שאלות חזרה למבחן 2014

במסמך זה תוכלו למצוא 5 שאלות חזרה למבחן. לכולן מבנה דומה (לא תוכן דומה) לשאלות שיופיעו בבחינה עצמה. רובן קשות יותר ממה שצפוי בבחינה. נסו לפתור אותן באמצעות דף ועט. אתם יכולים לבדוק את תוצאותיכם לאחר שסיימתם על ידי הקלדת הקוד והרצתו, אם כי זה דורש לרוב כתיבת קוד נוסף כדי להכין את הקלט לקוד שלכם.

סודוקו

המשחק סודוקו תפס פופולוריות גבוהה בישראל בשנים האחרונות. הרעיון: חידה מתמטית הדורשת הצבה של הספרות בין 1 ל 9 על גבי לוח בעל 81 תאים על פי החוקים הבאים. 1. כל ספרה צריכה להופיע בדיוק פעם אחת בכל שורה. 2. כל ספרה צריכה להופיע בדיוק פעם אחת בכל טור. 3. את הלוח נחלק לתשעה ריבועים בגודל 3*3 - בכל ריבוע כזה צריכה להופיע כל ספרה בדיוק פעם אחת.
נרצה ליישם את המשחק בקוד:

1. א. דוד הגדיר את הפונקציה init_board באופן הבא:

```
ROW_SIZE = 9
def init_board():
    row = [0]*(ROW_SIZE)
    board =[]
    for i in range(ROW_SIZE):
        board.append(row)
   return board
```

מה הטעות של דוד? תן דוגמא המסבירה כיצד קוד זה שגוי.

ב. תקן את הפונקציה init_board

2. יישם את הפונקציה

```
legal_placement(board,col,row)
```

המקבלת כקלט לוח ואת מיקום האיבר האחרון שהוסף:

```
board[c][r]
```

על הפונקציה לבדוק האם ההצבה הנוכחית חוקית. הפונקציה תניח כי כל ההצבות הקודמות חוקיות ועל כן בהנתן והספרה האחרונה שהוספה היא X, הפונקציה צריכה לבדוק האם X קיים כבר בשורה הנתונה, בטור הנתון ובתת ריבוע המתאים. במידה והוא קיים באחד מהם (או יותר) ההצבה לא חוקית והפונקציה תחזיר False. אחרת תחזיר True.

3. יישמו את הפונקציה

```
print_board(board, filename)
```

המקבלת לוח סודוקו ומדפיסה אותו לקובץ filename בפורמט הבא:
בשורה הראשונה בקובץ תודפס השורה הראשונה בלוח - כאשר פסיק מפריד בין כל תא בלוח.
לאחר השורה הראשונה עליכם לרדת שורה ולהדפיס את השורה הבאה וכו.

4. דוד רוצה ליישם את הפונקציה

```
play_game(board)
```

המקבלת לוח ריק בגודל 9 על 9, ומנסה לפתור אותו באופו רקורסיבי.
דוד החל ביישום הבא:

```
FILENAME = SUDOKU_SOL.txt

def play_game(board):
  if play_game_helper(board,0,0):
     print_board(board,FILENAME)
  else:
     print("There's no solution for this board")

def play_game_helper(board,c,r):
```

יישמו את הפונקציה play_game_helper כך שתבדוק האם ניתן לפתור את board (עליכם לשנות את הלוח המקורי במהלך הריצה).הפונקציה מחזירה ערך בוליאני חיובי במידה והלוח הוא פתיר ואחרת False
רמזים:

- האינדקסים יצביעו על השורה והטור של ההצבה הנוכחית.
- נסו לפתור באופן הבא: קדמו בכל קריאה רקורסיבית לפונקציה את אחד האינדקסים, כך שבסופו של דבר תעברו על כל התאים בלוח.
  - מה הם תנאי הקצה שלכם?
- בכל אחד מהתאים נסו להציב ספרה מסוימת. אם הספרה חוקית המשיכו לתא הבא, אחרת עברו לספרה הבאה.
- אם לא מצאתם ספרה חוקית חזרו אחורה ותקנו את התא הקודם.
- אתם רשאים להשתמש בפונקציות הקודמות שיישמתם בשאלה זו.

