# תרגיל בית מס' 3 במבוא מורחב למדעי המחשב – 315328963

## שאלה 1

א.

1.**$n \log n = O(n!)$ - הטענה נכונה!**

נוכיח באמצעות הגדרת הגבול:

נבחין כי: $\lim_{n\to\infty} \frac{log(n)}{(n-1)!} = \lim_{n\to\infty} \frac{nlog(n)}{n!}$. מכיוון ש- $n \in \mathbb{N}$, ולפי כללי אלגברה, נוכל לומר כי:

$\frac{log(n)}{n-1} \leq \frac{log(n)}{(n-1)!} \leq 0$ (1). ניעזר בכלל לופיטל, ונראה (באמצעות גבול ידוע) כי: $\lim_{n\to\infty} \frac{\frac{1}{n\cdot\ln(2)}}{1} = \lim_{n\to\infty} \frac{log(n)}{n-1} =$ 0.

נשתמש בכלל הסנדוויץ' (בעזרת משוואה (1)), ואז: $\lim_{n\to\infty} \frac{log(n)}{(n-1)!} = 0 < \infty$, ולכן לפי הגדרה מתקיים:

$n \log n = O(n!)$.

2. בהינתן מספר חיובי קבוע k, קבוע חיובי $a_k \in \mathbb{R}^+$, וקבועים $a_0, \ldots, a_{k-1} \in \mathbb{R}$, מתקיים:

$$n^k = O(\sum_{i=0}^{k} a_i n^i)$$

**הטענה נכונה.**

נוכיח באמצעות הגדרת הגבול:

נבחין כי: $\lim_{n\to\infty} \frac{n^k}{\sum_{i=0}^{k} a_i n^i} = \lim_{n\to\infty} \frac{n^k}{a_0 + a_1 n + \cdots + a_k n^k} = \lim_{n\to\infty} \frac{n^k \cdot 1}{n^k(\frac{a_0}{n^k} + \frac{a_1 n}{n^{k-1}} + \cdots + a_k)} = \lim_{n\to\infty} \frac{1}{\frac{a_0}{n^k} + \frac{a_1 n}{n^{k-1}} + \cdots + a_k} =$

$\frac{1}{a_k} < \infty$ , וזאת מכיוון ש-$a_k \in \mathbb{R}^+$. לכן, לפי הגדרת הגבול, הוכחנו כי

$$n^k = O(\sum_{i=0}^{k} a_i n^i)$$

3. אם $f_1(n) = O(g_1(n))$ וגם $f_2(n) = O(g_2(n))$ אז $f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$.

**הטענה נכונה.**

$f_1(n) = O(g_1(n))$ אז קיים $c_1>0$ כך שלכל n שמקיים: $n>n_1$ מתקיים: $f_1(n) \leq c_1 \cdot g_1(n)$.

$f_2(n) = O(g_2(n))$ אז קיים $c_2>0$ כך שלכל $n$ שמקיים: $n>n_2$ מתקיים: $f_2(n) \leq c_2 \cdot g_2(n)$.

אזי, נכפיל את המשוואות ונקבל: $f_1(n) \cdot f_2(n) \le c_1 \cdot g_1(n) \cdot c_2 \cdot g_2(n) = c_3 \cdot (g_1(n) \cdot g_2(n))$, סימנו $c_3 = c_1 \cdot c_2$, והוכחנו כי אכן $f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$ (לפי הגדרה).

4. סגירות להרכבה: אם f(n)=O(g(n)) וגם h(n)=O(g(n)) אזי $f \circ h(n) = O(g(n))$.

