מבוא למדעי המחשב 67101

תרגיל 10 - רשת ויקיפדיה

להגשה בתאריך 30/12/2015 בשעה 22:00

בתרגיל זה נבנה רשת שמייצגת את מערכת המאמרים בויקיפדיה. לכל מאמר יש שם, שהוא הכותרת של המאמר, וקבוצה של כל המאמרים שהוא מפנה אליהם בלינק. בתרגיל זה ממשו את כל הפונקציות בקובץ ex10.py. טרמינולגיה: לשם פשטות בתרגיל זה נקרא למאמרים שמאמר כלשהו A מפנה אליהם "שכנים שיוצאים מA" ולמאמרים שמפנים לA נקרא "שכנים שנכנסים לA".

משימה 1 - בניית רשת ויקיפדיה

במשימה זאת עליכם לבנות את מבנה הנתונים של רשת ויקיפדיה.לשם כך תסתייעו בקובץ אשר מכיל חלק ממבנה הרשת ויקיפדיה באנגלית. המידע מוצג ע"י רשימה של זוגות מאמרים על פי הפורמט הבא:

articleA articleB
articleA articleC
articleC articleH
⋮

כאשר כל זוג מאמרים מופרד בטאב (לא מופרד ברווח).
כאשר הרשומה articleA articleB מציינת שיש הפניה (לינק) מarticleA ל articleB (שימו לב שזה לא מחייב שתהיה הפניה מ articleB ל articleA). שימו לב שבכדי להפריד את המאמרים בכל זוג ניתן להשתמש בפקודה split. כלומר,

```
text.split('\t')
```

ובכדי להפריד בין שורות הזוגות השתמשו ב-

```
text.split('\n')
```

כאשר text הוא תוכן הקובץ.
בכדי לקרוא את הקובץ links.txt עליכם לפתוח אותו ע"י הפונקציה:

```
f=open(name, 'r')
```

אשר מקבלת את שם הקובץ ופרמטר נוסף 'r' אשר מציין שהקובץ מיועד לקריאה. הפונקציה מחזירה אובייקט של קובץ הניתן לקריאה. שימו לב שבכדי שהפונקציה תפתח את הקובץ כראוי, קובץ המידע צריך להימצא בתיקיית העבודה שבה נמצא קובץ התכנית שלכם.
בכדי לקרוא את תוכן הקובץ השתמשו בread() אשר ממירה (ומחזירה) את תוכן הקובץ למחרוזת.

כתבו פונקציה:

```
def read_article_links(file_name):
```

אשר מקבלת את שם הקובץ ומחזירה רשימת כל זוגות המאמרים, כאשר כל זוג הוא רשומה (tuple) של שני שמות של המאמרים (כלומר בכל רשומה שני איברים), כפי שמוצג בפורמט הקובץ.

בנו מחלקה עם קונסטקטור שמקבל את שם המאמר באופן הבא:

```
class Article:
        def __init__(self, name):
```

באוביקט מסוג Article נשמר collection (רשימה\רשומה\קבוצה\מילון) המאמרים השכנים היוצאים (Articles).

בנוסף ממשו את המתודות הבאות של המחלקה Article:

```
def get_name(self):
```

אשר מחזירה את שם המאמר.

```
def add_neighbor(self, neighbor):
```

אשר מוסיפה אובייקט `Article` לcollection של מאמר.

```
def get_neighbors(self):
```

אשר מחזירה רשימה של שכניו של מאמר. (רשימת אובייקטים מסוג Article)

וכן,

```
def __repr__(self):
```

אשר מחזירה מחרוזת שמייצגת את שם המאמר, הסימן , ולאחריו רשימת שמות השכנים (כלומר זהו פורמט של tuple). לדוגמא, אם 'a' הוא שם מאמר ולו שכנים יוצאים 'b' ו'c' אז נסמן את האיבר ב

`('a', ['b', 'c'])`

וכן את הפונקציות:

```
def __len__(self):
```

אשר מחזירה את מספר השכנים של מאמר.

```
def __contains__(self, article):
```

אשר מקבלת אובייקט של מאמר ומחזירה האם **אובייקט המאמר** הוא שכן יוצא.

