אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2021

תרגיל בית מספר 3 - להגשה עד 24 באפריל בשעה 23:55

קיראו בעיון את הנחיות העבודה וההגשה המופיעות באתר הקורס, תחת התיקייה assignments. חריגה מההנחיות תגרור ירידת ציון / פסילת התרגיל.

הגשה:

- תשובותיכם יוגשו בקובץ pdf ובקובץ py בהתאם להנחיות בכל שאלה.
- השתמשו בקובץ השלד skeleton3.py כבסיס לקובץ ה py אותו אתם מגישים.
  לא לשכוח לשנות את שם הקובץ למספר ת"ז שלכם לפני ההגשה, עם סיומת py.
- בסה"כ מגישים שני קבצים בלבד. עבור סטודנטית שמספר ת"ז שלה הוא 012345678 הקבצים שיש להגיש הם hw3_012345678.pdf ו- hw3_012345678.py.
- מכיוון שניתן להגיש את התרגיל בזוגות, עליכם בנוסף למלא את המשתנה SUBMISSION_IDS שבתחילת קובץ השלד. רק אחת הסטודנטיות בזוג צריכה להגיש את התרגיל במודל.
- הקפידו לענות על כל מה שנשאלתם.
- תשובות מילוליות והסברים צריכים להיות תמציתיים, קולעים וברורים.
  להנחיה זו מטרה כפולה:
  1. על מנת שנוכל לבדוק את התרגילים שלכם בזמן סביר.
  2. כדי להרגיל אתכם להבעת טיעונים באופן מתומצת ויעיל, ללא פרטים חסרים מצד אחד אך ללא עודף בלתי הכרחי מצד שני. זוהי פרקטיקה חשובה במדעי המחשב.

אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2021

שאלה 1

א. הוכיחו או הפריכו את הטענות הבאות. ציינו תחילה בברור האם הטענה נכונה או לא, ואח"כ הוכיחו / הפריכו באופן פורמלי תוך שימוש בהגדרת $O(\cdot)$.

הנחיה: **יש להוכיח / להפריך כל סעיף בלא יותר מ- 4 שורות.**
הפתרונות הם קצרים, ואינם דורשים מתמטיקה מתוחכמת. אם נקלעתם לתשובה מסורבלת וארוכה, כנראה שאתם לא בכיוון. לאורך השאלה n הוא משתנה ואינו קבוע, כל הפונקציות הן מהטבעיים לעצמם ($f: \mathbb{N} \to \mathbb{N}$) והאופרטור $\log$ הוא לפי בסיס 2.

1. $8^{\log n} = O(n^2)$
2. $n \log n = O(\log(n!))$
3. בהינתן מספר חיובי קבוע k, קבוע חיובי $a_k \in \mathbb{R}^+$ וקבועים $a_0, \ldots, a_{k-1} \in \mathbb{R}$ מתקיים:
$$n^k = O\left(\sum_{i=0}^{k} a_i n^i\right)$$
4. אם $f_1(n) = O\big(g_1(n)\big)$ וגם $f_2(n) = O\big(g_2(n)\big)$ אז $f_1(n) \cdot f_2(n) = O\big(g_1(n) \cdot g_2(n)\big)$
5. אם $f_1(n) = O\big(g_1(n)\big)$ ו $f_2(n) = O\big(g_2(n)\big)$ אז $f_1 \circ f_2(n) = O\big(g_1 \circ g_2(n)\big)$
תזכורת: $f \circ h(n) = f\big(h(n)\big)$
6. סימטריה:
   a. אם $f(n) = O\big(g(n)\big)$ אז $g(n) = O\big(f(n)\big)$
   b. אם $f(n) = O\big(g(n)\big)$ אז $g(n) \neq O\big(f(n)\big)$
7. לכל שני קבועים $0 < \epsilon < 1$ ו-$k \geq 1$ מתקיים: $(\log n)^k = O(n^\epsilon)$ (רמז: השתמשו בהגדרת הגבול שראינו בתרגול).

