1 א מכניסים את איברי הקבוצה לטבלת גיבוב בגודל $m = n$ עם שרשור ופונקציית גיבוב כלשהי שמקיימת בקירוב את הנחת הגיבוב הפשוט. רצים על הטבלה ולכל איבר x, מחפשים את $z - x$. אם האיבר שמצאנו שונה מ-x אז סיימנו, אחרת עוברים לאיבר הבא בטבלה ושוב מחפשים כמו מקודם.

מקדם העומס של הטבלה הוא $\alpha = 1$, לכן לפי משפטים 11.1 ו-11.2 בספר, תוחלת זמן החיפוש בטבלת הגיבוב היא $\Theta(1)$. מספר החיפושים הוא לינארי ב-n ובסה"כ תוחלת זמן הריצה של האלגוריתם היא $\Theta(n)$.

1 ב הפעם גודל הטבלה הוא $m = n^2$ ומכניסים אליה את סכומי $\binom{n}{2} = \frac{n^2-n}{2}$ הזוגות בקבוצה. נכניס את איברי הקבוצה למערך עזר A ואז נקרא לשגרה הבאה:

```
1:  procedure SumOfFours(A, z)
2:      create empty hash table T of size n^2
3:      for i ← 1 to n − 1 do
4:          for j ← i + 1 to n do
5:              s = A[i] + A[j]
6:              if lookup z − s in T ≠ nil then
7:                  return True
8:          ▷ insert pair of sums into T
9:          for k ← 1 to i − 1 do
10:             s = A[i] + A[j]
11:             if lookup s in T = nil then
12:                 insert s into T
13:     return False
```

הלולאה שבשורות 4-8 רצה על כל הזוגות שמימין ל-i ומחפשת בטבלה את ההפרש של z והסכום של זוג האיברים. בשורות 10-11 אנחנו מכניסים את סכומי זוגות האיברים שנמצאים משמאל ל-i. כלומר כשמחפשים בטבלה בשורה 6 אז האינדקסים i, j שונים מכל זוגות האינדקסים שסכומם נמצא בטבלה. לכן אם החיפוש הצליח, מובטח שמצאנו 4 איברים שונים שסכומם z.

תוחלת זמן החיפוש בטבלה היא $\Theta(1)$ שכן מספר האיברים בטבלה הוא $\binom{n}{2} = \Theta(n^2)$ ולכן מקדם העומס הוא $\alpha = \frac{\Theta(n^2)}{m} = \frac{\Theta(n^2)}{n^2} = \Theta(1)$. אנחנו עושים $\Theta(n^2)$ חיפושים בטבלה ולכן תוחלת זמן הריצה של $\textsc{SumOfFours}$ היא $\Theta(n^2)$.

2 א יהי A_i המאורע "נוצרה שרשרת באורך 4 בתא ה-i" עבור $i = 1, \ldots, m$. אז $Pr\{A_i\} = \frac{1}{m^4}$. ההסתברות המבוקשת היא

$$Pr\{A_1 \cup \ldots \cup A_m\} \underset{\substack{\downarrow \\ \text{המאורעות זרים}}}{=} Pr\{A_1\} + \cdots + Pr\{A_m\} = m \cdot \frac{1}{m^4} = \frac{1}{m^3}$$

2 ב נסתכל על הכנסת k_3: שני תאים בטבלה כבר תפוסים ואנחנו רוצים לדעת מהי ההסתברות שנצטרך לבדוק 3 תאים לפני ש-k_3 יוכנס. כדי שזה יקרה, בבדיקה הראשונה אנחנו צריכים לבדוק את אחד משני התאים התפוסים וזה יקרה בהסתברות $\frac{2}{m}$ (תחת ההנחה שהגיבוב הוא אחיד). בבדיקה השניה אנחנו צריכים לבדוק את התא התפוס הנותר וזה כאמור יקרה בהסתברות $\frac{1}{m-1}$.

ההסתברות המבוקשת אם כך היא $\frac{2}{m(m-1)}$.