**הטענה איננה נכונה.**

נפריך את הטענה ע"י דוגמה נגדית. תהיינה $f(n) = n^2 - 1, g(n) = n^2, h(n) = n^2 - 2$. קל לראות כי מתקיים $f(n) = O(g(n))$, שכן לכל n מתקיים $n^2 - 1 < n^2$, וכן מתקיים $h(n) = O(g(n))$, שכן לכל n מתקיים $n^2 - 2 < n^2$. עם זאת, נשים לב כי $f \circ h(n) = f(h(n)) = f(n^2 - 2) = (n^2 - 2)^2 - 1 = n^4 - 4n^2 + 3$. נשים לב כי לא קיים c>0 כך שמתקיים עבור כל n החל מ-n מסויים: $n^4 - 4n^2 + 3 \le n^2$, ולכן מתקיים $f \circ h(n) \ne O(g(n))$ והפרכנו את הטענה.

5. טרנזיטיביות: אם $f(n) = O(g(n))$ וגם $g(n) = O(h(n))$ אז $f(n) = O(h(n))$.

**הטענה נכונה.**

$f(n) = O(g(n))$ אז קיים $c_1>0$ כך שלכל n שמקיים: $n>n_1$ מתקיים: $f(n) \le c_1 \cdot g(n)$.

$g(n) = O(h(n))$ אז קיים $c_2>0$ כך שלכל $n$ שמקיים: $n>n_2$ מתקיים: $g(n) \le c_2 \cdot h(n)$.

אזי, מטרנזיטיביות היחס => נוכל להגיד כי $f(n) \le c_1 \cdot g(n) \le c_1 \cdot c_2 \cdot h(n)$. נסמן $c_3 = c_1 \cdot c_2$ ואז נקבל שהחל מ- $n_3 = \max\{n_1, n_2\}$ מתקיים: $f(n) \le c_3 \cdot h(n)$, כלומר $f(n) = O(h(n))$, כנדרש.

6. לכל שני קבועים $0 < \epsilon < 1$ ו-$k \ge 1$ מתקיים: $(\log(n))^k = O(n^\epsilon)$.

**הטענה נכונה.**

נוכיח את הטענה באמצעות הגדרת הגבול. נשים לב שגזירה של הביטוי $\lim_{n\to\infty} \frac{(log(n))^k}{n^\epsilon}$ ע"י כלל לופיטל נותנת: $\lim_{n\to\infty} \frac{(log(n))^k}{n^\epsilon} = \lim_{n\to\infty} \frac{(log(n))^{k-1} * \frac{1}{n*\ln(2)}}{\epsilon n^{\epsilon-1}} = \lim_{n\to\infty} \frac{(log(n))^{k-1}}{\epsilon n^\epsilon * \ln(2)}$. כלומר החזקה של n במכנה אינה משתנה, ואילו החזקה של ביטוי הלוג יורדת באחד. נבצע גזירה זאת j פעמים, עד שהחזקה של הלוג תהיה קטנה מ-0, כלומר k-j<0, ונקבל את הביטוי: $\lim_{n\to\infty} \frac{(log(n))^{k-j}}{\epsilon^j n^\epsilon * \ln^j(2)}$, כלומר כעת $\lim_{n\to\infty} (log(n))^{k-j} = 0$ וגם $\lim_{n\to\infty} \frac{1}{n^\epsilon} = 0$ (כל השאר קבועים), ומחשבון גבולות נקבל בסה"כ (לפי לופיטל) $\lim_{n\to\infty} \frac{(log(n))^k}{n^\epsilon} = \lim_{n\to\infty} \frac{(log(n))^{k-j}}{\epsilon^j n^\epsilon * \ln^j(2)} = 0 < \infty$, והוכחנו לפי הגדרת הגבול.

7. a. לכל n פונקציות $f_1(n), \dots, f_n(n)$ מתקיים:

$f_1(n) + f_2(n) + \cdots + f_n(n) = O(\max\{f_1, \dots, f_n\}(n))$.