ב. לכל אחת משתי הפונקציות הבאות, נתחו את סיבוכיות זמן ריצתה במקרה הגרוע כתלות ב- n (אורך הרשימה L).
הניחו כי פעולות אריתמטיות ופעולות append רצות בזמן $O(1)$. ציינו את התשובה הסופית, ונמקו. על הנימוק להיות קולע, קצר וברור, ולהכיל טיעונים מתמטיים או הסברים מילוליים, בהתאם לצורך.
על התשובה להינתן במונחי (…)O, ועל החסם להיות הדוק ככל שניתן. למשל, אם הסיבוכיות של פונקציה היא O(n) ובתשובתכם כתבתם O(nlogn), התשובה לא תקבל ניקוד (על אף שפורמלית O הוא חסם עליון בלבד).

1.

```
def f1(L):
    n = len(L)
    while n > 0:
        n = n // 2
        for i in range(n):
            if i in L:
                L.append(i)
    return L
```

2.

```
def f2(L):
    n = len(L)
    res = []
    for i in range(500, n):
        m = math.floor(math.log(i))
        for j in range(m):
            k=1
            while k<n:
                k*=2
                res.append(k)
    return res
```

ג. להלן פונקציה המייצרת מטריצה ריבועית בגודל $n \times n$:

```
def make_mat(n):
    return [[0] * n] * n
```

להלן פונקציה שמקבלת מטריצה ומבצעת השמה לתא במטריצה:

```
def set(mat, i, j, value):
    mat[i][j] = value
```

אסף הריץ את הקוד הבא:

```
m = make_mat(3)
set(m, 1, 1, 2)
print(m)
```

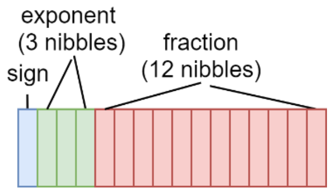
להפתעתו, הקוד הדפיס את הפלט הבא:

```
[[0, 2, 0],
[0, 2, 0],
[0, 2, 0]]
```

מה אסף ציפה שהקוד ידפיס? הסבירו <u>בקצרה</u> מדוע הקוד מתנהג בצורה לא רצויה, כלומר מה גורם לבאג. היעזרו במה שלמדנו על מודל הזיכרון של פייתון. הציעו שינוי לשורה אחת בקוד שתתקן את הבאג.

<u>שאלה 2</u>

ראינו בכיתה את הפונקציות bin_to_float, float_to_bin שממירות בין מספר ממשי למחרוזת המתארת את ייצוגו הבינארי. בשאלה זו נממש פונקציות דומות הממירות בין מספר ממשי למחרוזת המתארת את ייצוגו בבסיס 16 (הקסדצימלי). נייצג מספר ממשי בעזרת 16 ספרות בבסיס 16 (ספרה בבסיס 16 נקראת ניבל nibble). ע״פ התרשים הבא:



והנוסחה לחישוב המספר היא: $num = (-1)^{sign} \cdot 16^{exp-2047} \cdot fraction \cdot 16^{-11}$, כאשר:

- $sign \in \{0,1\}$, כלומר הוא יכול לקבל רק אחד משני ערכים במקום 16 ערכים שונים.
- exp מתאר מספר טבעי המקיים $0 \leq exp \leq 16^3 - 1$.
- fraction בדומה לexp, מתאר מספר טבעי ולא כפי שראיתם בייצוג float בינארי בכיתה. בנוסף, הניבל הראשון משמאל בfraction לא יכול להיות אפס (אלא בין 1 לf). במילים אחרות, $1 \leq fraction \cdot 16^{-11} < 16$.
- לבסוף, המספר אפס מיוצג ע״י 16 אפסים.

לדוגמה, את המספר $-\left(29 + \frac{3}{16}\right) = -29.1875$ נייצג באופן הבא:

- sign=1 כי המספר שלילי.
- החזקה צריכה להיות 1 כדי שfraction יקיים $1 \leq fraction \cdot 16^{-11} < 16$, כלומר $exp - 2047 = 1$ ולכן $exp = (2048)_{10} = (800)_{16}$.
- בfraction נשאר לתאר את המספר $\left(29 + \frac{3}{16}\right) \cdot 16^{-1} = 1 + \frac{13}{16} + \frac{3}{16^2}$, כלומר $fraction \cdot 16^{-11} = (1.d3)_{16}$ ולכן $fraction = (1d3000000000)_{16}$.
- סה״כ נקבל שהייצוג של -29.1875 הוא $'18001d3000000000'$.