5. שפרו את הפונקציה שלכם כך שתוכל להתמודד עם לוח פתור בחלקו (כמו זה המופיע בעיתוני סוף השבוע).

סכומים חלקיים

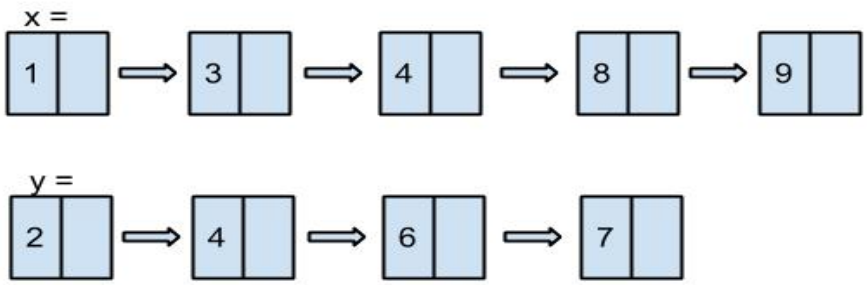
- ממשו את הפונקציה partial_sum על פי התאור שלהלן:
- הפונקציה partial_sum מקבלת רשימה lst של מספרים מטיפוס int ומספר יעד num ובודקת האם קיימת תת קבוצה של אברי lst אשר סכומם שווה ל - num.
  - אם קיימת קבוצה כזו, הפונקציה תחזיר את רשימת האיברים מתוך lst שסכומם שווה ל - num.
  - אם קיימת יותר מקבוצה אחת המספקת את התנאי, יש להחזיר קבוצה כלשהי (לא משנה איזו) ולמען הסר ספק - **אין** צורך להחזיר את כל תתי הקבוצות של lst המקיימות את התנאי.
  - אם לא קיימת קבוצה כזו, הפונקציה מחזירה None.
  - תזכורת מתורת הקבוצות - קבוצה A היא תת קבוצה של עצמה. לכן, אם סכום כל האיברים ב - lst שווה ל - num אזי זוהי תוצאה אפשרית אחת.
  - אי אפשר להניח על המספרים ברשימה או מספר היעד דבר מלבד היותם שלמים.
  - אין צורך לחשוש מגבולות הייצוג של סכום המספרים ברשימה - אפשר להניח כי סכום המספרים המקסימלי ניתן לייצוג בפייתון.
- אם הרשימה lst מכילה - n איברים, מהי סיבוכיות זמן הריצה של המימוש במקרה הגרוע ?

מיזוג רשימות מקושרות

נתונה המחלקה Node המייצגת איבר ברשימה משורשרת (סופית).

```
class Node:
    def __init__(self, data, next=None):

        self.data = data
        self.next = next
```



- נתונים שני קודקודים: X ו-Y המהווים ראש לשתי רשימות.

א. ממשו את הפונקציה merge המקבלת שתי רשימות מקושרות ומחזירה מצביע לראש הרשימה המאוחדת.
שימו לב כי אין ליצור אוביקטים חדשים - כלומר עליכם להשתמש בקודקודים של הרשימות הנתונות ולשנות בהם שדה ה- next. במקרה של שוויון בין שני ערכים משתי הרשימות, יופיע התא מx לפני התא מ-y.

def merge(x,y):

ב. מהי סיבוכיות הפונקציה שמימשתם בהינתן כי ב-x יש n>=0 איברים, וב-y יש m>=0 איברים?

משחק החיים:

אם table היא רשימה דו-מימדית N X N אז נגדיר את קבוצת השכנים של אבר table[i][j] להיות האברים בשמונה התאים ש״נוגעים״ בתא [i][j]. כדי שלכל תא יהיו בדיוק שמונה שכנים, אנחנו נניח שהתאים בשורה הראשונה נוגעים בתאים בשורה האחרונה, והתאים בעמודה הראשונה נוגעים בתאים בעמודה האחרונה.

