פתרון לממ"ן 13 בקורס 20407 2019א

שאלה 1

סעיף א

אם S הוא מערך ממוין, כל שנדרש הוא להציב מצביע greater בסוף המערך ומצביע smaller בתחילת המערך. אם סכום שני הערכים גדול מ-z, נקטין את הערך של greater (וכך יקטן גם הסכום) ע"י קידום שלו פנימה אל תוך המערך. אם סכום שני הערכים קטן מ-z נגדים את הערך של smaller (וכך יגדל גם הסכום) ע"י קידום שלו פנימה אל תוך המערך. נסיים כאשר smaller ו-greater ייפגשו או לפני כן, כאשר S[smaller] + S[greater] = z.

שלב זה באלגוריתם דורש מעבר לינארי על איברי המערך (על כולם, במקרה הגרוע) ולכן סיבוכיות זמן הריצה שלו היא $\Theta(n)$.

אך ההנחה בשלב זה היא שהמערך ממוין, לכן על מנת לשמור על סיבוכיות זמן הריצה הלינארית, ניאלץ להשתמש במיון בזמן לינארי. מכיוון שהנחת הקלט היחידה שעומדת לרשותי היא שכל איברי הקבוצה S שייכים לתחום $[0, \lceil n\sqrt{n} \rceil]$ המכיל $\lceil n\sqrt{n} \rceil$ איברים, אשתמש במיון בסיס.

נחשב כמה ספרות נחוצות על מנת לייצג כל מספר בתחום זה בבסיס n לפי הנוסחה בעמוד 118 במדריך הלמידה:

$$d = \lfloor \log_n \lceil n\sqrt{n} \rceil \rfloor \leq \lfloor \log_n n^2 \rfloor = \lfloor 2 \rfloor = 2$$

לכן, הייצוג של כל מספר בטווח שנתון לנו ידרוש ייצוג של 2 ספרות בבסיס n לכל היותר.

לפי עמוד 143 בספר הלימוד סיבוכיות זמן הריצה של מיון בסיס של n איברים כאשר הבסיס מונה n ספרות וכל מספר בטווח דורש ייצוג של 2 ספרות (ע"פ הסימונים בספר k=n ו- d=2) הוא $\Theta(d+k) =$ $\Theta(2+n) = \Theta(n)$.

<u>IS_THERE_TWO_NUMBERS_WITH_SUM_Z</u>(S,z):

```
RADIX_SORT(S, 2) // from page 143 in the book
larger ← length[S];
smaller ← 1;
while larger ≠ smaller:
    sum ← S[larger] + S[smaller];
    If sum = z then:
        return S[larger] ≠ S[smaller];
    else if sum < z then:
        smaller ← smaller + 1;
    else:
        larger ← larger – 1;
Return FALSE;
```

בשורה 1 אנו מבצעים את מיון הבסיס לפי ההסבר שניתן בתחילת הסעיף. מיון זה מתבצע בסיבוכיות ריצה של $\Theta(n)$.

בשורות 2-3 אנחנו מציבים מצביע אחד על תחילת המערך (האיבר הקטן ביותר במערך) ומצביע שני על סוף המערך (האיבר הגדול ביותר במערך). אלו פעולות בזמן קבוע ולכן סיבוכיות הריצה היא $\Theta(1)$.

בלולאה שבשורות 4-11 אנחנו עוברים עם שני המצביעים פנימה לתוך המערך כל עוד שני המצביעים לא נפגשו. ראשית, בשורה 5, שומרים את סכום שני האיברים שאנו מצביעים עליהם.

אם סכום זה שווה ל-z (שורות 6-7) נחזיר TRUE אם המספרים שונים (אלה שני מספרים שונים זה מזה שסכומם הוא z, כפי שהתבקשנו). אם המספרים שווים, הרי שכל המספרים ביניהם שווים מכיוון שהמערך ממוין ואין סיבה להמשיך ולעבור על יתר האיברים, לכן נחזיר FALSE.

אחרת, אם הסכום קטן מ-z (שורות 8-9), נקדם את smaller לאיבר הבא שהוא גדול או שווה לו, כך שסכום שנחשב באיטרציה הבאה יהיה גדול או שווה לסכום מהאיטרציה הנוכחית.

