בר אברהם דעבול ת.ז 324079243, ונדב יוסף זדה ת.ז 208825257

תרגיל בית 2 – הסברים על התוכניות

שאלה 1

בשאלה זו נדרשנו לכתוב תוכנית שמקבלת כאינפוט מספר שלם חיובי n, ויוצרת את כל תתי-הגרפים הקשירים מגודל n. התוכנית תדפיס את הפלט בקובץ טקסט בפורמט המבוקש.

החלטנו לכתוב את התוכנית בשפת Python, מכיוון שהיא שפת תכנות שמאפשרת לייצג רשתות וגרפים באופן פשוט וגמיש, וגם מכילה ספריות קוד ידועות ובטוחות שנוח לעבוד איתן. בתוכנית שלנו הסתכלנו על כל הקומבינציות של הקשתות האפשריות, ומתוכן חילצנו רק את תתי-הגרפים הקשירים ולא איזומורפיים בזוגות בגודל n. לאחר שקיבלנו את תתי-הגרפים, כתבנו אותם לקובץ טקסט בפורמט המבוקש. לצורך כך, כתבנו מספר מתודות, ונסביר את אופן הפעולה של כל אחת מהן.

outputAllConnectedSubGraphs – המתודה הזאת מקבלת כאינפוט מספר שלם חיובי n ומבצעת את הפעולה המבוקשת בשאלה. היא מייצרת את כל תתי הגרפים מגודל n בעזרת שימוש במתודה generateAllConnectedSubGraphs. לאחר מכן, היא מדפיסה את תתי הגרפים בקובץ טקסט לפי הפורמט הרצוי, בעזרת שימוש במתודה writeSubgraphsToFile.

generateAllConnectedSubGraphs – זוהי המתודה שמבצעת את הפעולה הראשית בתוכנית שלנו. היא מקבלת כקלט מספר שלם חיובי n ומחזירה את כל תתי הגרפים המכוונים וקשירים מגודל n. נסביר כיצד היא עובדת.

ראשית, אנחנו בודקים את המקרה המיוחד שבו n=1, במקרה זה נחזיר את התת-גרף היחיד שכולל קודקוד אחד עם לולאה עצמית.

אחרת, אנחנו יוצרים את כל הקשתות האפשריות ואת כל הקומבינציות האפשריות של קשתות (בעזרת שימוש בספרייה itertools). לכל קומבינציה של קשתות, נבנה גרף מכוון מתאים, ונבצע עליו מספר בדיקות: 1. אם מספר הקודקודים של הגרף שווה ל-n. 2. אם הגרף קשיר (בעזרת isConnected). 3. אם הגרף אינו איזומורפי לכל אחד מהגרפים שכבר הוספנו לרשימת תתי-הגרפים (בעזרת isIsomorphic). במידה וכל הבדיקות נכונות, נוסיף את הגרף לרשימת תתי-הגרפים.

isConnected – המתודה הזאת היא מתודת עזר שמקבלת גרף מכוון, ומחזירה תשובה בוליאנית אם הוא קשיר או לא. בבדיקה אנחנו הופכים את הגרף ללא מכוון ובודקים אם הגרף המתקבל הוא קשיר. הבדיקה הזאת נעשית בעזרת שימוש בספריית networkx.

isIsomorphic – המתודה הזאת היא מתודת עזר שמקבלת גרף מכוון ורשימה של גרפים מכוונים. היא בודקת אם הגרף הנתון איזומורפי לאחד מהגרפים שברשימה, ומחזירה תשובה בוליאנית בהתאם. הבדיקה הזאת נעשית בעזרת שימוש בספריית networkx.

writeSubgraphsToFile – הפונקציה הזאת מקבלת כקלט רשימה של תתי-גרפים, ומספר שלם חיובי n. היא יוצרת קובץ טקסט בפורמט "subgraphs_n_{n}.txt", וכותבת בתוכו את תתי-הגרפים הנתונים בפורמט הנדרש.

שאלה 2

בשאלה זו נדרשנו לכתוב תוכנית שמקבלת כאינפוט מספר שלם חיובי n וגרף מכוון בפורמט נתון (גרף שמיוצג ע"י קשתות בקובץ טקסט). התוכנית יוצרת את כל תתי-הגרפים הקשירים מגודל n (כמו בשאלה 1), ובודקת לכל תת-גרף את מספר ההופעות שלו בגרף הנתון. התוכנית תדפיס את הפלט בקובץ טקסט בפורמט המבוקש.