2 ג אם מקדם העומס הוא $\alpha = 1 - \frac{1}{\lg n}$, אז לפי משפט 11.6 תוחלת זמן החיפוש עבור חיפוש כושל היא:

$$\Theta(\frac{1}{1-\alpha}) = \Theta(\frac{1}{1-(1-\frac{1}{\lg n})}) = \Theta(\lg n)$$

3 נסתכל על עץ מלא בגובה h, יש לו 2^h עלים שכל אחד מהם בעומק h. נקבל ש-$\sum_{i=1}^{2^h} \frac{1}{2^h} = 1$. נתבונן בעלה מסוים l_i, אם נסיר את l_i מהעץ אז נאבד מהסכום $\frac{1}{2^h}$ ושאר האיברים ישארו ללא שינוי ולכן הסכום נשאר ≤ 1. בצורה דומה הסכום ישאר קטן שווה מ-1 אם נסיר את שאר העלים, למעט המקרה שבו הסרנו זוג עלים שהם גם אחים. אם הסרנו זוג כזה אז איבדנו מהסכום $2 \cdot \frac{1}{2^h} = \frac{1}{2^{h-1}}$ אבל אז האבא שלהם הוא עכשיו עלה בעומק $h-1$, כלומר יש להוסיף לסכום $\frac{1}{2^{h-1}}$ וסה"כ הוא נותר ללא שינוי. בצורה רקורסיבית אפשר לראות שהדבר תקף לכל עלה בעומק d.

הראינו שאם מתחילים מעץ מלא אז הסכום המבוקש הוא 1 וכאשר מסירים עלים בעומקים שונים הסכום נשאר אותו הדבר או שהוא קטן. מאחר וניתן לצאת מעץ מלא ולהגיע לכל עץ נתון ע"י מחיקת צמתים שלא בעץ הנתון, האמור לעיל מוכיח את הנדרש.

הסכום שווה בדיוק ל-1 אם לכל צומת בעץ יש בדיוק 0 או 2 בנים.

4 נשתמש בטענת העזר הבאה:

טענה: אם לצומת z בעץ חיפוש בינארי שני בנים, אזי לעוקב לו אין בן שמאלי.

הוכחה: נניח ש-y העוקב של z. אם ל-y היה בן שמאלי x, אז בשגרה Tree-Successor היינו מחזירים את Tree-Minimum של תת העץ ששורשו y, שהוא הבן הכי שמאלי בתת עץ זה, כלומר x. זאת בסתירה לכך ש-y הוא העוקב של z.

אם כך לפי האלגוריתם המתואר בשאלה לצומת y אין בן שמאלי. תת העץ ששורשו $left[z]$ קטן מ-y ולכן כשמחליפים אותו עם $left[y]$ (**nil=**) אז תכונת עץ החיפוש עדיין מתקיימת עבור y. כעת ל-z יש רק בן ימני ואפשר לטפל בו כמו במקרה השני. כל הפעולות מלבד מציאת העוקב y לוקחות זמן קבוע ולכן זמן הריצה במקרה הגרוע הוא $\Theta(h)$ לעץ בגובה h.

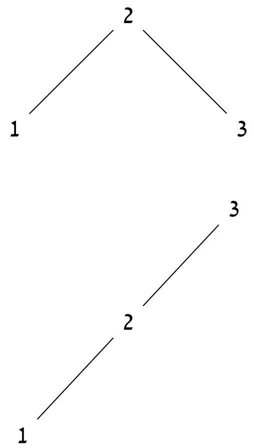
להבדיל מהשיטה שבספר שלא נוגעת בתת העץ השמאלי של הצומת שנמחק, שיטה זו מזיזה את כולו לבן הימני. לכן שיטה זו עלולה לגרום לשינוי יחסית רציני במבנה של העץ ולפגוע באיזון שלו.

חיסרון של השיטה שמוצגת בספר הוא שהיא מעתיקה את y למקום אחר בעץ. אז כל ההצבעות שהיו ל-y כבר לא תקינות.

5 בסריקה סופית מבקרים בשורש אחרון. לכן בסדרה הנתונה שורש העץ הוא k_n. לפי תכונת עץ החיפוש הבינארי, כל האיברים הקטנים מ-k_n בסדרה נמצאים בתת העץ השמאלי שלו, וכל האיברים הגדולים ממנו בתת העץ הימני. האיבר האחרון בכל תת סדרה כזאת הוא שורש של תת העץ הרלוונטי, כאשר בצורה דומה למקודם כל האיברים שקטנים ממנו בתת הסדרה הם בתת העץ השמאלי, והשאר בימני. ממשיכים בצורה כזאת עד שלא נשארים עוד איברים בסדרה לעבור עליהם וכך אפשר לבנות את העץ המקורי מתוצאה של סריקה סופית.

בסריקה תחילית מבקרים בשורש ראשון, אז האיבר הראשון בסדרה הוא שורש העץ וכמו מקודם כל הקטנים ממנו הם בתת העץ השמאלי והגדולים בימני וממשיכים כמו מקודם רק שהשורש של כל תת עץ כזה הוא האיבר הראשון בתת סדרה (ולא האחרון).

אי אפשר לשחזר עץ חיפוש בינארי מהסריקה התוכית שלו. לדוגמה שני העצים האלה שונים:



אבל הסריקה התוכית שלהם זהה: $[1,2,3]$.