**הטענה אינה נכונה.**

נראה זאת באמצעות דוגמה נגדית. תהיינה $f_1(n),\dots,f_n(n)$, פונקציות אשר מקיימות: $f_i = n+i$. נשים לב $\max\{f_1,\dots,f_n\}(n) = 2n$. כמו כן, לפי סכום של סדרה חשבונית, נראה כי $f_1(n)+f_2(n)+\dots+f_n(n) = \frac{n(n+1+n+n)}{2} = \frac{3n^2+1}{2}$. כמו כן, $\frac{3n^2+1}{2} = O(n^2)$ (כי קיים c>0 שמקיים את יחס הקטן שווה), ולכן ניתן לראות כי $f_1(n)+f_2(n)+\dots+f_n(n) = \frac{3n^2+1}{2} = O(n^2) \neq O\big(\max\{f_1,\dots,f_n\}(n)\big) = O(2n)$.
לכן הפרכנו את הטענה.

b. לכל n פונקציות $f_1(n),\dots,f_n(n)$ מתקיים:

$f_1(n)+f_2(n)+\dots+f_n(n) \neq O(\max\{f_1,\dots,f_n\}(n))$.

**הטענה אינה נכונה.**

נראה זאת באמצעות דוגמה נגדית. תהיינה $f_1(n),\dots,f_n(n)$, פונקציות אשר מקיימות: $f_1(n) = n, f_2(n),\dots,f_n(n) = 1$. נשים לב $\max\{f_1,\dots,f_n\}(n) = n$. כמו כן, נראה כי מתקיים $f_1(n)+f_2(n)+\dots+f_n(n) = n+1+1+\dots+1 = n+n-1 = 2n-1$. ניתן לראות כי עבור c=2, מתקיים $2n-1 \leq 2*n$ לכל n, ולפיכך $f_1(n)+f_2(n)+\dots+f_n(n) = 2n-1 = O(n) = O\big(\max\{f_1,\dots,f_n\}(n)\big)$.
לכן הפרכנו את הטענה.

ב. 1. סיבוכיות הפונקציה היא $O(n^2)$. נשים לב כי ישנה לולאה אחת שרצה על n מספרים, ובמקרה הגרוע ביותר כל המספרים בטווח של 0 עד n נמצאים ב-L. הסיבוכיות של פעולת הבדיקה in היא O(n), כלומר לכל מס' בטווח, נעשית פעולה בסיבוכיות זאת. לכן הסיבוכיות של הפונקציה עד שלב זה היא $n \cdot n = n^2$, כלומר $O(n^2)$. הסיבוכיות של פעולת append היא O(1) כפי שנאמר, לכן הסיבוכיות הכללית של הפונקציה היא $O(n^2)$.

2. סיבוכיות הפונקציה היא $O(n^2 \log(n))$. ראשית נתייחס ללולאה הראשונה והשנייה שהיא מקוננת בה. בראשונה יש ריצה על n-500 אובייקטים, ובשנייה יש ריצה על 2i אובייקטים (כאשר i הוא אובייקט בין 500 ל-n, והריצה היא בהתאם לגודל של i). כלומר כפי שלמדנו בתרגול, נשתמש בסכום, שבמקרה זה מתנהג כמו סכום סדרה חשבונית: $\sum_{i=500}^{n} 2i$, וכפי שלמדנו הסיבוכיות של זה היא $O(n^2)$. נשים לב שהלולאה הפנימית הנוספת אינה תלויה בגודל של האחרות אלא מציגה משתנה חדש k שמתחיל בערך של 1 (או $2^0$) וגדל בטור הנדסי עם מנה 2, עד שמגיע לגודל של n. כלומר, נשאל את עצמנו באיזו חזקה j יתקיים $2^j > n$. נזהה כי הדבר יקרה כאשר $j > \log(n)$, ולכן הסיבוכיות של לולאה זאת היא O(log(n)). מכיוון שהלולאות אינן תלויות, נכפיל את הסיבוכיות שלהן אחת בשנייה (כפי שלמדנו בתרגול) ונקבל: $O(n^2) * O(\log(n)) = O(n^2 \log(n))$.