כעת נגיע לחלק העיקרי של בניית הרשת. לצורך זה עליכם לייצר מחלקה שמכילה את Article של כל המאמרים. המחלקה נראית כך:

```
class WikiNetwork:
```

בניית הרשת תעשה ע"י הבנאי:

```
    def __init__(self, link_list = []):
```

אשר מקבל את רשימה של כל הזוגות בקובץ (שנוצרה ע"י הפונקציה `read_article_links`) ומייצר collection (כלומר רשימה/מילון/קבוצה) של מאמרים (מסוג Article שהוגדר למעלה) כך שלכל זוג, המאמר ששמו שנמצא באיבר השני בכל זוג הוא שכן של המאמר ששמו נמצא באיבר הראשון בזוג. כלומר, המאמר השני ימצא ברשימת השכנים של המאמר הראשון.

הנחיה: גישה לאיברים במילון (dictionary) בפייתון היא מהירה (נעשית בזמן קבוע, ללא תלות בגודל המילון) ולכן מומלץ להשתמש במילון בכדי לייעל את בניית הרשת.

ממשו את המתודות הבאות של `WikiNetwork` :

```
def update_network(self,link_list):
```

המקבלת רשימה של זוגות של מאמרים (בדומה למה שמחזיקה הפונקציה `read_article_links`) בדומה למה שהוגדר ב__init__, ומעדכנת לפיה את הרשת. **שימו לב, פונקציה זאת לא מהווה תחליף ל__init__, באופן זה שהיא לא מייצרת רשת חדשה.**

```
def get_articles(self):
```

אשר מחזירה רשימה של כל ה- Articles של מאמרים ברשת.

וכן את,

```
def get_titles(self):
```

שמחזירה את רשימת שמות של כל המאמרים ברשת.

```
def __contains__(self,article_name):
```

שמחזירה האם נמצא ברשת מאמר **לפי שמו**.

```
def __len__(self):
```

שמחזירה את מספר המאמרים ברשת.

```
def __repr__(self):
```

אשר מחזירה מחרוזת שמייצגת את הרשת. מחרוזת תוחזר ככמחרוזת המייצגת מילון שמפתחותיו הם שמות של מאמרים, וערכיו הם מאמרים. לדוגמא, רשת המכילה מאמרים בשמות 'a', 'b', 'c', יכולה להיות מיוצגת ע"י המחרוזת:

```
{'a': ('a', ['b', 'c']), 'b': ('b', ['a']), 'c': ('c', [])}
```

וכן,

```
def __getitem__(self,article_name):
```

אשר מקבלת שם של מאמר ומחזירה את אובייקט המאמר המתאים לשמו. הדבר הנכון לעשות במידה ולא נמצא מאמר עם שם זה הוא להעלות שגיאה ע"י כתיבת השורה הבאה:

raise KeyError(article_name)

התנהגות זו תואמת להתנהגות של מילון של פייתון, כשמבקשים ערך לפי מפתח שלא קיים במילון. כיון שלא לימדנו בהרצאות על העלאת שגיאה, אנחנו לא דורשים טיפול כלשהו במקרה שבו לא נמצא ברשת מאמר מתאים. מומלץ לטפל במקרה זה כמו שצריך, אך אין חובה. אין צורך, וגם לא מומלץ, לטפל במקרה זה ע"י הדפסת ערך שגיאה, או החזרה מכוונת של ערך כלשהו.

ניתן להוסיף אטריביוטים (משתנים) או פונקציות נוספות למחלקות Article ו-WikiNetwork.

<u>משימה 2 - Page Rank</u>

כאשר מבצעים חיפוש בגוגל, מתקבלת רשימת האתרים שמוצגת בסדר מסוים. קביעת הסדר נעשית בין היתר ע"י אלגוריתם Page Rank (שנקראית על שם Larry Page - אחד ממייסדיה של גוגל) אשר מאפשר לדרג "חשיבות" של האתרים ברשת בדרך מסויימת. הנחת האלגוריתם היא שלאתרים חשובים יש מספר רב יותר של לינקים שמפנים אליהם (לינקים נכנסים) מאתרים פחות חשובים. רעיון האלגוריתם הוא נסיון להתחקות אחר משתמש אקראי ש"גולש" מאתר לאתר ע"י לחיצה על הלינקים.