אחרת, (שורות 10-11, כלומר, כאשר הסכום גדול מ-z) בדומה, נקדם את larger פנימה אל תוך המערך (כלומר, צעד אחורה) לאיבר קטן או שווה לו, מה שיהפוך את הסכום באיטרציה הבאה לקטן או שווה לסכום מהאיטרציה הנוכחית.

בשורה 12, אם הגיע תנאי העצירה ו- smaller ו-larger "נפגשו", לא מצאנו זוג ערכים שונים במערך שסכומם Z ולכן נחזיר FALSE. זמן ריצה קבוע ולכן סיבוכיות זמן הריצה היא $\Theta(1)$.

הלולאה בשורות 4-11 עוברת, במקרה הגרוע, על כל האיברים במערך אך מכיוון שבכל איטרציה מתקדם אחד המצביעים לכיוון האחר עד שהם נפגשים, לעולם לא נעבור על איבר מסויים מבלי להתקדם (בכל איטרציה, אם לא יצאנו מהלולאה, אחד המצביעים ייתקדם). לכן זוהי סיבוכיות זמן ריצה לינארית $\Theta(n)$.

לסיכום, סיבוכיות זמן הריצה של אלגוריתם זה הוא לינארי.

בסדר

סעיף ב

נשנה מעט את השגרה מסעיף א' ע"י החלפת שורה 7 בשורות:

1. If S[larger] ≠ S[smaller] then:
2. return array [S[larger], S[smaller]];
3. else return NIL;

ובמקום להחזיר FALSE בשורה 12, נחזיר NIL. נסמן שגרה זו בשם FIND_TWO_NUMBERS_WITH_SUM_Z. השינוי אינו פוגע בסיבוכיות זמן הריצה מכיוון החלפנו פעולות בזמן קבוע בפעולות אחרות שלוקחות זמן קבוע.

כעת, FIND_TWO_NUMBERS_WITH_SUM_Z מחזירה מערך בן שני איברים המכיל שני מספרים שונים שסכומם הוא z ואם אין כאלה, מחזירה NIL.

אפשר להעביר את מיון S לשגרה הזו ואז לחסוך את המיון בכל איטרציה, זה יחסוך בזמן הריצה, אך אסימפטומטית אין השפעה.

נשתמש בשגרה זו על מנת לחפש עבור כל איבר i ב-S זוג מספרים שונים ב-S שסכומם הוא z פחות S[i]. אם מצאנו שני מספרים כאלה, נבדוק האם הם שונים מ-S[i]. אם כן, מצאנו שלשה מספרים שונים שסכומם z ולכן נחזיר TRUE. אם עברנו על כל איברי S ולא מצאנו איברים כאלה, נחזיר FALSE.

<u>IS_THERE_THREE_NUMBERS_WITH_SUM_Z</u>(S, z):

1. for i ← from 1 to *length*[S] do:
2. twoNumbersReasult ← FIND_TWO_NUMBERS_WITH_SUM_Z(S, z - S[i]);

```
3.        if twoNumbersReasult ≠ NIL
            and S[i] ≠ twoNumbersReasult[1]
            and S[i] ≠ twoNumbersReasult[2] then:
4.            return TRUE;
5.  Return FALSE;
```

בלולאה שבשורה 1-4 נעבור בצורה לינארית על המערך S. עבור כל איבר i במערך S נריץ את השגרה FIND_TWO_NUMBERS_WITH_SUM_Z על המערך S ונחפש שני איברים שונים שסכומם z פחות ערך האיבר i (שורות 1-2).

סיבוכיות הריצה של השיטה שנקראת בשורה השנייה היא $\Theta(n)$.

אם אנחנו מוצאים שני מספרים באלה a ו-b (שורות 3-4), נבדוק שהם שונים מערך האיבר הנוכחי. אם שני המספרים שונים מערך האיבר הנוכחי, אז שלוש המספרים שונים זה מזה ומתקיים a + b + S[i] = z שכן a + b = z – S[i] (כי כך בחרנו אותם). בדיקה זו לוקחת זמן קבוע ולכן סיבוכיות הריצה שלה היא $\Theta(1)$.