ג. הדבר נובע מתכונות הפעולות השונות שבמרכז הפונקציות, והשפעתן על מודל הזיכרון של פייתון. הפונקציה הראשונה מוסיפה את האיברים ל-l באמצעות אופרטור +. דרך זאת יוצרת, בכל איטרציה,

רשימה חדשה בשם l, הכוללת את l ואת האיבר החדש. עם זאת, רשימה זאת נוצרת במקום אחר בזיכרון מאשר רשימת l המקורית. לכן, לולאת ה-for נעצרת לאחר שעוברת מספר איטרציות שהוא כגודל רשימת l המקורית (קרי, 3 איטרציות). לעומת זאת, הפונקציה השנייה, בעזרת האופרטור =+ מוסיפה את האיברים ל-l המקורית (ולא יוצרת רשימה חדשה במקום אחר בזיכרון). לכן, עם כל איטרציה הגודל של l מתעדכן, ומכיוון שהוא גם גדל בכל איטרציה (בשל פעולת הפונקציה השנייה), לולאת ה-for תרוץ על יותר יותר איברים. כך שנוצר מצב בו מס' האיברים עליה רצה הלולאה גדל מאיטרציה לאיטרציה, מה שיוצר לולאה אינסופית.

## **<u>שאלה 2</u>**

ט. להלן הטבלה הראשונה של התוצאות:

| n | 23 | 53 | 101 | 199 | 401 |
|---|---|---|---|---|---|
| complete_graph | 96.9 | 242.8 | 436.3 | 1233.3 | 2545.8 |
| cycle | 262 | 1026.7 | 6418.7 | 21364.3 | 84922.9 |
| inv_cycle | 132.9 | 469 | 1059.1 | 2572.4 | 6338.3 |

נתייחס לכל פונקציה בנפרד:

- **Complete_graph**: ניתן לראות מתוצאות הגרף שהתוצאות בטבלה גדלו בצורה כמעט לינארית (כלומר פי 2), בדומה לקפיצה בערכי n (גודל הגרף. זאת, אולי מלבד בין הערכים 101 ו-199. לכן, נסיק כי הפונקציה מתנהגת כמו O(n). ניתן להסביר זאת בכך שהפונקציה דורשת בערך 2n פעולות: n פעולות של יצירת שורות המטריצה (שהינן זהות), ועוד *n* פעולות של העתקת שורות אלה למטריצה (סידורן במטריצה). קל לראות כי ניתן לחסום מספר זה ב-O(n). כמו כן, בגרף זה יותר "קל" להשלים את ההילוך, כי לכל צומת כל שאר הצמתים הם שכניו ולכן יש *n* אפשרויות לכל צעד.
- **Cycle**: ניתן לראות מתוצאות הגרף שהתוצאות בטבלה גדלו ריבועית. כלומר, בעוד שערכי גודל הגרף (n) גדלים פי 2 בערך, ערכי הפונקציה גדלים פי 4 בערך, כלומר $2^2$. לכן, נסיק כי הפונקציה מתנהגת כמו $O(n^2)$. ננסה להסביר זאת בכך שהפונקציה כוללת לולאה שרצה על *n-2* ערכים, ובמסגרתה במהלך כל איטרציה נעשית יצירה של רשימת אפסים באורך *n*. כמו כן, הגרף שנוצר כאן לא מכיל הרבה שכנים, מכאן שאפשר להניח כי ייקחו לו יותר צעדים בכדי להשלים את ההילוך. מכאן, שהדבר מעניק לפונקציה חסם עליון של $O(n^2)$.
- **Inv_cycle**: ניתן לראות מתוצאות הגרף שתוצאות הטבלה גדלו בצורה כמעט לינארית (למרות שאין אחידות). בעוד ערכי גודל הגרף גדלו פי 2, ניתן לראות כי ערכי הפונקציה גדלו פי 2 בערך. לכן נאמר כי הפונקציה מתנהגת כמו O(n). הדבר יכול לנבוע מכך שלעומת *cycle*, כאן לכל צומת בגרף שנוצר יש שכן אחד יותר ויש פחות שכנים מרוחקים. לכן, מס' הצעדים להשלמת ההילך צפוי לרדת לעומת הפונקציה הקודמת, ומכאן שהפונקציה מתנהגת כמו O(n).