האלגוריתם ניתן לתאור באופן הבא: לכל מאמר i בזמן t יש כמות כסף $PR^t(i)$, אשר מסמלת את מידת החשיבות שלו. בתחילה לכל מאמר יש שקל אחד, כלומר $PR^0(i) = 1$ לכל מאמר i. האלגוריתם הוא איטרטיבי וכל איטרציה מורכבת משני שלבים:

1. כל מאמר מחלק d (אשר מוגדר להיות d=0.9 כברירת מחדל) מהכסף שברשותו באופן שווה בשווה לכל המאמרים שהוא מפנה אליהם (כלומר $d\frac{PR^t(A)}{OUT(A)}$ כאשר $PR^t(A)$ הוא הכסף של מאמר A בזמן t, ו $OUT(A)$ הוא מספר השכנים היוצאים של A) ומהחלק הנותר הוא מחלק מהכסף לכל שאר המאמרים (כלומר לא רק השכנים) באופן שווה , כך שלמעשה יוצא שכל מאמר "משאיר אצלו" 0.1 (ראו ע"פ המשוואת למטה) כל מאמר מעדכן את הכסף ע"י סיכום הכסף שנתקבל מכל מהמאמרים **השכנים** שמפנים אליו , ובנוסף את החלק הנותר של **כל** המאמרים.

שימו לב: אין צורך לחשב את החלק שנותר מכל המאמרים. סכום זה מסתכם ל $1-d$ מהכסף ההתחלתי בכל שלב ולכל מאמר (ראו ע"פ המשוואות למטה).
ניתן להניח שלכל המאמרים יש לפחות מאמר אחד שהם מפנים אליו.

ניתן לתאר את סכום הכסף שכל מאמר מקבל בזמן t+1 כפונקציה של סכום הכסף של המאמרים בזמן T באופן הבא:

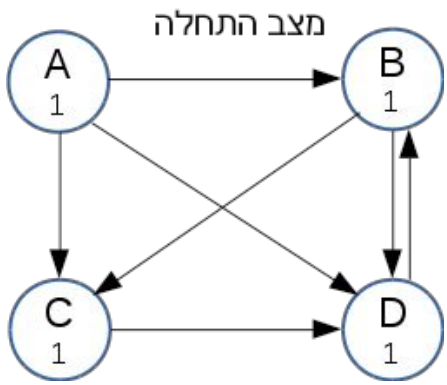
$$PR^{t+1}(A) = d\left(\frac{PR^t(B)}{OUT(B)} + \frac{PR^t(C)}{OUT(C)} + \ldots\right) + \frac{1-d}{n}\left(PR^t(A) + PR^t(B)\ldots\right)$$

$$= d\sum_{i\in N(A)} \frac{PR^t(i)}{OUT(i)} + \frac{1-d}{n}\sum_i PR^t(i)$$

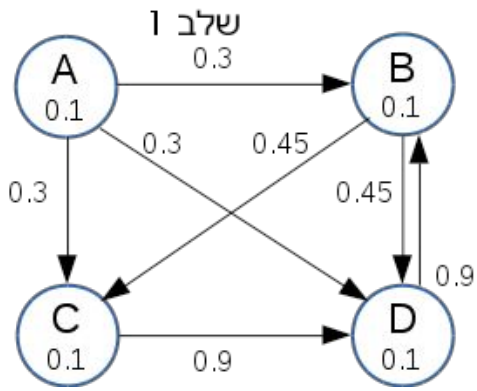
$$= d\sum_{i\in N(A)} \frac{PR^t(i)}{OUT(i)} + (1-d)$$

כאשר N(A) זאת קבוצת השכנים הנכנסים של A, ו OUT(i) מסמן את **מספר** השכנים היוצאים של מאמר i, ו $PR^t(i)$ מסמן את הדירוג של מאמר B בזמן T, ו-n זה מספר המאמרים ברשת. **המעבר מהמשוואה השניה לשלישית הוא נכון כיוון שסכום הכספים של כל המאמרים בכל איטרציה הוא n.**
מכיוון ש $1-d=0.1$ הוא סכום קבוע נקרא לו "היתר".