במקרה הגרוע, ניאלץ לעבור על n האיברים בקבוצה S ולכן ללולאה יהיו n איטרציות.

אחרת, בשורה 5, אם לכל איבר i ב-S לא מצאנו שני איברים שונים שסכומם z פחות ערך האיבר i, נחזיר FALSE. פקודה לוקחת זמן קבוע ולכן סיבוכיות הריצה שלה היא $\Theta(1)$.

לסיכום, סיבוכיות זמן הריצה של האלגוריתם היא $\Theta(n*(n+1)+1) = \Theta(n^2+n+1) = \Theta(n^2)$ בנדרש

סעיף ג

נשתמש בשיטה דומה לפתרון בסעיף א', גם במקרה הזה אפשר לוותר על המיון בשגרה שתיקנו בסעיף ב' מכיוון שהמערך כבר יהיה ממוין, הדבר יקל על זמן הריצה אך אין לו השפעה אסימפטוטית.

עבור כל זוג מספרים, נריץ את השגרה FIND_TWO_NUMBERS_WITH_SUM_Z, ונחפש ביתר המערך עוד שני איברים שסכום ארבעתם הוא z. אם נמצא כאלה, נבדוק שערכם שונה. אם כן, נחזיר TRUE, אם הסכום של ארבעת המספרים קטן מדי, נגדיל אותו באיטרציה הבאה ע"י קידום smaller לאיבר הבא הגדול יותר או שווה, אם הסכום של ארבעת המספרים גדול מדי, נקטין אותו באיטרציה הבאה ע"י קידום larger פנימה אל המערך, לאיבר הקודם הקטן יותר או שווה.

אם לא מצאנו ו-smaller ו-larger נפגשו או שהם מכילים ערכים שווים, נחזיר FALSE.

<u>IS_THERE_FOUR_NUMBERS_WITH_SUM_Z</u>(S, z):

```
RADIX_SORT(S, 2) // from page 143 in the book
larger ← length[S];
smaller ← 1;
while larger ≠ smaller:
    currentSum ← S[larger] + S[smaller];
    twoNumbersSum ← FIND_TWO_NUMBERS_WITH_SUM_Z(S[smaller…larger], z - sum);
    If twoNumbersSum ≠ NIL:
        totalSum ← twoNumbersSum[1] + twoNumbersSum[2] + currentSum
        If totalSum = z then:
            If S[larger] ≠ S[smaller] then:
                if S[larger] ≠ twoNumbersSum[1]
                  and S[larger] ≠ twoNumbersSum[2]
```

```
                    and S[larger] ≠ twoNumbersSum[1]
                    and S[larger] ≠ twoNumbersSum[2] then:
12.                    return TRUE;
13.                 Else:
14.                     smaller ← smaller + 1;
15.             Else:
16.                 Return FALSE
17.         else if totalSum < z then:
18.             smaller ← smaller + 1;
19.         else:
20.             larger ← larger – 1;
21.     else:
22.         smaller ← smaller + 1;
23. Return FALSE;
```

בשורה 1 אנו מבצעים את מיון הבסיס לפי ההסבר שניתן בתחילת סעיף א'. מיון זה מתבצע בסיבוכיות ריצה של $\Theta(n)$.

בשורות 2-3 אנחנו מציבים מצביע אחד על תחילת המערך (האיבר הקטן ביותר במערך) ומצביע שני על סוף המערך (האיבר הגדול ביותר במערך). אלו פעולות בזמן קבוע ולכן סיבוכיות הריצה היא $\Theta(1)$.

בלולאה שבשורות 4-14 אנחנו עוברים עם שני המצביעים פנימה לתוך המערך כל עוד שני המצביעים לא נפגשו. ראשית, בשורה 5, שומרים את סכום שני האיברים שאנו מצביעים עליהם.