להלן הטבלה השנייה:

| | 125 | 250 | 500 | 1000 | 2000 |
|---|---|---|---|---|---|
| 0.3 | 758.6 | 1477.6 | 3533.9 | 7661.5 | 15746.5 |
| 0.5 | 671.5 | 1404.1 | 3615 | 7187.5 | 15924 |
| 0.7 | 627.1 | 1342.9 | 3321.3 | 7523.9 | 15522 |

נראה כי זמן הכיסוי מתנהג באופן לינארי, כלומר $O(n)$. נשים לב כי ככל שגודל הגרף עולה, במקרה הזה פי 2, מספר הצעדים עולה בערך פי 2 גם כן. לכן הדבר מצביע על התנהגות לינארית של זמן הכיסוי.

ניתן לראות השפעה מינורית של הסיכוי לקשת. ככל שהסיכוי לקשת עולה, ישנם מעט פחות צעדים הדרושים עד להשלמת ההילוך. הדבר מסתדר בכך שהסיכוי לגרף בעל מספר צמתים גבוה יותר, הוא גדול יותר. באופן אינטואיטיבי, ככל שגרף הוא בעל מספר קשתות גבוה יותר, ישנן יותר אפשרויות לכל צומת "לצעוד" אליהן, מה שמגדיל את הסיכוי להשלמה מהירה יותר של ההילך.

להלן הנתונים שנמדדו לגבי return_graph:

| | | | | | |
|---|---|---|---|---|---|
| n | 2 | 4 | 8 | 16 | 24 |
| return_graph | 2 | 10 | 229.7 | 47592.4 | 5063687 |

נשים לב כי מדובר בגידול מעריכי (אקספוננציאלי), כך שזמן הכיסוי פועל בצורה של $O(2^n)$. נסביר זאת בכך שהפונקציה פועלת על צמתים בצורה שבה 'יש להחליט' (לכאורה) לכל צומת האם הוא מחובר בקשת לצומת הבא או שלא. הסיכוי שזה אכן יקרה הוא 0.5. עבור *n* צמתים, הסיכוי הוא בערך (0.5^n), לכן ההסתברות היא *2^n*. מכאן שזמן הכיסוי פועל בצורה של $O(2^n)$ (מעריכית). לפי ההערכה והניסוי, ה-*n* המקסימלי עבורו זמן הריצה הוא פחות מדקה הוא 24, כאשר היו גם הרצות שלו שלקחו יותר מדקה.

## <u>שאלה 3</u>

ב. סיבוכיות זמן הריצה של הפונקציה היא $O(nk)$. נשים לב כי הפונקציה פועלת בסה"כ על $\left\lceil \frac{n}{k} \right\rceil$ רשימות באורך k. עבור כל רשימה, מופעלת הפונקציה selection_sort. אנו רואים כי בפונקציה הזאת ישנן לולאות מקוננות תלויות (כאשר הראשונה רצה על k ערכים). לפי מה שלמדנו בתרגול, ומכיוון שאין פעולות שגוררות סיבוכיות גבוהה יותר (כן יש פעולות אריתמטיות של השוואה ושינוי משתנים), הסיבוכיות של selection_sort תהיה $O(k^2)$. נזכיר כי הפונקציה generate_sorted_blocks מפעילה את selection_sort על $\left\lceil \frac{n}{k} \right\rceil$ רשימות באורך *k*, ולכן נכפיל את הערכים ונקבל בסה"כ סיבוכיות של $O(nk)$.

