לBFS במערכות מקביליות

שלב ראשון: חילוץ הZIP למקום הנכון

הטסטים מכילים:

- תיקייה בשם slowBFS
- תיקייה בשם test-bfs-files
- קובץ בשם test-bfs.sh
- קובץ בשם ThreadPoolDetachedThreads.c (שלא רלוונטי לטסטר עצמו ויהיה עליו הסבר בסוף הקובץ)

הקבצים והתיקיות (חוץ מ-ThreadPoolDetachedThreads) מצפים לשבת באותו מקום כמו הקובץ part_III.out

אחרי החילוץ זה אמור להראות ככה:

```
rootDir/
| slowBFS/
   | SyncGraph/
      | graph.c
      | graph.h
   | BFS.c
   | BFS.h
   | main.c
   | Makefile
   | slowBFS.out
| test-bfs-files/
   | checkResults.py
   | createGraphs.py
| test-bfs.sh
| part_III.out
```

שלב שני: התקנת ספריית פייתון.

בשביל ליצור את הגרפים נצטרך להתקין ספרייה בפייתון ליצירת גרפים על ידי הפקודה:
pip install networkx

שלב שלישי: ניתנת הרשאות הרצה לסקריפט בדיקה

נחזור לתיקייה שבה היינו בשלב 1 ונריץ chmod +x ./test-bfs.sh

שלב רביעי: ניתן את הקלט לטסטר

דברים חשובים לפני שמתחילים להריץ:

- אני מצרף את קובץ הריצה של BFS סדרתי, הוא קומפל על השרתים של האוניברסיטה, ואמור לעבוד עקרונית גם אצלכם אבל ליתר ביטחון צירפתי את קבצי המקור ותוכלו לעשות MAKE בעצמכם
- **שימו לב שאתם מריצים את הסקריפט מהמיקום שבו הוא נמצא ולא דרך תיקייה אחרת כי הCWD חשוב.**

הטסטר ישאל 3 שאלות:

בכל הקלטים, אם הברירת מחדל מתאימה אפשר פשוט ללחוץ אנטר.

1) כמה טסטים להריץ, עם ברירת מחדל 10
2) מה כמות הקודקודים **המינימלית** בגרפים שניצור, עם ברירת מחדל 50
3) מה כמות הקודקודים **המקסימלית** בגרפים שניצור, עם ברירת מחדל 100

דוגמא:

```
How many tests? (default: 10):
3
What is the minimum amount of nodes? (default: 50):

What is the maximum amount of nodes? (default: 100):
```

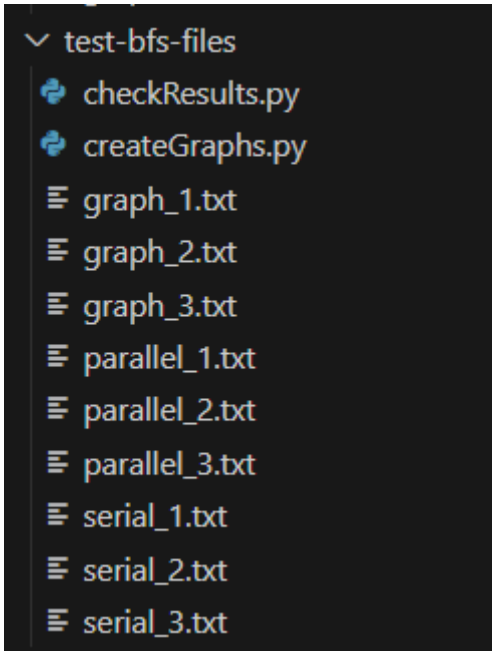
<u>שלב חמישי: להבין את הפלטים</u>

הנה פלט לדוגמא של הטסטר

```
Removed Old files from test-bfs-files.
Finished generating new test cases.
Finished running tests.: 100%

Graph 1: time for parallel: 0m2.551s, time for serial: 0m0.010s
Graph 2: time for parallel: 0m2.118s, time for serial: 0m0.008s
Graph 3: time for parallel: 0m0.602s, time for serial: 0m0.007s

Comparison 1: Success
Comparison 2: Success
Comparison 3: Success

Summary: I checked 3 files, successes: 3, failures: 0
```

לכל גרף הוא מראה כמה זמן לקח לסדרתי וכמה למקבילי (ואפשר לראות שאצלי המקבילי לוקח יותר זמן 🙃 🙃 🙃 🙃 🙃) וגם האם הפלט זהה או שונה, וההצלחות והכשלונות נספרים בהתאם.

במידה ויש גרף שנתן פלט שונה אפשר להסתכל בתיקייה bfs-test-files כדי לראות מה הקלטים ומה הפלטים. אחרי הרצה התיקייה תיראה ככה:



graph_i – הקלט שיצרנו

parallel_i - הפלט של הקוד המקבילי

serial_i – הפלט של הקוד הסדרתי

זה המקום להגיד: **כל הרצה היא אקראית לגמרי ומוחקת את הקלטים והפלטים של ההרצה הקודמת. אם אתם רוצים לשמור אותם תוציאו מהתיקייה את הקבצים כי כל קובץ txt בתוך התיקייה bfs-test-files נמחק בתחילת כל טסט.**

<u>טיפול בשגיאות זיכרון מהthread pool של המתרגלים</u>

אם נתקלתם בשגיאה הבאה:

כנראה שכמוני ניסיתם להריץ עם קלטים יחסית גדולים, הסיבה שזה קורה היא כי thread תופס מקום בזיכרון גם כשהוא מסיים לרוץ. כדי לפתור את זה שיניתי קצת את הקובץ של הthreadPool שיצור detached threads ואז הם מפנים את הזיכרון כשהם מסיימים לעבוד, מי שמעוניין להשתמש בדרך הזו יכול פשוט להעתיק מהקובץ threadPoolDetachedThreads.c.