תכנות פייתון – תרגיל בית 4

תאריך הגשה: 27.06.22

הנחיות להגשה:

- ההגשה היא **ביחידים בלבד**, דרך המודל
- קובץ שאינו רץ או אינו בפורמט הנכון **לא ייבדק ויקבל ציון 0**
- ניתן להשתמש בספריות פייתון
- פורמט:

יש להגיש כל שאלה וכל סעיף **בקובץ פייתון נפרד** ולאחד לzip בשם "ex3_ID" עבור כל השאלות בתרגיל.

  - שם כל קובץ הוא מספר השאלה ,מספר הסעיף ותעודת הזהות של הסטודנט.
    לדוגמא, עבור שאלה 1 סעיף א שם הקובץ הוא **Q1_A_ID.py**
  - עבור שאלה ללא סעיפים שם הקובץ הוא מספר השאלה ותעודת זהות הסטודנט.
    לדוגמא, עבור שאלה 2 שם הקובץ הוא **Q2_ID.py**

יש לרשום את שמות הפונקציות ומחלקות **בדיוק** לפי הרשום בשאלה (וודאו כי האיות נכון בכל שאלה)

- שימו לב, אם בשאלה רשום: def solve(A) פונקציה הרשומה def Solve(A) אינה בפורמט הנכון ותקבל ציון 0.

הערה: רצוי להריץ את הדוגמאות ולראות שאכן הקוד מחזיר פתרון נכון.

שאלה 1

נתון עץ חיפוש T בו הוחלפו 2 ערכים כך שהעץ אינו תקין.

א. כתבו תוכנית check_tree(A) המקבלת כקלט את A השורש של העץ T ומוצאת מוצאת את 2 הערכים שהוחלפו ומחזירה את ערכים אלו.

ב. כתבו תוכנית correct_tree(A) המקבלת כקלט את A השורש של העץ T ומתקנת את העץ ומחזירה עץ חיפוש תקין ואת מספר ההחלפות שצריך לבצע כדי לתקן את העץ.

עץ מוגדר ע"י האובייקט הבא:

```
# class TreeNode:
#   def __init__(self, x):
#       self.val = x
#       self.left = None
#       self.right = None
```

לדוגמא קלט:

A שורש העץ T

A.val =1

A.left = 2

A.right = 3

```
T =  1
    /   \
   2     3
```

פלט סעיף א:

[1, 2]

פלט סעיף ב:

```
1
    2
  /   \
 1     3
```

שאלה 2

מנהל מלון צריך לקבל N הזמנות חדרים לעונת הקיץ. במלון יש C חדרים. הזמנה כוללת תאריך הגעה ותאריך יציאה מהמלון. עליכם לכתוב תוכנית hotel_schedule(A,B,C) הבודקת אם יש מספיק חדרים במלון לענות על כל ההזמנות. התוכנית מקבלת כקלט, מערך מספרים A המיצג תאריכי הגעה, מערך מספרים B המייצג תאריכי יציאה, וC מספר החדרים במלון. התוכנית מחזירה True אם יש מספיק חדרים וFalse אם אין.

עליכם לכתוב תוכנית שרצה בזמן ריצה O(NlogN)

לדוגמא:

קלט:

A = [1, 3, 5]

B = [2, 6, 8]

C = 1

פלט:

False

הסבר:

ביום החמישי יש 2 אנשים במלון אך יש למלון רק חדר אחד.

שאלה 3

א. נתון כקלט לוח סודוקו מלא חלקית בצורה של רשימה של מחרוזות, כתוב תוכנית check_sudoku(A) הבודקת אם הלוח תקין. אם כן, החזר 1 אחרת החזר 0.
לוח תקין לא חייב להיות עם פתרון.
הלוח נתון כמטריצה בה משבצות ריקות נתונות כ"."

ב. בהינתן לוח סודו תקין, כתוב תוכנית solve_sudoku(A) אשר פותרת את הלוח ומחזירה את הפתרון המלא.

לדוגמא:

לוח סודוקו:

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

מיוצג כקלט ע"י:

["53..7....", "6..195...", ".98....6.", "8...6...3", "4..8.3..1", "7...2...6", ".6....28.", "...419..5", "....8..79"]

לאחר פתרון הסודוקו:

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

מיוצג כפלט ע"י:

[[534678912], [672195348], [198342567], [859761423], [426853791], [713924856], [961537284], [287419635], [345286179]]

שאלה 4

נתון כקלט גרף מכוון G כרשימת שכנות (adjacency list).

א. כתוב תוכנית connected_comp(G) אשר מקבלת את הגרף G כקלט ומחזירה את מספר הרכיבים המקושרים בגרף (connected components) ואת מספר הקודקודים ברכיב המקושר הכי גדול. עשה זאת ללא שימוש בספריות.

ב. כעט כתוב את אותה תוכנית כאשר אתה מבצע שימוש בספריית networkx.

קלט:

[[1,3],[0],[3],[2,0],[]]

פלט:

([[4] ,[2 ,3 ,1 ,0]], 4)

שאלה 5

א. כתבו תוכנית בשם maze_generate(N,M) אשר מקבלת כקלט את גודל המבוך הנדרש N הוא מספר השורות וM הוא מספר העמודות. התוכנית תחזיר את המבוך כמטריצה mat של 0 ו1, כאשר 1 זה דרך ו0 קיר. נקודת כניסה למבוך S ונקודת יציאה E. וודאו שניתן לפתור את המבוך.

ב. ישנו מבוך הנתון כמטריצה NXM עם כניסה ויציאה אחת, בדומה למבוך מסעיף א. כתוב תוכנית maze_solver(mat,S,E,K) אשר פותרת את המבוך אך בכל צעד חייבים לבצע K צעדים. נניח כי אנו נמצאים בנקודה (i,j) אז הצעדים המותרים הם (i,j-k),(i-k,j),(i,j+k),(i+k,j) לא ניתן לבצע צעד באלכסון.
האלגוריתם צריך להחזיר את הדרך הכי קצרה לפתרון המבוך. אם אין דרך כזאת החזר -1.

המבוך צריך להיות מיוצג ע"י מטריצה.

דוגמא:

קלט סעיף א:

N = 9, M=10

פלט סעיף א:

```
mat =    {{1, 0, 1, 1, 1, 1, 0, 1, 1, 1 },
   {1, 0, 1, 0, 1, 1, 1, 0, 1, 1 },
   {1, 1, 1, 0, 1, 1, 0, 1, 0, 1 },
   {0, 0, 0, 0, 1, 0, 0, 0, 0, 1 },
   {1, 1, 1, 0, 1, 1, 1, 0, 1, 0 },
   {1, 0, 1, 1, 1, 1, 0, 1, 0, 0 },
   {1, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
   {1, 0, 1, 1, 1, 1, 0, 1, 1, 1 },
   {1, 1, 0, 0, 0, 0, 1, 0, 0, 1 }};
S= {0, 0};
E = {3, 3};
```

קלט סעיף ב:

maze_solver(mat, S, E, 2)

המבוך בדוגמא נלקח מדוגמא סעיף א.

פלט סעיף ב:

11