דוגמא: להלן רשת של 4 מאמרים. כל עיגול מייצג מאמר וחץ מעיגול i לעיגול j מייצג הפניה ממאמר i למאמר j.



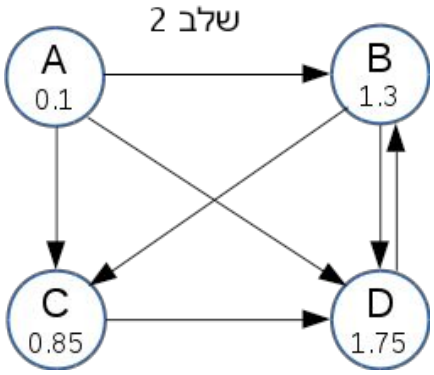
מצב התחלה - כל המאמרים מתחילים עם שקל בודד.



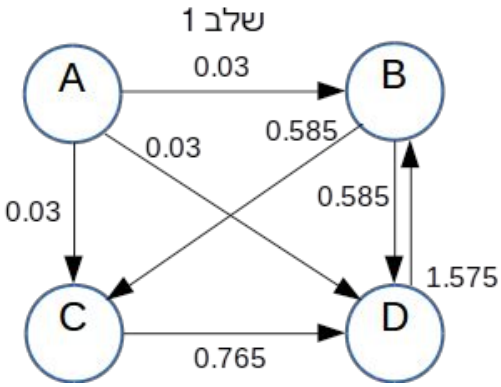
<u>איטרציה 1, שלב 1</u> - כל מאמר מחלק 0.9 שקלים (כיוון שבחרנו d=0.9) באופן שווה לכל שכניו. בהתאם לכך A מחלק 0.3 לכל אחד משלושת משכניו, מאמר B מחלק 0.45 לכל אחד משני שכניו, ומאמרים C וD מעבירים 0.9 במלואם לשכן הבודד שלהם. באיור מסומן 0.1 בכל מאמר שמסמן שכל מאמר מחלק 0.1=$1-d$ לשאר המאמרים (לא רק שכנים),ופועל יוצא מכך הוא שבשלב 2 מתווסף לכל מאמר 0.1.

<u>איטרציה1, שלב 2</u> - הסכומים של המאמרים הבאים מצטברים באופן הבא:

- מאמר A נותר עם יתר של 0.1 משום שאין לו אף שכן נכנס שמעביר לו כסף.
- מאמר B מקבל 0.3 מA, ו0.9 מD. מכיוון שB מקבל בנוסף יתר 0.1, כעת יש לB בסה"כ 1.3 שקלים.
- מאמר C מקבל 0.3 מA, ו0.45 מB. מכיוון שC מקבל בנוסף יתר 0.1, כעת יש לC בסה"כ 0.85 שקלים.
- מאמר D מקבל 0.3, מקבל 0.45 מB ו0.9 מC. מכיוון שD מקבל יתר 0.1, כעת יש לD בסה"כ 1.75 שקלים.

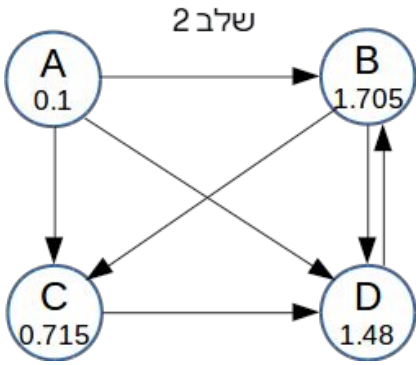


<u>איטרציה 2, שלב 1</u> - כל מאמר i מחלק $PR^1(i) * d$ (כלומר מחלק את הסכף שצבר באיטרציה 1 כפול 0.9), באופן שווה לכל שכניו. בהתאם לכך A מחלק 0.03 לכל אחד משלושת משכניו, מאמר B מחלק 0.585 לכל אחד משני שכניו, מאמרים מעביר 0.765 לD, ו-D מעביר 1.575 לB. בנוסף כל מאמר מחלק שווה בשווה בין כל המאמרים 0.1*הכסף שברשותו.



