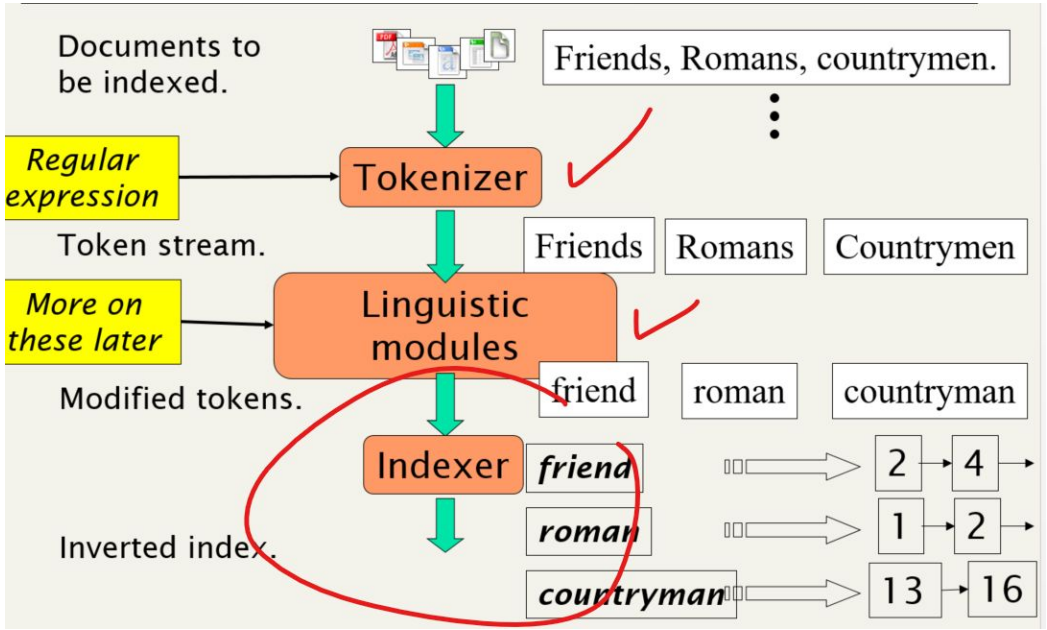
מבוא לאיחזור מידע

מעבדה 2 המשך - מודל ייצוג המסמכים ובניית שאילתה

לאחר החלטה על שלבי עיבוד מקדים ויצירת מילון מצומצם .....



יצוג מסמכים:

- מטריצה
- אינדקסים הפוכים
- שיפורים שונים לאינדקסים הפוכים

בניית שאילתה:

שאילתה למטריצה

שאילתה לאינדקסים הפוכים

בניית אופן הייצוג למסמכים

במעבדה נשווה בין דרכים שונות לייצג את אוסף המושגים שלנו ביחס למסמכים לטובת חיפוש.

הגבילו את גודל הייצוג ל 100 טוויטים ראשונים

שימרו את התוצאה בקובץ (כטקסט או כ python pickle לדוגמא)

- מהו גודל הקובץ ?
- מהו אחוז המידע הרלוונטי בייצוג (לדוג' אחדות במטריצה)?

בדקו את התוצאות עבור 1000 טוויטים

- מהו גודל הקובץ ?
- מהו אחוז המידע הרלוונטי בייצוג (לדוג' אחדות במטריצה)?

בניית שאילתה

השאילתה הינה פונקציה המקבלת כקלט את הביטוי לחיפוש (מילה אחת או יותר) ומחזירה את מספרי הטוויטים הרלוונטיים.

על השאילתה לקחת בחשבון את אותם תהליכי עיבוד מקדים שבוצעו על המילון.

עליכם לממש עבור כל צורת ייצוג, פונקציית חיפוש מתאימה.

יש להציג מדידת זמני חיפוש עבור ביטויים תואמים בכל צורת ייצוג (לדוגמא השוואת זמנים בין חיפוש במטריצה לחיפוש באינדקסים הפוכים).

יצירת מטריצה בוליאנית - מסמכים למול מושגים

צרו מטריצה בוליאנית כאשר העמודות מייצגות את המסמכים ושורות עבור המושגים (השורות הן המילון שבנינו).

