שאלה 1:

א. בחרנו לחשב את כל המוטיבים על ידי זה שבהתחלה אנחנו מחשבים את כל האופציות שיש לגרף עם n קודקודים, שאין לו לולאות עצמיות. לאחר מכן, לקחנו את כל האופציות שקיבלנו והורדנו את כל הגרפים שהם אינם קשירים. לאחר מכן, לקחנו שוב את האופציות שנותרו והשארנו רק גרף ייחודי יחיד מכל הגרפים שחוזרים על עצמם (איזומורפיים). בסוף לאחר שקיבלנו את מה שנותר, שמרנו את התוצאות לקובץ txt לפי הפורמט שהוגדר לנו.
כל השלבים שתיארנו לעיל נעשו על ידי הפונקציות הבאות:
Permutations- פונקציה שמקבלת את n, מספר הקודקודים, ומחזירה את כל האופציות של המחרוזות הבינאריות בגודל $n(n-1)$. הסיבה לכך היא מכיוון שגרף עם n קודקודים, ניתן לתאר אותו בעזרת מטריצה nXn. בגלל שאין לולאות עצמיות ידוע שעל האלכסון של המטריצה צריך להיות אפסים. לכן, נשאר לנו עוד $n^2 - n = n(n-1)$ איברים חופשיים שצריך לתת להם ערך של 0 או 1. ניתן לפרוש את המטריצה ולייצג אותה במערך אחד ארוך שמכיל $n(n-1)$ איברים, וכך זה בעצם שקול לייצוג גרף מסוים.

RemoveUnConnectedGraphs- מקבלת את כל האופציות מהפונקציה הקודמת, ומורידה את כל הגרפים הלא קשירים. סינון ראשוני על ידי אלו שמספר הקשתות שלהם קטן ממספר הקודקודים + 1, שזה בהכרח אומר לא קשיר. סינון נוסף זה על ידי זה שעוברים על כל אופציה של גרף ומבצעים DFS. אם יוצא שיש רכיב קשירות יחיד, הגרף קשיר, אם לא, מוציאים אותו.

RemoveNames- מקבלת את כל האופציות שנותרו לאחר הפונקציה הקודמת, ומורידה עותקים של אותו גרף (להוריד איזומורפיות). עושים זאת על ידי זה שעוברים על כל גרף, ומשנים לכל גרף את מספרי האינדקסים שלו, לכל מיפוי אפשרי של שמות לקודקודים. לכל מיפוי כזה בודקים אם יש גרף אחר שקיים כבר וזהה לזה, אם כן, מורידים את הגרף שמסתכלים עליו, כי יש כזה בייצוג אחר של שמות לקודקודים בהמשך.

SaveResultsToFile- מקבלת את כל האופציות שנותרו לאחר הפונקציה הקודמת, ושומרת לקובץ txt לפי הפורמט שהוגדר. נשמר בתיקייה q1_results.

* נעזרנו בספריית numpy שהקלה עלינו כאשר נדרשנו לעבוד עם מטריצות, לחשב transpose וכדומה. נעזרנו גם בספריית itertools, שהקלה עלינו בחישוב כל הפרמוטציות האפשריות למספרים בין אחד עד n וגם כאשר חישבנו את כל האופציות למספרים בינאריים בגודל n. יכלנו לממש הכל בעצמנו, אבל הרגשנו שזה מיותר כי זה היה פשוט להעתיק את המימוש של הפונקציות האלה מהאינטרנט...

ב. הקבצים מצורפים בתיקייה q1_results.

ג. מניתוח סיבוכיות הקוד שלנו, יש לנו בפונקציה RemoveNames, שמשאירה רק את הגרפים הייחודיים (מוטיבים), יש לה מעבר על כל גרף, שזה $2^{n(n-1)}$ אפשרויות, ולכל אפשרות כזו אנו עוברים על $n!$ אופציות לתת שמות לקודקודי הגרף. לכן, בהכרח בסיבוכיות שלנו יהיה לנו איבר שהוא מהצורה $2^{n(n-1)} * n!$.