ד. נשים לב כי ראשית, הפונקציה מפעילה את generate_sorted_blocks, שהסיבוכיות שלה היא O(nk) (סעיף ב'). לאחר מכן הפונקציה מפעילה את merge_sorted_blocks, שלפי הדרישה היא בסיבוכיות של $O(m \cdot t \cdot \log(m))$. *נבחין כי* $m = \left\lceil \frac{n}{k} \right\rceil$ (מספר הרשימות הקטנות), וכי t=k (אורך כל רשימה קטנה), לכן נציב ונקבל כעת את הסיבוכיות: $O(nlog\left(\frac{n}{k}\right))$. מכיוון שהפונקציות פועלות אחת אחרי השנייה והן אינן תלויות ממש אחת בשנייה, נחבר את הסיבוכיות של שתיהן ונקבל $O(nk + nlog\left(\frac{n}{k}\right))$.

ה. נשים לב כי אם נציב $k=O(n^c)$, כאשר c הוא מספר חיובי, אנחנו נגדיל את הביטוי השמאלי בסיבוכיות אליה הגענו, ונגיע לסיבוכיות גבוהה מאשר O(nlogn). לכן, הערך האסימפטוטי הגבוה ביותר של k צריך להיות **O(logn)**. במקרה זה הסיבוכיות תהיה $O\left(nk + nlog\left(\frac{n}{k}\right)\right) = O\left(nlogn + nlogn\left(\frac{n}{logn}\right)\right) =$ $O(nlogn)$.

## <u>שאלה 4</u>

א. *B*. סיבוכיות זמן הריצה היא O(log(n)). בכתיבת הקוד נעשה שימוש בשיטה של 'חיפוש בינארי'. כלומר, מדי איטרציה (וכל עוד האינדקסים לא צמודים), מצמצמים את החיפוש על אורך הרשימה חלקי 2. כל איטרציה לוקחים את האיבר האמצעי. אם הוא שווה לאינדקס בו הוא נמצא, אנו במצב 'תקין' (אין פער/הפרש), ולכן נחפש בצד הימני של הרשימה (בו מסתמן 'פער' כזה). אם הוא גדול מהאינדקס בו הוא נמצא, נחפש בצד השמאלי של הרשימה (נחפש אחר התאמה בין אינדקס לאיבר עצמו, כלומר שהם שווים). בסה"כ נשאל את עצמנו עבור איזה *k* יתקיים: $\frac{n}{2^k} = 1$, והתשובה היא O(log(n)).

ב. D. סיבוכיות זמן הריצה היא *O(log(n))*. ראשית, אנו עושים חיפוש על מנת למצוא את האינדקס בו הרשימה "נופלת" מבחינת ערך המספרים. חיפוש זה נעשה בשיטת 'חיפוש בינארי', שכפי שהוסבר קודם לכן, היא בסיבוכיות *O(log(n))*. לאחר מכן נעשה חיפושים בשיטה דומה (של 'חיפוש בינארי'), על כל הרשימה, שנעשה על הרשימה בצורה דומה לאם הרשימה היתה מסודרת כרגיל. ע"י משחק עם האינדקסים אנו מתייחסים לרשימה כאל ממוינת בסדר עולה. כל חיפוש כזה הוא גם בסיבוכיות *O(log(n))*. לכן עלינו לחבר בין הסיבוכיות של השניים, ולכן הסיבוכיות של זמן הריצה הכללי היא גם *O(log(n))*.

ג. סיבוכיות זמן הריצה במקרה הגרוע ביותר היא *O(n)*. לעומת הסעיפים הקודמים, בהם הסיבוכיות נמוכה יותר, בסעיף זה, תתכן אפשרות בה לא ניתן למצוא את 'אינדקס הסיבוב', בו הרשימה "יורדת" מבחינת ערך מספרי. זאת במקרים של רשימה ארוכה המורכבת ממספר מסוים *i*, מלבד מיקום אחד בו נמצא מספר אחר *j* (מצביע על אינדקס זה). במקרה כזה, לא ניתן לדעת לאן "לפנות" בעת חיפוש בינארי, כי יש שוויון בין כמעט כל המספרים. לכן, במקרה בו מצאנו את האינדקס הזה, נפנה לפתרון של סעיף ב', שיעזור למצוא האם המספר נמצא ברשימה (וייתן לו סיבוכיות של *O(logn)*). במקרה ולא, נעבור "ידנית" על איברי הרשימה, אחד אחד, מה שייתן סיבוכיות של *O(n)*.
לכן, סיבוכיות זמן הריצה במקרה הגרוע ביותר היא *O(n)*.