בשורה 6 נקרא לשגרה FIND_TWO_NUMBERS_WITH_SUM_Z ונחפש בתת המערך שבין שני המצביעים שני איברים שונים אחד מהשני שסכומם הוא z פחות סכום שני האיברים שאנחנו מצביעים עליהם. סיבוכיות זמן הריצה של שורה זו היא $\Theta(n-(larger-smaller))=\Theta(n)$

אם מצאנו שני מספרים המקיימים את הדרישה אזי נחשב את סכום ארבעת המספרים (שורות 7-8) בזמן ריצה קבוע.

- אם סכום זה שווה ל-z (שורה 9):
  - אם שני המספרים שאנחנו מצביעים עליהם שונים זה מזה (שורה 10):
    - נבדוק האם כל ארבעת המספרים שונים זה מזה (שורות 11-12) נחזיר TRUE. אם יש זוג מספרים שווים מבין הארבעה, נמשיך לחפש.
    - אחרת, נמשיך לחפש ע"י קידום smaller. (שורות 13-14).
  - אחרת נחזיר FALSE. אין למה להמשיך לחפש כי מכיוון שהמערך ממוין, כל המספרים ביניהם שווים. (שורות 15-16)
- אחרת, אם הסכום קטן מ-z (שורות 17-18), נקדם את smaller לאיבר הבא שהוא גדול או שווה לו, כך שסכום שנחשב באיטרציה הבאה יהיה גדול או שווה לסכום מהאיטרציה הנוכחית.
- אחרת, (שורות 19-20), כלומר, כאשר הסכום גדול מ-z) בדומה, נקדם את larger פנימה אל תוך המערך (כלומר, צעד אחורה) לאיבר קטן או שווה לו, מה שיהפוך את הסכום באיטרציה הבאה לקטן או שווה לסכום מהאיטרציה הנוכחית.

אחרת, נקדם את smaller וננסה שוב. (שורות 21-22)

כל הפעולות בשורות 7-22 רצות בזמן קבוע ולכן סיבוכיות זמן הריצה היא $\Theta(1)$.

בדומה לסעיף א', ללולאה לכל היותר n איטרציות ולכן סיבוכיות זמן הריצה של הלולאה בשורות 4-22 היא $\Theta(n*n)=\Theta(n^2)$.

בשורה 23, אם הגיע תנאי העצירה ו- smaller ו-larger "נפגשו", לא מצאנו רביעיית ערכים שונים במערך שסכומם Z ולכן נחזיר FALSE. זמן ריצה קבוע ולכן סיבוכיות זמן הריצה היא $\Theta(1)$.

לסיכום, סיבוכיות זמן הריצה של אלגוריתם זה הוא $\Theta(n^2)$ כנדרש.

שאלה 2

נתון כי התפלגות הנקודות אחידה, לכן "נכין את השטח" על מנת להשתמש במיון דלי.

נרצה לחלק את הנקודות ל-n דליים לפי θ של כל נקודה.

הזווית בחצי הימני של העיגול היא π. ע"י קרניים שמקורן בראשית הצירים נחלק את הזווית ל-n חלקים שווים, הזווית של כל חלק בגודל $\frac{\pi}{n}$. ומכיוון שזהו החצי הימני של העיגול, הקרן הראשונה היא בסיבוב של $90°$ ימינה, כלומר $-\frac{\pi}{2}$, הזווית של כל חלק j כך ש-$0 \le j \le n$, תהיה מוגדרת ע"י $\alpha_j = j * \frac{\pi}{n} - \frac{\pi}{2}$.

לכל j, כך ש-$0 \le j \le n$ מתקיים $-\frac{\pi}{2} \le \alpha_j \le \frac{\pi}{2}$ לכן $0 \le \frac{\alpha_j + \frac{\pi}{2}}{\pi} = \frac{\alpha_j}{\pi} + \frac{1}{2} \le 1$. מצאנו את הנוסחה שיש לחשב על מנת למצוא עבור כל נקודה את המספר בקטע [0,1] המתאים לה.

נסמן n דליים, כך שכל דלי $1 \le b_j \le n$ יכיל את הנקודות שה θ שלהן נמצאות בין α_{j-1} ו-α_j.