<u>איטרציה2, שלב 2</u>- הסכומים של המאמרים הבאים מצטברים באופן הבא:

- מאמר A נותר עם יתר של 0.1 משום שאין לו אף שכן נכנס שמעביר לו כסף
- מאמר B מקבל 0.03 מA, ו1.575 מD. מכיוון שBמקבל בנוסף יתר 0.1, כעת יש לB בסה"כ ~~1.3~~ **1.705** שקלים
- מאמר C מקבל 0.03 מA, ו0.585 מB. מכיוון שC מקבל בנוסף יתר 0.1, כעת יש לC בסה"כ 0.715 שקלים
- מאמר D מקבל 0.03, מקבל 0.585 מB ו0.765 מC. מכיוון שD מקבל בנוסף יתר 0.1, כעת יש לD בסה"כ 1.48 שקלים.



ממשו את הפונקציה:

```
def page_rank(self,iters,d=0.9):
```

אשר מקבלת מספר שלם iters שמייצג את מספר האיטרציות ש-pagerank מופעל, ומספר ממשי d בין 0 ל-1 , שמייצג את גודל התרומה למאמרים השכנים. הפונקציה ולאחר מכן מחזירה את רשימת שמות המאמרים בסדר ממוין מהגדול לקטן לאחר ביצוע page rank, במידה וישנו יותר ממאמר אחד עם אותו דירוג, אז החזירו אותו לפי סדר לקסיקורפי מהקטן לגדול . לדוגמא, הפונקציה עשויה להחזיר רשימה כזאת:

```
['b' (=2),'aaa' (=0.8),'aba' (=0.8),'bbb' (=0.8),'a' (=0.6)]
```

כתבו את שלושת המאמרים עם הpage rank הגבוה ביותר בREADME **עם הפרמטרים d=0.9 ו iters=50**
(למרות שגם עבור מספר הרבה יותר קטן של איטרציות נקבל את אותה תוצאה)

<u>משימה 3 - אינדקס ג'אקרד</u>

אינדקס ג'אקרד הוא מדד לקירבה של שתי קבוצות אשר מוגדר ע"י גודל החיתוך של שתי קבוצות חלקי גודל האיחוד שלהן כלומר עבור קבוצות A וB אינדקס ג'אקרד שלהם מוגדר כך: $\frac{|A \cap B|}{|A \cup B|}$.

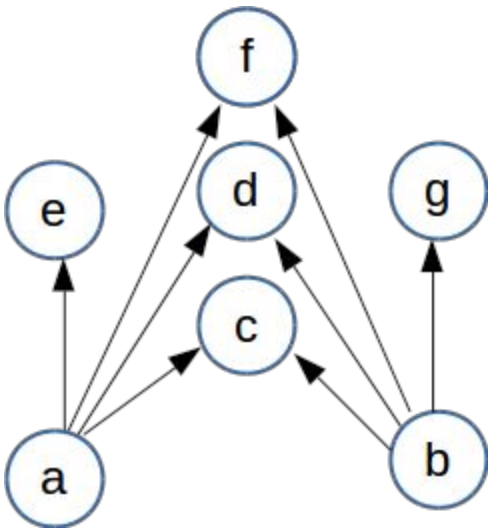
קצת אינטואיציה: אינדקס ג'אקרד מסמן קירבה בין שני מאמרים באופן זה שלמאמרים "דומים" יש יחסית הרבה שכנים יוצאים משותפים לשניהם (ולכן החיתוך), מצד שני ככל שלמאמר יש יותר שכנים יוצאים יש לו סיכוי רב יותר לשכנים משותפים עם מאמר אחר ללא קשר לדימיון בינהם, ולכן מנרמלים (מחלקים) באיחוד השכנים היוצאים (ראו איור למטה).