ערכי המטריצה יהיו אחד או אפס בהתאם. (בבחירת מבנה הנתונים יש לחשוב על השאילתה)

יצירת מטריצה בוליאנית - שאילתה

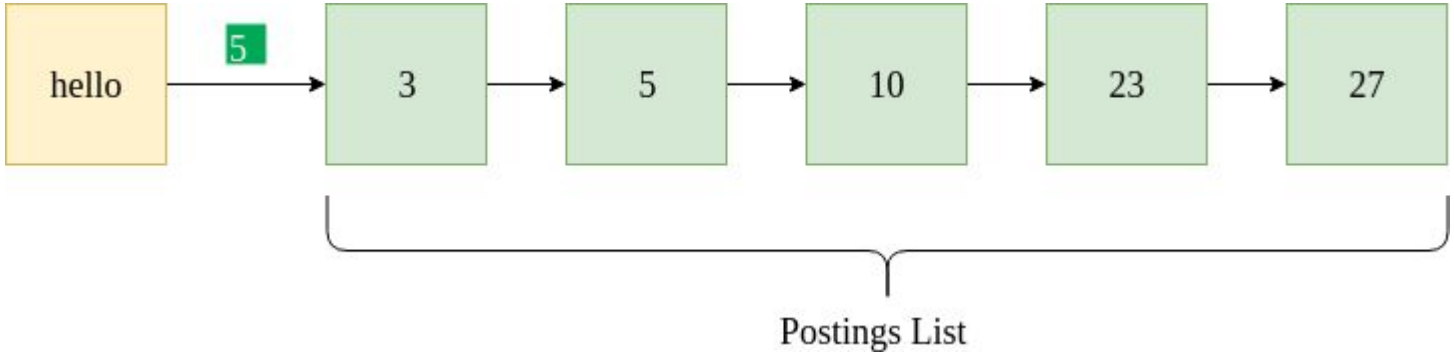
כתבו פונקציה המחזירה את מספרי הטוויטים עבור מושג או אוסף מושגים (OR,NOT,AND)

שימו לב ניתן לממש זאת כסוג של bitwise operations בין ווקטורים המייצגים מושגים שונים

יצירת מערך אינדקסים הפוכים

צרו או השתמשו במבנה נתונים דינאמי, אשר עבור כל מושג יחזיק את כמות הטוויטים בהם הוא מופיע, ואת מספרי הטוויטים בהם הוא מופיע.

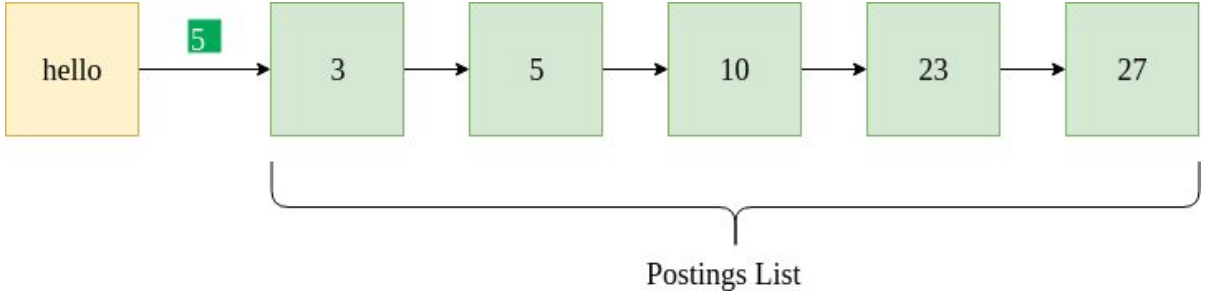
שימו לב לאופן השמירה של הטוויטים השונים (סדר)



יצירת מערך אינדקסים הפוכים - שאילתה #1

כתבו פונקציה המחזירה את מספרי הטוויטים עבור מושג או אוסף מושגים (OR,NOT,AND)

שימו לב יש לממש merge כפי שנלמד בכיתה, ולהעזר בעובדה שיש לנו תדירות מסמכים עבור כל מושג (שקפים 19-28 מהרצאה)



יצירת מערך אינדקסים הפוכים - שאילתה #2

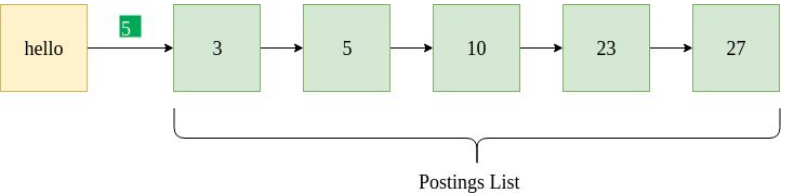
שפרו את מבנה הנתונים
הוסיפו למבנה הנתונים גם מצביעי קפיצה (skip pointers) לצורך ייעול השאילתה.