א. השלימו בקובץ השלד את הפונקציה hex_to_float(s), אשר מקבלת מחרוזת המתארת ייצוג הקסדצימלי של מספר ממשי ומחזירה את המספר כfloat.
דוגמאות הרצה:

```
>>> hex_to_float('17ff200000000000')
-2.0
>>> hex_to_float('07ff610000000000')
6.0625
```

ב. השלימו בקובץ השלד את הפונקציה float_to_hex(num), אשר מקבלת מספר ממשי (float) ומחזירה מחרוזת המתארת ייצוג הקסדצימלי שלו.
דוגמאות הרצה:

```
>>> float_to_hex(10 * 16**2 + 7 * 16 + 11/16 + 3/16**2)
'0801a70b30000000'
>>> float_to_hex ( - (11/16**2 + 3/16**3 + 12/16**5) )
'17fdb30c00000000'
```

ג. מהו המספר הגדול ביותר הניתן לייצוג בקידוד זה? מהו המספר החיובי הקטן ביותר הניתן לייצוג בקידוד זה? הסבירו.

ד. כמה מספרים שונים ניתן לייצג בקידוד זה? האם קידוד אחר של 16 ניבלים יכול לייצג יותר מספרים? הסבירו.

שאלה 3

נגדיר רשימה משולבת להיות רשימה לא ריקה באורך n זוגי שמכילה מספרים שלמים ובה:

- האיברים באינדקסים ה**זוגיים** מסודרים **מקטן לגדול** ושונים זה מזה, כלומר מהווים רשימה עולה ממש.
- האיברים באינדקסים ה**אי-זוגיים** מסודרים **מגדול לקטן** ושונים זה מזה, כלומר מהווים רשימה יורדת ממש.
- יתכן שערך יופיע בשתי הרשימות.

דוגמאות:

combined_lst1 = [1, 30, 10, 4, 15, 0, 100, -1]

combined_lst2 = [-100, 100, -80, 80]

א. השלימו בקובץ השלד את מימוש הפונקציה search_combined(lst, key), שמקבלת כקלט רשימה משולבת lst וערך key. אם key הינו איבר ברשימה lst, הפונקציה תחזיר את האינדקס i עבורו מתקיים: $lst[i] == key$, אחרת הפונקציה תחזיר None.

על הפונקציה להיות מסיבוכיות זמן $O(\log n)$, כאשר n הינו אורך הרשימה.

דוגמאות הרצה:

```
>>> combined_lst1 = [1,30,10,4,15,0,100,-1]
>>> search_combined(combined_lst1, 10)
2
>>> search_combined(combined_lst1, 0)
5
>>> search_combined(combined_lst1, 200)
None
```

ב. השלימו בקובץ השלד את הפונקציה sort_combined(lst) שמקבלת כקלט רשימה משולבת lst ומחזירה רשימה חדשה באורך זהה שמכילה את איברי lst מסודרים מקטן לגדול. על הפונקציה להיות בסיבוכיות זמן קטנה ככל האפשר, במונחי O(.). מותר להשתמש בפונקציות שראינו בכיתה.
ציינו (בעמוד הבא) מהי סיבוכיות הזמן של האלגוריתם שתיארתם במקרה הגרוע (חסם הדוק ככל האפשר, במונחי O(.)) ונמקו מדוע זה החסם.

ג. השלימו בקובץ השלד את הפונקציה find_duplicate(lst) שבהינתן רשימה משולבת lst לא ריקה (כלומר באורך זוגי לפחות 2) מחזירה אינדקס i זוגי עבורו מתקיים ש- $lst[i] == lst[i + 1]$. אם לא קיים i כזה יוחזר הערך None. על הפונקציה להיות מסיבוכיות זמן $O(\log n)$, כאשר n הינו אורך הרשימה.

דוגמאות הרצה:

```
>>> combined_lstA = [100, 100]
>>>find_duplicate(combined_lstA)
0
>>> combined_lstB = [80, 120, 100, 100]
>>>find_duplicate(combined_lstB)
2
>>>combined_lstC = [100, 250, 200, 210, 210, -300, 400, -400, 500, -500]
>>>find_duplicate(combined_lstC) == None
True
```

שימו לב: בדוגמה האחרונה ישנם שני איברים רצופים זהים אך הראשון מביניהם נמצא באינדקס אי-זוגי.