נרצה להגדיר את אינדקס ג'אקרד של שני מאמרים באופן דומה. החיתוך של מאמרים A וB הוא קבוצת כל המאמרים שA **וגם** B מפנים אליהם (כלומר כל השכנים שיוצאים מ-A וגם מ- B). האיחוד של שני מאמרים הוא קבוצה כל המאמרים שA **או** B מפנים אליהם (כל השכנים שיוצאים מ-A או מ-B). דוגמא (ראו איור):
ממאמר a יוצאים שכנים c,d,e,f וממאמר B יוצאים שכנים c,d,f,g. במקרה זה החיתוך בינהם הוא c,d,f, כלומר גודל החיתוך הוא 3, והאיחוד שלהם הוא c,d,e,f,g, כלומר גודל האיחוד הוא 5. לכן אינדקס ג'אקרד של מאמר a ומאמר b הוא 3/5.
ממש את הפונקציה

```
def jaccard_index(self, article_name):
```

שמקבלת שם של מאמר, ומחזירה רשימה ממוינת של כל **שמות** המאמרים הממוינים לפי אינדקס ג'אקרד מהגדול לקטן (כלומר האיבר הראשון ברשימה הוא המאמר בעל האינדקס ג'אקרד הגדול ביותר מבין כל המאמרים, האיבר השני הוא בעל ההאינדקס ג'אקרד השני הגדול ביותר וכו'). שימו לב שאינדקס ג'אקרד של מאמר עם
עצמו הוא 1. במקרה שבו יש כמה מאמרים בעלי אותו אינדקס ג'קארד, המאמרים מוחזרים הללו ממויינים לפי סדר לקסיקורפי מהקטן לגדול. במידה ושם המאמר לא נמצא ברשת, או שלמאמר ברשת עם שם article_name אין אף צלע יוצא החזירו None.

מה האינדקס ג'אקרד **השני** הגבוה ביותר **(כלומר הגבוה ביותר שהוא לא המאמר עצמו)** למאמרים "United_Kingdom", "Israel", "United_States" ו- "Algebra", "World_War_II"? כתבו את התוצאות שלכם בREADME.



משימה 4 - טיול ברשת

המעבר ממאמר א' למאמר ב' חוקי אם יש הפניה ממאמר א' למאמר ב'. טיול ברשת הוא סדרה של מעברים חוקיים על מאמרים. נגדיר דרגת כניסה של מאמר, כמספר השכנים שנכנסים אליו.

טיול עולה ברשת הוא מעבר חוקי על המאמרים כאשר בכל צעד המעבר נקבע על פי הקריטריון הבא:

1. נבחר את המאמר שלו הדרגת כניסה הגבוהה ביותר במידה והוא יחיד
2. אם יש יותר ממאמר אחד כזה אז נבחר את המאמר שלשם שלו יש את הערך הנמוך ביותר לפי הסדר הלקסיקוגרפי .

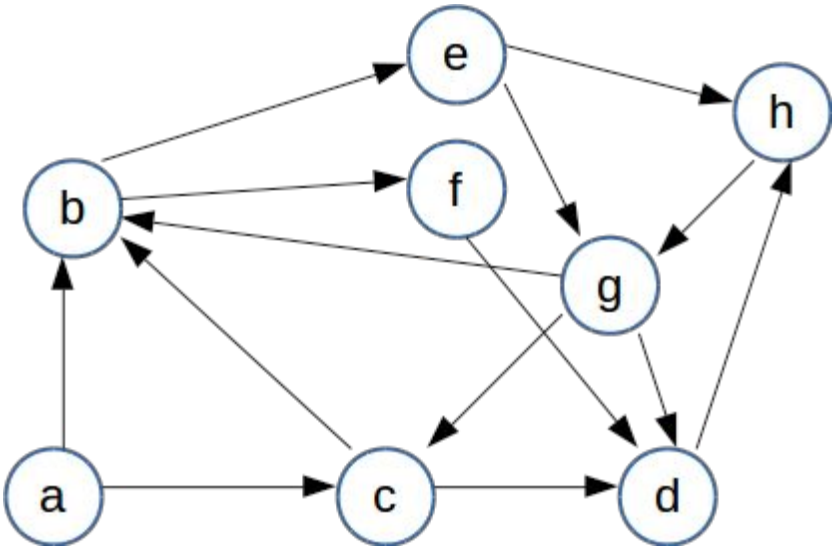
הטיול מסתיים (זה לא מובטח) כאשר מגיעים למאמר שאין לו שכנים יוצאים. שימו לב שבכדי למצוא את דרגת הכניסה של אובייקט מאמר ברשת, יש צורך לעבור על כל מאמרי הרשת.