א. השלימו את הפונקציה הבאה:

```
def alive(table, place)
    '''
    :param table - a list of N lists of size N, boolean entries
    :param place - a tuple (i,j) indexing an entry in table
    :return True if table[i][j] lives in next iteration of the game, False otherwise
    an entry table[i][j] is alive in the next iteration if it is alive now and has 2 or
    3 neighbors that are alive now, or it is dead now and it has exactly 3 neighbors
    that are alive now. In all other cases, it is dead in the next iteration.
    '''
```

בסעיפים ב-ד עליכם להקפיד שהמימוש לא יחזיק יותר ממספר קבוע של טבלאות N X N בזיכרון בכל רגע נתון.

ב. ממשו איטרטור אינסופי שהסדרה שהוא מייצר היא טבלאות כמו table לעיל, החל מטבלה התחלתית שניתנת לו באתחול. בכל צעד הטבלה החדשה מיוצרת מהטבלה הקודמת כך: תא (i,j) יכיל True אם הפונקציה alive החזירה true עבור המיקום הנ״ל והטבלה הקודמת. שימו לב כי האיטרטור יוצר טבלה חדשה בה כל המיקומים מעודכנים על פי הכלל המוגדר לעיל. לאיטרטור יש לקרוא בשם Life.

ג. מה צריך לשנות באיטרטור שלכם כדי שיעצור כאשר הטבלה הבאה זהה לקודמת? (אין צורך לכתוב את כל הקוד מחדש). עבור ההמשך, נקרא לאיטרטור הזה LifeC.

ד. מה צריך לשנות באיטרטור שלכם כדי שיעצור כאשר הטבלה הבאה זהה לטבלה הראשונה? (אין צורך לכתוב את כל הקוד מחדש) עבור ההמשך, נקרא לאיטרטור הזה LifeD.

ה. חמושים במחלקות הללו (כלומר, Life, LifeC, LifeD), ממשו איטרטור UltimateLife שעוצר כאשר הוא מגיע לטבלה שהופיעה בעבר.
(לא חייבים להשתמש בכל המחלקות הקודמות, אבל מותר אם זה מקצר את הקוד, אפילו על חשבון יעילות. אין צורך לעמוד בדרישות זמן או מקום כלשהן.)

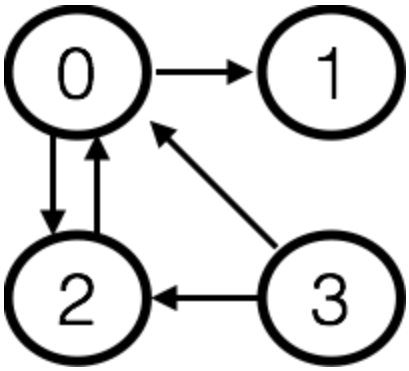
שאלת מסלולים בגרף:

גרף הוא אוסף של פריטים (כל פריט נקרא קודקוד) המכילים מצביעים (שנקראים צלעות) למספר כלשהו של צמתים אחרים. בגרף, בניגוד לרשימה משורשרת או עץ בינארי, מותר שלאותו קודקוד יצביעו כמה קודקודים אחרים וגם מותר שיווצר מעגל.

דרך אחת לייצג גרף היא באמצעות כל צלעותיו.
נניח כי קיים לנו גרף המורכב מקבוצת קודקודים:

V = {0,1,2,..,n-1}

וקבוצה של צלעות בין הקודקודים.
את קבוצת הצלעות ניצג באמצעות רשימה של רשימות שתיקרא- graph, כך שבמקום ה-i ב-graph תהיה רשימת כל השכנים של הקודקוד ה-i (שימו לב שהצלעות מכוונות, כלומר אם קודקוד X מחובר לקודקוד Y לא בהכרח Y מחובר ל - X ועל מנת לייצג קשר כפול כזה יש לעשות שימוש בצלע נוספת מ - Y ל - X).
על כן הגרף הנ״ל ייוצג באופן הבא:




```
graph[0] = [1,2]
graph[1] = []
graph[2] = [0]
graph[3] =[0,2]
```

או סה״כ:

```
graph = [[1,2,3],[],[0],[0,2]]
```

נגדיר כי קיים מסלול בין קודקוד i וקודקוד j אם ניתן ״ללכת״ על הגרף בין הקודקודים. למשל בין הקודקוד 3 לקודקוד 2 קיימים שני מסלולים: מסלול ישיר הכולל את הצלע 3->2 ומסלול עקיף הכולל שתי צלעות 3->0->2.
לעומת זאת בין קודקוד 1 לקודקוד 3 לא קיים מסלול בגרף הנ״ל.