ומכיוון שמתקיים $\tan(\theta_i) = \frac{x_i}{y_i}$ אזי

$$\theta_i = \arctan(\tan(\theta_i)) = \arctan(\frac{x_i}{y_i})$$

כלומר, כל נקודה $p_i = (x_i, y_i)$, כך ש $1 \le i \le n$, שייכת לדלי b_j אמ"מ $\alpha_{j-1} \le \theta_i \le \alpha_j$.

לכן, נשתמש במיון דלי כפי שמתואר באלגוריתם בעמוד 145 בספר, את שורה 3 המבצעת את החלוקה של האיברים לדליים המתאימים נחליף בשורה:

do insert A[i] into list B[n $* \left(\frac{\arctan(\frac{x_i}{y_i})}{\pi} + \frac{1}{2}\right)$];

ומכיוון שלא שינינו את סיבוכיות הריצה ע"י החלפת שורה זו (זו פעולה שלוקחת זמן קבוע), סיבוכיות הריצה של מיון הדלי לאחר ההחלפה היא $\Theta(n)$.

שאלה 4

בתשובה לשאלה זו אשתמש בדו-תור המאפשר מחיקה של איברים משני הצדדים. השגרה להכנסה בתחילת הדו-תור תהיה HEAD-ENQUEUE, השגרה למחיקת האיבר בתחילת הדו-תור תהיה HEAD-DEQUEUE, השגרה למחיקת האיבר מסוף הדו-תור תהיה TAIL-DEQUEUE.

סיבוכיות הריצה של שגרות אלה תהיה $O(1)$ לפי שאלה 10.1-5.

עבור כל אחד מערי המערך, נעדכן את דו-התור. דו-התור יכיל את אינדקס הערך המקסימלי ב-k האיברים האחרונים ואת כל האינדקסים שיש להם "פוטנציאל" להיות מקסימלי ב-k האיברים הבאים, כלומר, כל האיברים שהאיבר הנוכחי לא גדול או שווה להם. כשעברנו את k האיברים הראשונים במערך, נתחיל לעדכן את מערך התוצאה על פי הסדר באיבר המקסימלי שתמיד נמצא בראש דו-התור.

MAX_IN_K(A, k):

```
1.  Create new array B[1, …, n - k + 1];
2.  Q ← Create new dequeue;
3.  for j ← 1 to length[A] do:
4.      if head[Q] ≠ tail[Q] and key[head[Q]] ≤ j - k:
5.          HEAD-DEQUEUE(Q);
6.      while head[Q] ≠ tail[Q] and A[j] ≥ A[tail[Q]] do:
7.          TAIL-DEQUEUE(Q);
8.      TAIL-ENQUEUE(j);
9.      If j ≥ k then:
10.         B[j – k + 1] ← A[head[Q]]
11. return B
```

לכל j כך ש $1 \leq j \leq n-k+1$. B[j] יהיה שווה הערך המקסימלי מ-j ו-k האיברים העוקבים ל-j

לכן, בחלק הראשון של האלגוריתם (שורות 1-2) נגדיר את B בתור מערך התשובה שלנו ודו-תור Q.

בלולאה שבשורות 3-10, עבור כל איבר j במערך A נעדכן את Q כך שתכיל את האיבר הגדול ביותר מבין האיברים שיש לבדוק (j עד j+k-1) ואת האינדקסים של האיברים הקטנים ממנו הבאים אחריו ויש להם פוטנציאל להיות מקסימליים (כלומר, שאין אחריהם מספר גדול מהם בטווח שלנו). כלומר, בתחילת כל איטרציה של הלולאה, head[Q] תמיד יהיה האינדקס של הערך המקסימלי ב-k האיברים האחרונים ב-A ואחריו יהיו האינדקסים לפי הסדר של האיברים הקטנים ממנו, המופיעים אחריו ושעוד לא מצאנו ערך גדול מהם (ברגע שנראה מספר גדול מהם, הם מאבדים את הסיכוי להיות האיבר המקסימלי בטווח ולכן נוציא אותם מהדו-תור). וב-B[j - k + 1], אם j≥k, הערך המקסימלי יישמר.