דוגמא: נתונה לנו הרשת להלן (באיור). נניח שאנחנו מתחילים את הטיול במאמר a. לאחר מכן ניתן לעבור למאמרים b וc. המאמר הבא יהיה b מכיוון שיש לו דרגת כניסה 3 ואילו למאמר c יש דרגת כניסה 1. לאחר שהגענו לb ניתן לעבור למאמרים e ו f. לשני המאמרים הללו אותה דרגת כניסה, ולפי הסדר הלקסיקוגרפי e הוא נמוך מf, ולכן הבא יהיה מאמר e. המאמר הבא שייבחר יהיה g (ע"פ דרגת כניסה) ולאחריו b (גם כן ע"פ דרגת כניסה) .

ממשו את הפונקציה:

```
def travel_path_iterator(self , article_name):
```

אשר מקבלת שם של מאמר ומחזירה איטרטור לפי סדר את שמות המאמרים בטיול. במידה והטיול מסתיים, האיטרטור תפסיק לייצר ערכים להחזרה. **אין מניעה שהטיול ייכנס ללולאה אינסופית.** במידה ושם המאמר לא קיים ברשת, תוחזר איטרטור של טיול עם מסלול ריק, שאינה מחזירה שמות מאמרים. **המאמר שהפוקנציה מקבלת צריך להיות המאמר הראשון במסלול. שימו לב שגנרטור מייצר איטרטור.**



משימה 5 - חברים ממרחק d

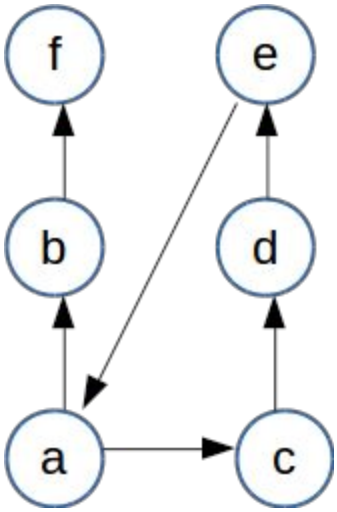
מעבר ממאמר א' למאמר ב' הוא חוקי אם יש הפניה ממאמר א' למאמר ב' (כמו ההגדרה במשימה 4). חבר ממרחק d של מאמר i, הוא מאמר אשר ניתן להגיע אליו ממאמר ב d מעברים חוקיים **לכל היותר**. לדוגמא (ראו איור) קבוצת כל החברים מממרחק 2 של a, הם b,c,d,f. מאמר e אינו חבר ממרחק 2, משום שניתן להגיע אליו רק ע"י 3 מעברים חוקיים. נגדיר חבר מדרגה 0 של מאמר הוא המאמר עצמו (ולכן מאמר הוא חבר של עצמו מכל דרגה).

ממשו את הפונקציה:

```
def friends_by_depth(self, article_name, depth):
```

אשר מקבלת שם של מאמר ומספר שלם חיובי ומחזירה את רשימת שמות כל החברים ממרחק d שלה. במידה ושם המאמר לא נמצא ברשת הפונקציה מחזירה None.

רמז: מומלץ לבצע זאת ע"י שימוש בפונקציה פנימית אשר מקבלת קבוצת מאמרים A, ומחזירה את האיחוד של כל קבוצות השכנים של כל המאמרים A.



שאלות:
מה אחוז המאמרים מכלל המאמרים ברשת הם במרחק 1 מChristopher_Columbus?
מה אחוז המאמרים מכלל המאמרים ברשת הם במרחק 2 מDNA?
מה אחוז המאמרים מכלל המאמרים ברשת הם במרחק 3 מHistory?
כתבו את התוצאה שקיבלתם בREADME

הוראות הגשה

בתרגיל זה עליכם להגיש את הקבצים הבאים:

1. ex10.py – עם המימושים שלכם לפונקציות.
2. README על פי הפורמט שמפורט בנהלי הקורס.

יש להגיש קובץ zip הנקרא ex10.zip המכיל בדיוק את שני הקבצים הנ"ל.