## **שאלה 5**

ד. ראשית, הפונקציה יוצרת רשימת אפסים באורך $5^k$, שזאת פעולה בסיבוכיות $O(5^k)$. אחר כך, הפונקציה עוברת על הרשימה (באורך $n$) ומבצעת את הפונקציה *string_to_int*. כלומר, הסיבוכיות של פעולות אלה היא $O(nk)$. לאחר שהפונקציה הוסיפה מספרים לרשימת האפסים (בהתאם לנוכחות של כל מספר/מחרוזת ברשימת הקלט), אנו עוברים על רשימת העזר ומבצעים פעולה **רק** על האינדקסים ששונים מאפס, בסה"כ $n$ כאלה. הפעולה שאנו עושים היא הוספה שלהם לרשימת הפלט, כלומר $O(n)$ בסה"כ. לבסוף אנו הופכים כל מספר ברשימת הקלט ($n$) כאלה, למחרוזת המתאימה לו, בעזרת שימוש בפונקציה *int_to_string*, בסיבוכיות $O(k)$. כלומר בסה"כ $O(nk)$. נחבר את סך כל הפעולות האלה, ונקבל $O(nk + 5^k)$, כנדרש.

ה. נשים לב כי בפעולת הפונקציה לולאות מקוננות. הראשונה רצה על $5^k$ ערכים, והשנייה רצה על רשימת הקלט, כלומר על $n$ ערכים. מכיוון שהלולאה הראשונה רצה על ערכים שונים, נשים לב כי הפעולות שתבוצענה, ייעשו רק על $n$ ערכים בסך הכל. הפעולה שתבוצע היא *string_to_int*, ולכן בסך הכל בחלק זה הסיבוכיות היא $O(5^k*nk)$ (כי נדרש להכפיל את כל הערכים האלה). נשים לב כי ישנה עוד לולאה שרצה על $n$ ערכים ועל כל אחד מבצעת פעולה בסיבוכיות של $O(nk)$ (*int_to_string*). נחבר את שתי הסיבוכיות ונקבל כי בסה"כ הסיבוכיות היא כנדרש $O(5^k*nk)$.

## **שאלת בונוס**

מיכל משתמשת במעין שיטה של חיפוש בינארי. בחפיסה 8 קלפים סה"כ. ראשית, מיכל מחלקת אותם לשתי קבוצות באופן שרירותי, ואמיר אומר לה באיזו מהן נמצא המספר שלו (נסמן אותה בקבוצה א'). כעת מיכל אוספת את כל הקלפים, את קבוצה א' משאירה למטה, וכך היא מחלקת אותה לשתי תתי קבוצות (למשל ב' וג'). שאר הקלפים של ב' וג' הם איברים שמיכל יודעת **בוודאות** שאינם הקלף של אמיר (שכן הם לא מקבוצה א').

כעת אמיר בוחר את הקבוצה בה הקלף שלו מופיע (נגיד שזאת קבוצה ב'). מיכל לוקחת את הקלפים, זוכרת איפה ממוקמים 2 הקלפים 'החשודים' שנותרו לה, ואותם מפצלת לשתי קבוצות שונות (למשל ד' וה'). אמיר בוחר את הקבוצה בה נמצא המספר (קבוצה ד' למשל), ואז מיכל, שזוכרת את המיקום של הקלף החשוד (שהוא הקלף השלישי שהונח ברשימה, או השביעי בחפיסה המאוחדת), יודעת כעת שקלף זה הוא הקלף שאמיר בחר במקור. כך שבאמצעות מעין חיפוש בינארי, מיכל הצליחה לנחש את הקלף של אמיר.