**פרויקט גמר מערכות הפעלה**

**שאלה 1 :**

התבקשנו ליצור מבנה נתונים של גרף, בחרתי להשתמש במטריצת שכנויות.

כאשר יש צלע יופיע 1 וכאשר אין צלע inf .

|  |
| --- |
|  |

**שאלה 2** : בשאלה זו התבקשנו לממש אלגוריתם למציאת מעגל אויילר או הוכחה שאינו קיים.

נבצע זאת במספר שלבים:

שלב ראשון: נבחן האם כל דרגות הקודקודים הגרף זוגיות, במידה לא קיים מעגל אויילר (נלמד בדיסקרטיים). במקרה זה נדפיס את הקודקוד שדרגתו אי זוגית.

|  |
| --- |
|  |

שלב שני: נבחן האם הגרף קשיר, נעשה זאת באמצעות הפעלת DFS על הגרף. אם הגרף אינו קשיר (בתנאי שיש צלעות בשני רכיבי הקשירות) הוא בפרט לא מכיל מעגל אויילר נדפיס את הקודקוד שהתגלה כלא קשיר.

|  |
| --- |
|  |
|  |

שלב שלישי: אחרי שאנחנו יודעים שקיים נמצא אותו באופן הבא:

* ניצור מטריצת עזר שתהיה העתק של מטריצת השכנויות המקורית.
* יצירת מחסנית שתחזיק את הנוכחי.
* נתחיל מקודקוד מקור כלשהו, נכניס קודקוד שכן אחד שלו למחסנית ונסיר את הצלע בניהם.
* נעבור לקודקוד הבא במחסנית ונכניס קודקוד שכן אחד שלו ונסיר את הצלע בניהם.
* באופן זה נמשיך עבור כל הצלעות עד שנגיע לקודקוד שלא קיימות עבורו צלעות וזה אומר שסיימנו את הקטע ממסלול זה ונכניס אותו לווקטור של אויילר, נמשיך רקורסיבית על התוכן של המחסנית.
* נמשיך כך כל עוד המחסנית מכילה קודקודים.

|  |
| --- |
|  |

**שאלה 3 :**

בשאלה זו התבקשנו ליצור גרף רנדומלי בעזרת דגלים שנקבל מהטרמינל.

נעזרנו בפונקציה getopt\_long על מנת לפרסר בצורה נוחה את הקלט המתקבל.

נבצע בדיקות תקינות על הקלט (בדיקה שבאמת הגיע מספר, בדיקה שמספר הצלעות לא גבוה מידי , בדיקה שכל הארגומנטים אותחלו)

נאתחל את השדות של הגרף, נקצה מקום למטריצת השכנויות ונפעיל את הפונקציה היוצרת גרף רנדומלי.

|  |
| --- |
|  |
|  |
| את יצירת הגרף הרנדומלי נבצע כך:   * יצירת ווקטור שיכיל את כל זוגות הצלעות האפשריות. * ערבוב הווקטור בעזרת הגרעי הרנדומלי שקיבלנו בקלט. * הוספה לגרף את e הצלעות הראשונות בווקטור הרנדומלי (כאשר e זה מספר הצלעות הנדרשות) |
|  |
| לבסוף נפעיל את מציאת מעגל אויילר על הגרף שנוצר ונקבל אחת מהאופציות הבאות:   1. אין מעגל אויילר כי קיימת דרגה אי זוגית בגרף של הקודקוד i . 2. אין מעגל אויילר כי הגרף לא קשיר, הקודקוד s לא מגיע אל הקודקוד i. 3. יש מעגל אויילר והוא : (תצוגה של מעגל אויילר בקודקודים וחצים) |
|  |
| נציג 2 קלטים (seed=42):   1. 5 קודקודים , 10 צלעות – הגרף המלא כאשר כל הדרגות הם 4 לכן קיים מעגל אויילר. 2. 6 קודקודים, 15 צלעות – הגרף המלא כאשר כל הדרגות הם 5 – אין מעגל אויילר. |
|  |
|  |

**שאלה 4 :**

|  |
| --- |
| בתיקיית gcov\_reports נמצאים הדוחות של .cpp.gcov שמציגים את השורות שהגענו אליהם.  כאן נציג את הדוח שיוצא הטרמינל. |
| עבור main.cpp : עם ההרצה של 2 גרף עם מעגל אויילר וגרף ללא מעגל אויילר. |
|  |
| עבור Graph.cpp אתאר אילו הרצות ביצעתי בשביל להגיע לכיסוי:   * חוסר בדגלים , דגלים לא קיימים. * קלט בטווח לא מתאים או לא מספר. * יותר מידי צלעות. * קלט תקין ללא מעגל אויילר. * קלט תקין עם מעגל אויילר. |
|  |
| כעת נבצע בדיקת זליגת זיכרון באמצעות valgrind:  ניתן לראות שאין שום הקצאה של משתנים בheap (כי לא ביצענו new) ושאין זליגת זיכרון. |
|  |
| נציג גם את גרף הקריאות שמתאר את חלוקת הקריאות בין הפונקציות השונות, מי קוראת למי וכמה פעמים כל פונקציה נקראת. באופן זה נזהה צווארי בקבוק.  כדי לראות תצוגה גרפית תקינה נתקין כלי בשם KCachegrind עם הפקודה :  sudo apt install kcachegrind  ואחרי שהרצנו:  Make valgrind-callgrind  נריץ את השורה הבאה:  kcachegrind callgrind.out.\* |
|  |
|  |
|  |
| ניתן לראות כי הפונקציה שצרכה הכי הרבה משאבים היא parseFlags עם 4.38 מיליון הוראות למעבד. |

**שאלה 5 : נכתבה בטעות לא סופק קוד.**

**שאלה 6 :**