<u>שאלה 4</u>

בשאלה זו הניחו כי פעולות אריתמטיות והשוואת מספרים מתבצעות בזמן קבוע, וכי הקלט תקין.
כמו כן, על זמן הריצה של המימוש שלכם בכל אחד מהסעיפים להיות נמוך ככל הניתן במונחים אסימפטוטיים.

א. רשימה L היא <u>כמעט ממוינת</u> אם כל איבר בה נמצא לכל היותר במרחק אינדקס אחד מהמיקום שלו ברשימה הממוינת. כלומר, אם $\text{arg\_}sort(i)$ הוא האינדקס של $L[i]$ ברשימה הממוינת sorted(L) אז

$$\text{arg\_}sort(i) \in \{i-1, i, i+1\}$$

לדוגמא, הרשימה [2, 1, 3, 5, 4, 7, 6, 8, 9] היא כמעט ממוינת.

b. השלימו את הפונקציה find בשלד, שמקבלת את רשימה כמעט ממוינת L ומספר שלם s ומחזירה את האינדקס i כך ש $L[i] == s$ אם s הוא איבר ברשימה L, אחרת מחזירה $None$. למשל, עבור L מהדוגמה ו-$s = 5$, הפונקציה תחזיר 3 (כי המספר 5 נמצא באינדקס 3 ברשימה L). עבור $s = 11$ הפונקציה תחזיר `None` (כי המספר 11 לא נמצא ברשימה L).

c. מה היא סיבוכיות זמן הריצה? הסבירו בקצרה.

ב. נרצה למיין רשימה <u>כמעט ממוינת</u> ללא שימוש ברשימת עזר.

a. השלימו את הפונקציה sort_from_almost(lst) בשלד, שמקבלת רשימה כמעט ממוינת וממיינת אותה ללא שימוש ברשימת עזר (או כל מבנה בעל גודל יותר מ O(1)).

b. הסבירו בקצרה את הפתרון שלכם ואת סיבוכיות זמן הריצה שלו.

ג. מינימום מקומי ברשימה L הוא כל אינדקס i שקטן או שווה לשכניו המידיים. כלומר, כל אינדקס המקיים:

$$(i == 0 \ or \ L[i] \leq L[i-1]) \ and \ (i == n-1 \ or \ L[i] \leq L[i+1])$$

לדוגמא, ברשימה [5, 6, 7, 5, 1, 1, 99, 100] האינדקסים 5 ,4 ,0 הן מינימום מקומי.

a. האם בכל רשימה של מספרים יש מינימום מקומי? נמקו את תשובתכם.

b. השלימו את הפונקציה find_local_min בשלד, שמקבלת רשימה של מספרים (לא ממוינים וייתכנו חזרות) ומחזירה אינדקס i של מינימום מקומי (אם יש יותר מאחד אז ניתן לבחור שרירותית). למשל, עבור הרשימה מהדוגמה תשובה של 0, 4 או 5 תהיה תקינה.

c. מהי סיבוכיות זמן הריצה? הסבירו בקצרה.

שאלה 5

בכיתה ראינו את האלגוריתם מיון-בחירה (selection sort) למיון רשימה נתונה. האלגוריתם כזכור רץ בסיבוכיות זמן $O(n^2)$ עבור רשימה בגודל n. בקרוב נראה גם אלגוריתם מיון-מהיר יעיל יותר (quicksort), שרץ בסיבוכיות זמן ממוצעת $O(n \log n)$. לפעמים, כאשר יש לנו מידע נוסף על הקלט, אפשר למיין בסיבוכיות זמן טובה מזו. למשל, בשאלה זו, נעסוק במיון של רשימה שכל איבריה מוגבלים לתחום מצומצם יחסית: מחרוזות באורך k, עבור k>0 נתון כלשהו, מעל האלפבית a, b, c, d, e שמכיל 5 תווים.

ההשוואה בין זוג מחרוזות תהיה לקסיקוגרפית, כלומר השוואה מילונית רגילה.

<u>הערות</u>:

1. בשאלה זו אסור להשתמש בפונקציות מיון מובנות של פייתון.
2. בניתוח הסיבוכיות בשאלה זו נניח שהשוואה של זוג מחרוזות באורך k מבצעת בפועל השוואה של התווים של המחרוזות משמאל לימין, ובמקרה הגרוע תהיה מסיבוכיות זמן $O(k)$.
3. לשם פשטות ניתוח הסיבוכיות נתייחס הן לפעולות אריתמטיות והן לפעולות העתקה של מספרים ממקום למקום בזכרון כפעולות שרצות בזמן קבוע.