שפרו את השאילתה מסעיף קודם כך שיהיה ניתן להעזר במצביעים (או בהקבלה שלהם בפייתון).

יש לבחור שלושה גדלים שונים של קפיצות :

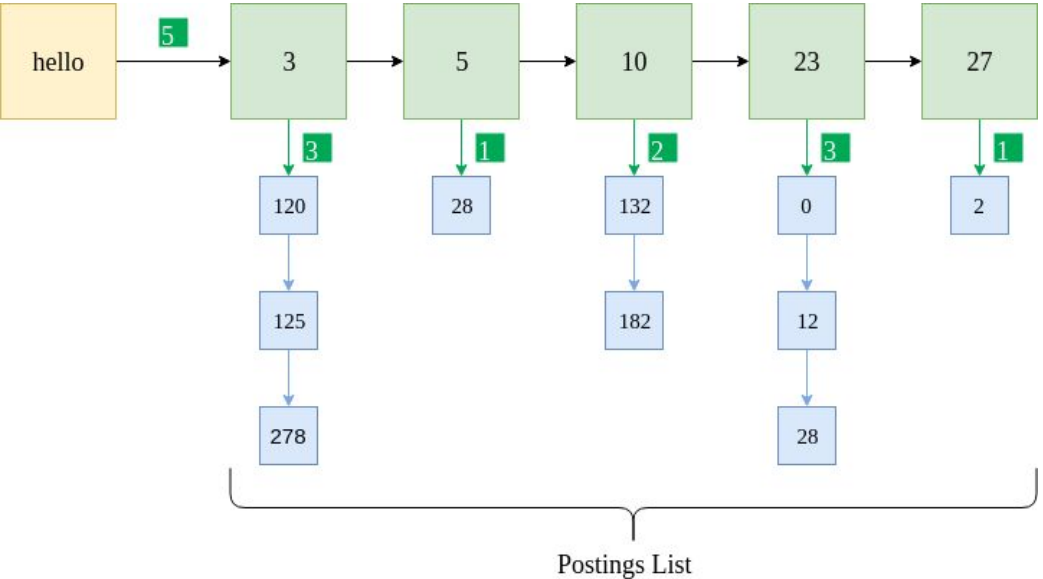
- יש להציג את הגדלים השונים של מבנה הנתונים
- יש להציג את זמני החיפוש (הערך האמצעי - צריך לעמוד בכלל שניתן בהרצאה - עמ 44)

על איזה עוד מדד אנו משלמים בבחירת גודל הקפיצה (Insertion cost) , חישבו על דרך למדוד אותו. (סיכום תיאורטי)



יצירת מערך אינדקסים הפוכים עם מיקומים

שפרו את מבנה הנתונים מהסעיף הקודם על ידי הוספת תדירות המושגים בכל מסמך,
והוספת המיקום של המושג (מיקום התו שמתחיל את המילה בכל מסמך)



שימוש ב hashedindex

In [1]:
```
#pip install hashedindex
```

In [2]:
```
 #from collections import Counter
```

In [3]:
```
import hashedindex

index = hashedindex.HashedIndex()

index.add_term_occurrence('hello', 'd1')
index.add_term_occurrence('world', 'd1')

index.get_documents('hello')


index.items()


example = 'The Quick Brown Fox hello hello Jumps Over The Lazy Dog'

for term in example.split():
    index.add_term_occurrence(term, 'd2')
```

In [4]:
```
print(index.items())
```

```
{'hello': Counter({'d2': 2, 'd1': 1}), 'world': Counter({'d1': 1}), 'The': Counter({'d2': 2}), 'Quick': Counter({'d2': 1}), 'Br
own': Counter({'d2': 1}), 'Fox': Counter({'d2': 1}), 'Jumps': Counter({'d2': 1}), 'Over': Counter({'d2': 1}), 'Lazy': Counter
({'d2': 1}), 'Dog': Counter({'d2': 1})}
```