הסבר לפתרון שאלה 1

א. בשאלה 1 התבקשנו ליצור קוד שבהינתן מספר טבעי וגדול מ0 נותן לנו את הקשתות של כל תתי הגרפים המכוונים האפשריים שניתן להרכיב ממס' קודקודים השווה למס' הנתון.

הקוד מורכב מפונקציה אחת שנשתמש בה מס' פעמים, וכעת נסביר אותה:

הקוד משתמש ב2 פונקציות מספריית itertools המובנת בפייתון, והראשונה היא: permutations, שבהינתן קבוצת מספרים S ומספר טבעי r מחזירה את כל הזיות האפשריות שניתן להרכיב מS ללא חזרות ועם חשיבות לסדר.

תחילה, אנחנו דואגים ליצור עם פונקציה זו את כל הקשתות האפשריות שניתן לקבל בגרף מכוון זה (pairs), כאשר הערך המוחזר מהפונקציה הוא אובייקט, אז אנחנו דואגים להמירו לlist, ושורה לאחר מכן אנחנו דואגים שהמשתנה יהי מורכב אך ורק מlists בתוכו. מיד אחרי כן אנו מדפיסים את המספר שנבחר להיות הinput בפורמט שרציתם.

אחר כך, אנחנו מתחילים לולאה שעוברת על כל גודל אפשרי של תת גרף שיכול להיות, ומונה באמצעות הפונקציה השנייה מספריית itertools המובנת בפייתון, combinations, את כל קבוצות הקשתות שיכולות להיות, כמובן ללא חזרות וכמס' הנוכחי באיטרציית הלולאה. על כל קבוצת קשתות (דהיינו, גרף אפשרי) אנו מוסיפים למשתנה count 1, וככה אנו עוקבים אחרי כמה גרפים יש. אנחנו דואגים לאחסן את קבוצות הקשתות בlists, ולכל קבוצה אנחנו עושים המרה לnp.array ולאחר מכן מבצעים הורדת מימדים מיותרים דרך squeeze. ללא squeeze ההדפסה לא תצא כפי שרציתם.

לבסוף, מדפיסים את count, מדפיסים את המס' הסידורי של תת הגרף ואז את תת הגרף עצמו. אנחנו השארנו את הסוגריים בהדפסות מטעמים פרקטיים של הקלה על כתיבת הפונקציה, ואנחנו מקווים שאתם מוצאים זאת בגדר הסביר ולא תורידו על כך נקודות.

ב. כתבנו 4 שורות הפולטות את התוצאות עבור 1 עד 4:

```
n=1
count=0
n=2
count=3
#1
[1 2]
#2
[2 1]
#3
[[1 2]
 [2 1]]
n=3
count=63
#1
[1 2]
#2
[1 3]
```

```
#4094
[[1 3]
 [1 4]
 [2 1]
 [2 3]
 [2 4]
 [3 1]
 [3 2]
 [3 4]
 [4 1]
 [4 2]
 [4 3]]
#4095
[[1 2]
 [1 3]
 [1 4]
 [2 1]
 [2 3]
 [2 4]
 [3 1]
 [3 2]
 [3 4]
 [4 1]
 [4 2]
 [4 3]]


Process finished with exit code 0
```

(צילומי מסך של תחילת הoutput וסוף הoutput בהתאמה)

הייתה לכם טעות קטנה בדוגמא שלכם: כללתם את [2 1] וגם את [1 2], [2 1] מה שאומר שהגרף מכוון (אחרת היה רק את [2 1] שכן בגרף לא מכוון [2 1] ו[1 2] הם זהים) אבל לא כללתם את גרף שמכיל את [1 2] בלבד. אנחנו כן כללנו אותו.

עד לכאן הקוד רץ. מעתה והלאה אנו מבצעים ניתוחים תיאורטיים.

לסעיפים ג. וד. עלינו לעשות ניתוח זמן ריצה (מבלי להריץ את הקוד כי זה ייקח יותר מדי זמן).

Permutations & Combinations	Order	Repetition	Formula
Permutations Order matters ^nP_r or $_nP_r$	Yes	Yes	n^r
	Yes	No	$\frac{n!}{(n-r)!}$
Combinations Order does NOT matter ^nC_r or $_nC_r$	No	No	$\frac{n!}{r!\,(n-r)!}$
	No	Yes	$\frac{(n+r-1)!}{r!\,(n-1)!}$

הלולאה הראשונה היא הכי ארוכה, והיא מחשבת את מספר הצלעות הבא:

$$\left(\frac{n!}{r!\,(n-r)!}\right)^{\frac{n!}{(n-r)!}} = \left(\frac{n(n-1)}{2}\right)^{n(n-1)}$$

כאשר n הוא האינפוט שלנו.

בשעה יש 3600 שניות.

נניח שלחשב צלע אחת לוקח t שניות.

$$\frac{1}{t} = \frac{3600}{\left(\frac{n(n-1)}{2}\right)^{n(n-1)}} \Rightarrow 3600t = \left(\frac{n(n-1)}{2}\right)^{n(n-1)}$$

עכשיו צריך להציב מספרים:

1.0 (המספר האמיתי הוא 0, כי 0 בחזקת כל דבר זה 0)

1.0

729.0

2176782336.0

1e+20

1.9175105923288408e+35

3.4135823067412403e+55

1.0986257024512136e+81

1.1318270138763687e+112

6.15355716984814e+148

אלו התוצאות עבור n=1~10. בוא נניח לרגע שt הוא שנייה (אף על פי שהדבר איננו מציאותי, אך קל לחישוב לפחות). זה אומר שעבור n=4, ייקח לנו 2176782336 שניות, שזה:

$$2\ 176\ 782\ 336 / 3600 =$$

$$604661.76$$

שעות.

N=3 הוא התשובה לכל הסעיפים, אבל זה תלוי במהו t.