א. השלימו בקובץ השלד את הפונקציה string_to_int(s) שמקבלת כקלט מחרוזת s באורך k <u>בדיוק</u> שמורכבת מהתווים a, b, c, d, e ומחזירה מספר שלם בין 0 ל $5^k - 1$ כולל, המייצג את הערך הלקסיקוגרפי היחסי של המחרוזת. על הפונקציה להיות חד-חד-ערכית. סיבוכיות הזמן שלה צריכה להיות $O(k)$.

ב. השלימו בקובץ השלד את הפונקציה int_to_string(k, n), ההפוכה לזו מסעיף א׳, שמקבלת כקלט מספר שלם k גדול מ0, וכן מספר שלם n בין 0 ל $5^k - 1$ כולל ומחזירה מחרוזת s באורך k <u>בדיוק</u> שמורכבת מהתווים a, b, c, d, e. גם על פונקציה זו להיות חד-חד-ערכית. סיבוכיות הזמן שלה צריכה להיות $O(k)$.

שימו לב שפונקציה זו צריכה לקיים לכל $0 \le i \le 5^k - 1$:

string_to_int(int_to_string(k, i)) == i

דוגמת הרצה:

```
>>> for i in range(5**3):
      if string_to_int(int_to_string(3, i)) != i:
        print("Problem with ", i)
>>> alphabet = ["a","b","c","d","e"]
>>> lst = [x+y+z for x in alphabet for y in alphabet for z in alphabet]
>>> for item in lst:
      if int_to_string(3, string_to_int(item)) != item:
        print("Problem with ", item)
>>> #Nothing was printed
```

ג. השלימו בקובץ השלד את הפונקציה sort_strings1(lst, k) שמקבלת כקלט רשימה lst של n מחרוזות כמתואר ומספר חיובי k כך שכל מחרוזת ברשימה הינה באורך k בדיוק. (הניחו כי הקלט תקין ואין צורך לבדוק את תקינותו.) על הפונקציה להחזיר רשימה <u>חדשה</u> ממויינת בסדר עולה (ולא לשנות את lst עצמה). בנוסף לרשימת הפלט שהיא בגודל n כמובן, על הפונקציה להשתמש ברשימת עזר (list) בעלת 5^k איברים.
עליכם להשתמש בפונקציות מסעיפים א', ב'.
על הפונקציה sort_strings1 להיות מסיבוכיות זמן $O(kn + 5^k)$.

ד. בקובץ ה pdf הסבירו מדוע הפונקציה מסעיף ג' עומדת בדרישות הסיבוכיות.

ה. השלימו בקובץ השלד את הפונקציה sort_strings2(lst, k) שמקבלת קלטים כמו הפונקציה מסעיף ג', ובדומה לפונקציה הקודמת עליה להחזיר רשימה חדשה ממויינת בסדר עולה (ולא לשנות את lst עצמה).
הפעם מותר להשתמש בזכרון עזר מגודל $O(k)$, לא כולל רשימת הפלט שעליכם לייצר שגודלה הוא n. על הפונקציה להיות מסיבוכיות זמן $O(5^k \cdot kn)$.

ו. בקובץ ה pdf הסבירו מדוע הפונקציה מסעיף ה' עומדת בדרישות סיבוכיות הזמן והזיכרון.

דוגמת הרצה:

```
>>> import random
>>> k = 4
>>> lst = ["".join([random.choice(["a", "b", "c", "d", "e"]) for i in
range(k)]) for j in range(10)]
>>> lst
['aede', 'adae', 'dded', 'deea', 'cccc', 'aacc', 'edea', 'becb', 'daea',
'ccea']
>>> sort_strings1(lst, k)
['aacc', 'adae', 'aede', 'becb', 'cccc', 'ccea', 'daea', 'dded', 'deea',
'edea']
>>> sort_strings2(lst, k)
['aacc', 'adae', 'aede', 'becb', 'cccc', 'ccea', 'daea', 'dded', 'deea',
'edea']
>>> sorted(lst) == sort_strings1(lst, k)
True
>>> sorted(lst) == sort_strings2(lst, k)
True
```