<u>אתחול:</u> באיטרציה הראשונה של הלולאה הדו-תור ריק ולכן שורות 4-7 לא מתקיימות. לכן האינדקס 1 נכנס לדו-תור. אם k=1 אזי j=k ולכן האיבר הראשון הוא האיבר המקסימלי ערכו יישמר ב-B[1]. אם k>1, שורות 9-10 לא ירוצו ואכן נסיים את האיטרציה כאשר בhead[Q] הערך המקסימלי עד כה.

<u>תחזוקה:</u> נניח כי תנאי שמורת הלולאה נכון בתחילת האיטרציה ה-i של הלולאה. אם הדו-תור לא ריק, ראשית נבדוק שהאינדקס הראשון שנמצא בא רלוונטי (כלומר, שייך ל-k האיברים האחרונים במערך), אם לא, נוציא את האיבר הראשון- שאר האיברים בוודאות בטווח כי השלב הזה מתבצע בכל איטרציה (שורות 4-5). לאחר מכן (שורות 6-7), כל עוד הדו-תור לא ריק, נמחק מסוף הדו-תור את כל האיברים הקטנים או שווים לאיבר הנוכחי, מכיוון שהם מאבדים את הסיכוי להיות האיבר המקסימלי בטווח זה ובכל טווח עתידי המכיל אותם (כל k איברים שנבחר מכאן והלאה שיכילו את אחד מהאינדקסים בדו-תור, יכילו גם את האינדקס הנוכחי).

לאחר מכן, בשורה 8, נוסיף את האינדקס הנוכחי לדו-תור, מכיוון שיש לו פוטנציאל להיות המקסימלי בטווח הזה או בטווחים עתידיים ובשורות 9-10 נבדוק אם עברנו מעל k איברים. אם כן, האיבר שבראש הדו-תור הוא האיבר המקסימלי ב-k האיברים האחרונים ולכן נכניס אותו לאיבר המתאים ב-B. אם לא, נמשיך ונחפש אחר האיבר המקסימלי הראשון.

<u>סיום:</u> באיטרציה האחרונה, j=n (בהנחה שזו היא לא האיטרציה הראשונה, אם כן, הוכחתי נכונות באתחול) הדו-תור מלא מכיוון שהוא מכיל את האיבר המקסימלי ב-k האיברים האחרונים. בדומה להוכחה בתחזוקה, בשורות 4-8, ניקינו את הדו-תור מאינדקסים שלא שייכים ל-k האיברים האחרונים ואיברים הקטנים או שווים לאיבר הנוכחי והוספנו את האיבר הנוכחי לסוף הדו-תור.

$k \leq n$ ולכן האיבר הראשון בדו-תור הוא בעל הערך הגדול ביותר ב-*k* האיברים האחרונים ב-*A* ולכן נוסיף את ערכו ב-*A* למקום המתאים במערך *B*.

לבסוף, נחזיר את המערך *B* המלא.

שורות 1-2 ו-11. רצות בסיבוכיות של $\Theta(1)$. ללולאה בשורות 3-10 יש *n* איטרציות מכיוון שהיא עוברת לינארית על איברי המערך *A*. בשורות 6-7 ישנה לולאת *while* שמוחקת את האיברים בדו-תור שקטנים או שווים לערך הנוכחי אבל בכל איטרציה נכנס לדו-תור לכל היותר איבר אחד לכן ל**אורך כל האיטרציות**, ללולאה בשורות 6-7 יכולות להיות *n* איטרציות לכל היותר. כלומר, סיבוכיות הריצה של הלולאה בשורות 3-10 היא $\theta(n)$.

לכן, סיבוכיות הריצה של שגרה זו היא $\theta(n)$.

שאלה 5

סעיף א

פונקציית הגיבוב h לא מחויבת לחלק את האיברים לפי גודלם ולכן אין שום הבטחה שהאיברים שנכנסו לאיבר מסוים יהיו קטנים יותר מהאיברים שמקושרים לאיבר שאחריו. דוגמא נגדית:

נגדיר את המפתחות להיות U=[1,2,3,4,5]. כלומר n=5. ונסמן m=3 כך ש- h(k) = k mod 3.

כלומר, קיים T[0,1,2].