אנה רצתה לכתוב פונקציה המוצאת עבור קודקוד i את כל הקודקודים בגרף עבורם קיים מסלול מקודקוד i אליהם. לדוגמה בשרטוט לעיל עבור הקודקוד 0 אנה רוצה למצוא את הקודקודים 2 ו - 1 כי קיים מסלול בין קודקוד 0 אליהם, אך לא את 3. כמו כן דרשה אנה כי הקודקודים מהם קיים מסלול מ - i אליהם יוחזרו באופן ממוין: על פי סדר המעבר עליהם במהלך ריצת האלגוריתם.

אנה חשבה על הרעיון הבא: עלי להחזיק שני מבני נתונים: מבנה אחד מסוג python list שיתאר את כל הקודקודים שבקרתי בהם (על פי סדר הביקור) ומבנה נוסף: מחסנית.
המחסנית תכיל את הפקודות הבאות:

```
class stack:
    def __init__(self): #init empty stack
    def push(self,data): #add data to the top of the stack
    def pop(self): #remove and return the value in the head of the stack
    def is_empty(self) # return true if the stack is empty
```

כעת רעיון האלגוריתם יהיה כנ״ל:

א. הכנס את כל שכניו של i למחסנית.

ב. כל עוד המחסנית לא ריקה:

1. הוצא קודקוד j מהמחסנית.
2. אם כבר ביקרת בקודקוד j, המשך הלאה.

אחרת,

a. הוסף את הקודקוד j לרשימת הקודקודים בהם כבר ביקרת (על ידי append)
b. הכנס את כל שכניו של הקודקוד הj למחסנית.

1. ישמו את הפונקציה dfs המקבלת כקלט graph (תקין) וקודקוד i ומחזירה רשימה של כל הקודקודים עבורם קיים מסלול בין i ובינם. על הרשימה להיות ממוינת על פי סדר המעבר עליהם בפונקציה. במידה וקודקוד i אינו בגרף על הפונקציה להחזיר None.

```
def dfs(graph,i):
```

2. מה סיבוכיות הפונקציה? הניחו כי סיבוכיותן של כל הפעולות על המחסנית הן O(1), פעולת הוספה לרשימה וגישה לאינדקס היא בסיבוכיות O(1), ופעולת בדיקה האם איבר ברשימה היא בסיבוכיות ליניארית בגודל הקלט O(n). **שימו לב כי מדובר בסעיף קשה במיוחד. נסו לחשוב מה תהיה הסיבוכיות במידה וזהו גרף מלא (כלומר כל קודקוד מחובר לכל הקודקודים). מעבר לכך אתם מוזמנים לקרוא בויקיפדיה על DFS.**

3. כיצד תשפרו את זמן ריצת הפונקציה כך שתוכלו להמנע מהפעולה היקרה של בדיקה האם איבר ברשימה? ביכולתכם להגדיר מבנה נתונים נוסף לצורך זה. אין צורך לכתוב מחדש את יישום הפונקציה אלא רק להסביר כיצד תשנו את הקוד בכדי לענות על דרישה זו. הסבירו מה תוסיפו ומה תשנו.

4. כעת לאנה אין מחסנית אלא יש ברשותה תור - מבנה נתונים בו איברים נכנסים על פי ראשון נכנס ראשון יוצא (FIFO). הסבירו בכתב, האם עדיין ביכולתה של אנה למצוא את כל המסלולים בגרף ואם כן איך תושפע התוצאה? אין צורך לכתוב קוד בתשובה זו.