שיעורי בית מספר 5

שאלה 3

א) במקרה הטוב: כל הספר מורכב רק ממילה אחת (למשל, "הודור"), ואז הפונקציה תרוץ על הספר פעם אחת ותספור כמה פעמים היא מופיעה. עם len(words) = n, הסיבוכיות תהייה O(n).
במקרה הגרוע הספר מורכב מ-n מילים שונות, ואז הפונקציה תרוץ על הספר n פעמים, על n מילים כל פעם. כאן הסיבוכיות תהייה O(n**2).
לכן הערכתי שייקח זמן רב (מספר דקות) עד שהפונקציה תסיים לרוץ על הספר. בפועל זה לקח עשר שניות.

ז)

```
>>> from urllib.request import urlopen
>>> alice = download ("http://www.gutenberg.org/cache/epub/11/pg11.txt")
>>> words = clean(alice).split()
>>> word_cnt = count_words(words)
>>> word_cnt_sorted = sort_by_cnt(word_cnt)
>>> word_cnt_sorted[:10]
[['the', 1818], ['and', 940], ['to', 809], ['a', 690], ['of', 631], ['it', 610],
 ['she', 553], ['i', 545], ['you', 481], ['said', 462]]
>>>
```

ח) לדעתי 200 אינו המספר האופטימלי לטקסט הנוכחי. בטקסט יש 3007 מילים שונות. במקרה הטוב יהיו 15 מילים בתא, מה שיאריך את זמן החיפוש.
בדקתי עבור טבלה בגודל 3007 ומצאתי שהגודל האופטימלי לטבלה, בה המילים יתחלקו למספר התאים הרב ביותר הוא 1901 . (כלומר בטבלה בת 3007 היו 1106 תאים ריקים).
ראוי לציין ש"מספר אופטימלי" תלוי בצורך של המתכנת – האם יותר קריטי לחסוך במקום, או שהתכנית תרוץ מהר? בתשובה שנתתי התייחסתי לאפשרות השניה.

שאלה 5

אני לא בטוח, אבל אני חושב שזה מרכז עלית לספורט באוניברסיטת ת"א.

כיצד פועל האלגוריתם:

בהתחלה האלגוריתם מרכז את כל החלקים (המטריצות) הקטנות בטבלה.
לאחר מכן הוא בוחר חלק באופן שרירותי, ועובר על כל החלקים הטבלה עד להרכבת טור בן עשרים חלקים, ע"פ השורות החופפות של החלקים.
במקביל, הוא מסיר גם שורות מיותרות מהחלקים.
לאחר סיום הרכבת הטור הוא בוחר בחלק שרירותי וחוזר על התהליך, עד לקבלת עשרים טורים.

לאחר מכן הוא בוחר טור באופן שרירותי, ובאופן דומה לחלק הראשון, ע"פ הטורים החופפים, הוא בונה את התמונה השלמה, תוך כדי שהוא מסיר את הטורים המיותרים.

לאחר שהתאים את כל הטורים, הוא מחזיר את התמונה השלמה.