אזי לאחר השורה הראשונה באלגוריתם נקבל את טבלת הגיבוב:

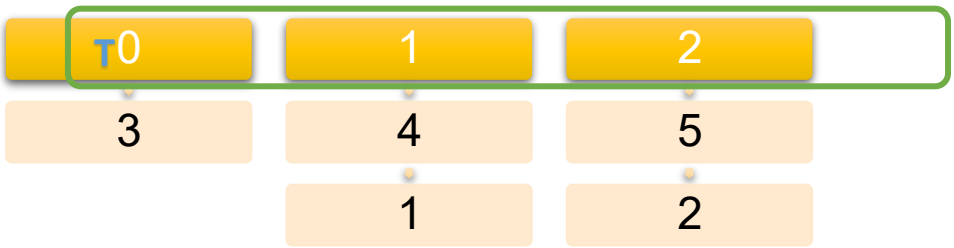
h(1) = 1 mod 3 = 1;

h(2) = 2 mod 3 = 2;

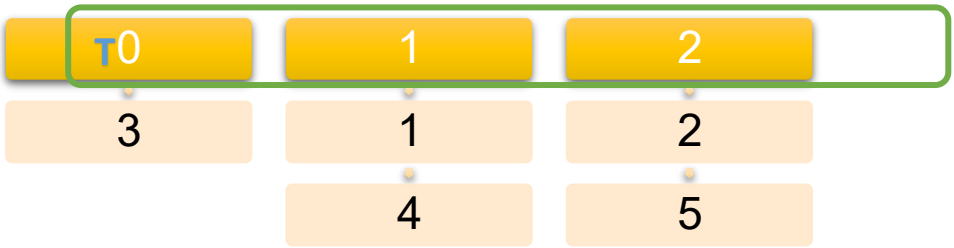
h(3) = 3 mod 3 = 0;

h(4) = 4 mod 3 = 1;

h(5) = 5 mod 3 = 2;



ולאחר השורה השנייה בה אנו ממיינים את הרשימות נקבל:



בשורה השלישית נצרף את הרשימות מהתא הראשון ועד האחרון למערך *T*':

מקום מקורי ב-T	0		1		2
הערכים ב-*T*'	**3**	**1**	**4**	**2**	**5**
אינדקס הערכים ב-*T*'	1	2	3	4	5

המערך אינו ממוין (האיבר הראשון 3 נמצא לפני האיבר השני 1 אך $3 > 1$ ולכן זו היא דוגמא נגדית.

סעיף ב

השורה הראשונה של האלגוריתם עוברת בצורה לינארית על המפתחות ומכניסה כל אחד מהמפתחות למקום המתאים בטבלת הגיבוב בסיבוכיות ריצה של $\Theta(1)$. לכן, סיבוכיות הריצה של השורה הראשונה באלגוריתם היא $\Theta(n)$.

בהנחה שפונקציית הגיבוב טובה (ע"פ עמוד 193, ההסתברות שמפתח כלשהו יגובב לתא מסוים זהה עבור כל *m* התאים) אזי בכל אחד מ-*m* התאים בטבלת הגיבוב ישנם $\frac{n}{m}$ איברים.

שורה 2 של האלגוריתם ממיינת את הרשימות באמצעות מיון מיזוג. סיבוכיות הריצה של מיון מיזוג עבור *k* איברים היא $\Theta(klg(k))$ ולכן עבור $\frac{n}{m}$ איברים היא $\Theta\left(\frac{n}{m}\lg\left(\frac{n}{m}\right)\right)$ עבור כל ערימה ומכיוון שיש *m* רשימות למיין במיון מיזוג אזי סיבוכיות הריצה של השורה השנייה היא $\Theta\left(m * \frac{n}{m}\lg\left(\frac{n}{m}\right)\right) = \Theta\left(\text{nlg}\left(\frac{n}{m}\right)\right)$

השורה השלישית עוברת בצורה לינארית על *n* הערכים ומצרפת אותם למערך אחד ולכן סיבוכיות הריצה שלה היא $\Theta(n)$.