בתוכנית שלנו השתמשנו בקוד של שאלה 1 כדי לייצר את כל תתי הגרפים. לכל תת גרף, עברנו על כל הקומבינציות של תתי הגרפים של הגרף הנתון עם אותו גודל, ובדקנו אם הם איזומורפים לתת-הגרף. בדרך הזאת, מצאנו את מספר ההופעות של כל תת גרף, ובסופו של דבר הדפסנו פלט מתאים לקובץ טקסט. לצורך כך, כתבנו מספר מתודות, ונסביר את אופן הפעולה של כל אחת מהן.

loadGraphFromText – הפונקציה הזאת מקבלת path לקובץ טקסט שמייצג את הגרף הנתון ע"י הפורמט הנתון, ומייצרת ממנו גרף מכוון מתאים.

writeSubgraphsWithCountingToFile - הפונקציה הזאת מקבלת כקלט רשימה של תתי-גרפים, רשימה של מספר ההופעות של כל תת-גרף בגרף הנתון ומספר שלם חיובי n. היא יוצרת קובץ טקסט בפורמט "SubgraphsWithCounting_n_{n}.txt", וכותבת בתוכו את תתי-הגרפים הנתונים ואת מספר ההופעות של כל אחד מהם בפורמט הנדרש.

countSubgraphOccurrences – הפונקציה הזאת מקבלת גרף מכוון ותת-גרף, ומחזירה את מספר ההופעות של אותו תת-גרף בגרף הנתון.

בפונקציה הזאת אנחנו מייצרים את כל הקומבינציות של הצמתים של הגרף הנתון בגודל של תת-הגרף הנתון (בעזרת הספרייה itertools). לכל קומבינציה כזאת, אנחנו מייצרים את כל תתי-הגרפים שיכולים להתקבל מהגרף הנתון עבור אותה קומבינציה של צמתים. לכל תת-גרף, אם הוא איזומורפי לתת-הגרף הנתון, אנחנו מוסיפים אחד למשתנה count שסופר את מספר ההופעות של תת-הגרף הנתון בגרף הנתון (בתחילה, הערך של count מאותחל ל-0).

countAppearancesOfMotifsInGraph – המתודה הזאת מקבלת כקלט מספר שלם חיובי n ו-path לקובץ טקסט שמכיל גרף נתון בפורמט המבוקש, והיא מבצעת את הפעולה המבוקשת בשאלה.

בתחילה, היא קוראת את הנתונים מקובץ הטקסט ומייצרת ממנו גרף מכוון מתאים בעזרת המתודה loadGraphFromText. לאחר מכן, היא מייצרת את כל תתי-הגרפים שלו בעזרת המתודה generateAllConnectedSubGraphs משאלה 1. בהמשך היא עוברת על כל תת גרף, ובודקת את מספר ההופעות שלו בעזרת countSubgraphOccurrences. בסופו של דבר היא כותבת את התוצאות בקובץ טקסט בפורמט המבוקש בעזרת המתודה writeSubgraphsWithCountingToFile.

שימוש בספריות קוד חיצוניות (עבור 2 החלקים):

Network – זוהי ספריית פייתון שמשמשת ליצירת רשתות מורכבות, לביצוע מניפולציות עליהן וללמוד אותן. בפרויקט שלנו, החלטנו להשתמש בספרייה הזאת, מכיוון שהיא עזרה לנו לבנות בצורה טובה את הגרפים. בנוסף, יש בה מימושים של מתודות להפיכת גרף מכוון ללא מכוון, לבדיקת קשירות של גרף ולבדיקת איזומורפיות בין 2 גרפים. אנחנו השתמשנו במתודות האלה בתוכנית שלנו.

Itertools – זוהי ספריית פייתון שמכילה סט של כלים שמשמשים לביצוע איטרציות, שעובדים בצורה מהירה וחסכונית בזיכרון. בפרט, בספרייה הזאת יש כלים שמייצרים פרמוטציות וקומבינציות, ומאפשרים לעבור עליהם כאיטרטורים. אנחנו השתמשנו בספרייה הזאת כדי לייצר את כל הקשתות האפשריות ואת כל הקומבינציות האפשריות של קשתות במתודה generateAllConnectedSubGraphs וכן כדי לייצר את כל הקומבינציות של הצמתים במתודה countSubgraphOccurrences.

timeit – זוהי ספריית פייתון ששימשה אותנו ב-main של התוכנית הראשונה, ובעזרתה מדדנו את זמן הריצה של הקוד עבור ההרצות השונות.