שאלה 6

כפי שנלמד בהרצאה, קידוד ASCII לטקסט מאפשר לקודד 128 תווים שונים. בשאלה זו, נתעסק בקידודים עבור טקסט שמכיל 36 תווים שונים בלבד – ספרות (0-9) ואותיות קטנות באנגלית (a-z).
אנחנו רוצים ליצור תחליף לקידוד ASCII כך שמחרוזת תתפוס פחות מקום בזיכרון. מיכל הציעה את הקידוד הבא לביטים. להבדיל מ ASCII, הקידוד של מיכל משתמש בקידודים באורכים שונים – 5 ביטים לספרות ו6 ביטים לאותיות – בצורה הבאה:

- כל ספרה (0-9) תקודד באופן הבא: הביט 0 ואחריו ארבעה ביטים שמתארים את ערך הספרה בבינארי. כלומר '0' תקודד ל**0**0000, '1' תקודד ל**0**0001, '9' תקודד ל**0**1001 וכו'.
- כל אות (a-z) תקודד באופן הבא: הביט 1 ואחריו חמישה ביטים שמתארים את האינדקס של האות בבינארי, כאשר האינדקס של a הוא 0, של b הוא 1 וכן הלאה
כלומר 'a' תקודד ל**1**00000, 'b' תקודד ל**1**00001, 'z' תקודד ל**1**11001 וכו'.

א. השלימו בקובץ השלד את הפונקציה code(string) המקבלת מחרוזת מעל הא"ב המתואר לעיל (ספרות ואותיות קטנות בלבד) ומחזירה מחרוזת המתארת את הקידוד הבינארי (כלומר מחרוזת המכילה את התווים '0' ו'1' בלבד) המוצע.
דוגמאות הרצה:

```
>>> code('7b')
>>> '00111100001'
>>> code('cs1001')
>>> '10001011001000001000000000000001'
```

ב. השלימו בקובץ השלד את הפונקציה ההופכית decode(bin_str) המקבלת מחרוזת המתארת הקידוד הבינארי הנ"ל ומחזירה את המחרוזת המקורית.
דוגמאות הרצה:

```
>>> decode('00111100001')
>>> '7b'
>>> decode('100000100010100000100001')
>>> 'acab'
```

ג. נניח כי אנו מייצרים מחרוזת s באורך $len(s) = n$ כך שכל תו במחרוזת נבחר רנדומלית מ- 36 התווים האפשריים (ספרות ואותיות קטנות) בהסתברות שווה. נסמן ב- $code(s)$ את הקידוד של מיכל למחרוזת s וב- $ASCII(s)$ את קידוד ה- ASCII של המחרוזת s. כאשר n גדול מאוד, מהו הערך הצפוי של $\frac{len(code(s))}{len(ASCII(s))}$? הסבירו. האם הקידוד של מיכל חוסך מקום יחסית ל- ASCII? הסבירו.

ד. נעם חשב על הקידוד של מיכל והחליט שזה בזבוז מקום להוסיף את הביט 0 לפני כל ספרה כאשר בטקסטים לרוב יש מספרים ארוכים, כלומר הרבה ספרות ברצף בלי אותיות ביניהן. לכן הציע לשנות את הקידוד כך:

- כל מספר (רצף ספרות) באורך n ספרות יקודד באופן הבא: הביט 0 ואחריו $4 \cdot n$ ביטים כך שכל ארבעה ביטים מתארים ספרה במספר.
- האותיות יקודדו ללא שינוי.

איזו בעיה אתם יכולים למצוא ברעיון של נעם? תנו דוגמה הממחישה את הבעיה והסבירו.

ה. אסף לא אהב את הקידוד של מיכל והציע קידוד אלטרנטיבי למחרוזות מעל הא"ב המתואר:

- כל תו (ספרה או אות) יקודד למספר בינארי שמייצג את האינדקס של התו בא"ב, כאשר הספרות לפני האותיות. כלומר האינדקס של '0' הוא 0, האינדקס של '1' הוא 1, האינדקס של 'a' הוא 10, האינדקס של 'z' הוא 35 וכו'.

כמה ביטים נדרשים בשביל לייצג כל תו בקידוד של אסף, בהנחה שכל התווים מקודדים למספר זהה של ביטים (fixed-length code)? הסבירו. איזה קידוד יותר יעיל בזיכרון, של אסף או של מיכל? הסבירו.

סוף