לסיכום, תוחלת זמן הריצה של האלגוריתם היא $\Theta\left(n + n\lg\left(\frac{n}{m}\right) + n\right) = \Theta(n\lg(\frac{n}{m}))$

אם m=n אזי מתקיים $\Theta\left(n\lg\left(\frac{n}{m}\right)\right) = \Theta\left(n\lg\left(\frac{n}{n}\right)\right) = \Theta(n\lg(1)) = \Theta(n)$ ולכן תוחלת זמן הריצה היא לינארית.

סעיף ג

בכל מקרה, זמן הריצה של השורה הראשונה לא משתנה מהסעיף הקודם, ולכן סיבוכיות זמן הריצה במקרה הגרוע של שורה 1 הוא $\Theta(n)$.

במקרה הגרוע, כל n האיברים מגובבים לאותו תא בטבלת הגיבוב לכן יש n איברים ברשימה המקושרת לאיבר מסוים מתוך m האיברים בטבלת הגיבוב.

לכן, בשורה השנייה, ניאלץ למיין רשימה אחת מקושרת בעלת n איברים במיון מיזוג ולכן סיבוכיות זמן הריצה של התהליך הזה יהיה $\Theta(nlgn)$.

בשורה השלישית, נעבור על n האיברים ונצרף אותם על פי הסדר ולכן בכל מקרה סיבוכיות זמן הריצה של שלב זה יהיה לינארי.

לכן לסיכום, סיבוכיות זמן הריצה של האלגוריתם במקרה הגרוע יהיה: $\Theta(n + nlgn + n) = \Theta(nlgn)$

בסדר

סעיף ד

מסעיף ב', על מנת שסיבוכיות זמן הריצה תהיה לינארית יש לבחור ב-m=n.

לכל $k \in [0, n^2 - 1]$ נגדיר את פונקציית הגיבוב להיות $h(k) = \left\lfloor \frac{k}{n} \right\rfloor$.

מכיוון $0 \le \mathrm{k} < n^2$ אז $0 \le \frac{\mathrm{k}}{n^2} < 1$ ומכיוון שהאיברים מתפלגים באופן אחיד בתחום, לפי עמוד 193, פונקציית גיבוב המקיימת את תנאי הגיבוב הפשוט והאחיד היא $h(k) = \left\lfloor \frac{k}{n^2} * m \right\rfloor = \left\lfloor \frac{k}{n^2} * n \right\rfloor = \left\lfloor \frac{k}{n} \right\rfloor$.

לפי סעיף ב', מכיוון שפונקציית הגיבוב מקיימת את התנאי, סיבוכיות זמן הריצה של האלגוריתם תהיה לינארית.

יהיו $k_1, k_2 \in [0, n^2 - 1]$, נניח $k_1 \le k_2$ אזי $\frac{k_1}{n} \le \frac{k_2}{n}$ ולכן $h(k_1) = \left\lfloor \frac{k_1}{n} \right\rfloor \le \left\lfloor \frac{k_2}{n} \right\rfloor = h(k_2)$.

כלומר, עבור כל איבר ברשימה המקושרת ששייכת לאיבר i בטבלת הגיבוב, כל שנמצאים ברשימה המקושרת ששייכת לאיבר i+1, אם קיים כזה, גדולים ממנו וכל האיברים שנמצאים ברשימה המקושרת ששייכת לאיבר i-1, אם קיים כזה, קטנים ממנו. לכן, לאחר מיון הרשימות המקושרות, כאשר נצרף את האיברים לפי הסדר שלהם בטבלת הגיבוב נקבל רשימה ממוינת, בדומה לתהליך במיון דלי.

אין סתירה בין סעיף זה לסעיף א' מכיוון שפונקציית הגיבוב שבחרנו בסעיף זה שומרת על סדר האיברים ומבטיחה לנו לכל מפתח k שאיברים גדולים ממנו יתגבבו לאיבר שלו בטבלת הגיבוב או לאיברים שבאים אחריו (כלומר, שהם יהיו אחריו כשנצרף את הרשימות הממוינות) ואיברים הקטנים ממנו יתגבבו לאיבר שלו בטבלת הגיבוב או לאיברים שבאים אחריו (כלומר, הם יהיו לפניו כשנצרף את הרשימות הממוינות).