נניח כי לכל פקודה לוקח זמן שווה לחישוב, לכן הזמן הכולל לתוכנית בערך יהיה גדול שווה מהזמן:

$$2^{n(n-1)} * n! * t_{command}$$

יצא לנו עבור $n = 2$: 0.1520357 שניות -> $t_{command} = 0.0190044625$

יצא לנו עבור $n = 3$: 0.1857483 שניות -> $t_{command} = 0.00048371953125$

יצא לנו עבור $n = 4$: 9.1060032 שניות -> $t_{command} = 0.0000926310546875$

גם אם ניקח את הזמן הטוב ביותר לחישוב פקודה בודדת, עדיין עבור $n = 5$ הזמן שייקח יהיה לפחות: $2^{5*4} * 5! * 0.0000926310546875 = 11,655\ s = 194\ m = 3\ h$, כלומר לפחות 3 שעות (כמובן שזה גם תלוי מחשב).

לכן, ניתן לומר כי עד $n = 4$ נוכל לחשב בזמן שקטן משעה.

גם הרצנו וראינו שלוקח יותר משעה עבור $n = 5$.

ד. נראה שניתן גג עד $n = 5$, אחרי נראה שזה לוקח כבר הרבה יותר זמן.

הערה: כתבנו קוד פייתון *mesureTime.py* אשר מחשב את הזמן שלקח לתוכנית לרוץ כתלות ב- *n*. צירפנו אותו בתיקייה. נעזרנו בו בשביל החישובים לעיל של זמני הריצה.

<u>שאלה 2:</u>

בחרנו לקבל את הקלט לגרף על ידי זה שמכניסים בתור קלט לאחר הרצת התוכנית את מספר הקודקודים, ואת הקשתות שיש בגרף, לפי הפורמט שבו עבור קשתות: $(a_1, b_1), (a_2, b_2), \ldots$, צריך להכניס כקלט את המחרוזת הבאה: $a_1\ b_1, a_2\ b_2, \ldots$.

כדי לחשב את מספר המוטיבים בגרף, נעזרנו בקוד שכתבנו עבור שאלה 1.

הרעיון שלנו לספירת הופעת כל מוטיב בגרף, הוא על ידי זה שאנחנו עוברים על כל מוטיב, ובודקים האם הוא מופיע בגרף הקלט. עושים זאת עבור כל אופציה של סידור שמות לקודקודים לאותו מוטיב. האופן שבו אנו בודקים אם מוטיב נמצא בגרף הקלט, הוא על ידי זה שאנו לוקחים את מטריצת השכנויות של המוטיב ואת מטריצת השכנויות של גרף הקלט, ועושים ביניהם *AND* באופן של איבר איבר. התוצאה תהיה גם מטריצה, ואם באמת המוטיב עבור סידור שמות לקודקודיו באופן מסוים, מופיע בגרף הקלט, אז התוצאה של *AND* צריכה להיות מטריצה ששווה ממש למטריצת השכנויות של המוטיב עם הלייבלינג הספציפי לקודקודים שבדקנו.

כל השלבים שתיארנו לעיל נעשו על ידי הפונקציות הבאות:

q1 functions- כל הפונקציות של שאלה 1 כפי שהסברנו לעיל.

ConvertGraphToMatrixForm- פונקציה שאחראית על להמיר את הפורמט שבו אנו מקבלים את הגרף כקלט למטריצת שכנויות.

AND2Lists- פונקציה שאחראית לעשות *AND* איבר איבר בין 2 מערכים ולהחזיר מערך שמכיל את התוצאות עבור כל זוג איברים.

FindNumberOfMotifs- פונקציה שאחראית למצוא את מספר הפעמים שכל מוטיב מופיע בגרף לפי העיקרון שהסברנו לעיל. לכל מוטיב, עוברים על כל אופציה לתאר את אותו מוטיב כתלות במספר הקודקודים, ועל כל אופציה כזו עושים *AND2Lists* ובודקים אם אותו מוטיב נמצא בגרף הקלט.

SaveResultsToFile- מקבלת את כל האופציות שנותרו לאחר הפונקציה הקודמת, ואת כמות הפעמים שמופיע כל סוג של מוטיב בגרף, ושומרת לקובץ txt. נשמר בתיקייה q2_results.

* נעזרנו בספריית numpy שהקלה עלינו כאשר נדרשנו לעבוד עם מטריצות, לחשב transpose וכדומה. נעזרנו גם בספריית itertools, שהקלה עלינו בחישוב כל הפרמוטציות האפשריות למספרים בין אחד עד n. יכלנו לממש הכל בעצמנו, אבל הרגשנו שזה מיותר כי זה היה פשוט להעתיק את המימוש של הפונקציות האלה מהאינטרנט...

הערה: הוספנו לשתי השאלות אופציה של ציור הגרפים בעזרת הספרייה networkx, אם רוצים להשתמש בזה, צריך להוריד את ההערות איפה שרשום בקוד.