מבני נתונים
פתרונות לסט שאלות דומה לשאלות מתרגיל 6

השאלות

1. הראו דוגמה של ערימת מקסימום בת n איברים שבה פעולת DeleteMax( ) דורשת $\Omega(\log n)$ פעולות.
שימו לב, דרושה דוגמה כללית ל- n כלשהו.

2. נסתכל על האלגוריתם הבא למיון מערך A בעל n איברים.

(i) נחלק את A ל- $\sqrt{n}$ תת מערכים $A_1, A_2, ..., A_{\sqrt{n}}$ כך שתת המערך A_1 יכיל את $\sqrt{n}$ האיברים הראשונים ב- A, תת המערך A_2 יכיל את $\sqrt{n}$ האיברים הבאים ב- A, וכך הלאה.

(ii) לכל i, $1 \le i \le \sqrt{n}$:
- אם $|A_i| < 4$ נמיין את A_i בעזרת bubble sort.
- אחרת נמיין את A_i ברקורסיה.

(iii) נמזג את תתי המערכים הממוינים $A_1, A_2, ..., A_{\sqrt{n}}$ למערך ממוין אחד בעזרת אלגוריתם למיזוג k מערכים ממוינים (אצלנו $k = \sqrt{n}$) הבא:

a. נכניס את האיבר הראשון מכל מערך (כלומר הקטן ביותר) לתור עדיפויות מינימום בגודל k.

b. נבצע n איטרציות כדלקמן:

i. נוריד את האיבר המינימלי בתור (בעזרת DeleteMin) ונדפיס אותו. נסמן את האיבר שהורדנו ב- x ונניח ש- x הגיע לתור מתת המערך A_i.

ii. נכניס לתור העדיפויות (בעזרת Insert) את האיבר שבא אחרי x ב- A_i (אם קיים איבר נוסף ב- A_i).

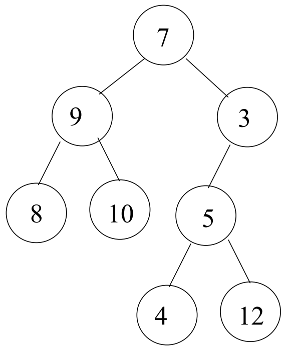
הניחו שתור העדיפויות ממומש ע"י ערימת מינימום, ושלב a ממומש בעזרת האלגוריתם של Floyd, הבונה ערימה בזמן לינארי.

חשבו חסם הדוק במונחי Θ לזמן הריצה של אלגוריתם המיון שתואר, לפי ההנחיות הבאות:

כתבו נוסחת נסיגה לזמן הריצה ופתרו אותה על ידי **עץ רקורסיה**. יש לשרטט את העץ בצורה ברורה ולהראות את כל החישובים המתבצעים על העץ.

3. נתון עץ בינארי T שבקדקודים שלו יש מספרים שלמים. העץ ממומש על ידי מצביעים לילדים.
כתבו אלגוריתם OrderedLayers(Tree T) שידפיס את תוכן הקדקודים של T, רמה אחרי רמה החל מהשורש, כאשר בכל רמה יודפסו המספרים בסדר עולה מהקטן לגדול.

דוגמה: עבור העץ הבא יודפס הפלט הבא משמאל לימין 12 ,4 ,10 ,8 ,5 ,9 ,3 ,7.



לשם כתיבת האלגוריתם השתמשו בשני תורי קדימויות מינימום Q1, Q2 המכילים זוגות מהצורה (נתונים, מפתח), כאשר שדה הנתונים יכול להכיל קדקוד בעץ (או מצביע לקדקוד), ואילו המפתחות הם מספרים שלמים. סדר הקדימויות בכל תור נקבע כמובן לפי שדה המפתח. לכל תור קדימויות כזה מוגדרות הפעולות הבסיסיות הבאות:

- MakeEmpty( ) – לרוקן את התור.
- IsEmpty ( ) – מחזירה 1 אם התור ריק, 0 אחרת.
- Insert(int key, TreeNode data) – מכניסה את הזוג (key, data) לתור הקדימויות.
- Min( ) – מחזירה את הזוג (key, data) שהמפתח שלו מינימלי מבין כל המפתחות בתור.
- DeleteMin( ) – מורידה מהתור את הזוג (key, data) שהמפתח שלו מינימלי מבין כל המפתחות בתור ומחזירה את הזוג כערך.

בנוסף אתם יכולים להשתמש רק בזיכרון נוסף **בגודל קבוע** (**אין להשתמש ברקורסיה**).

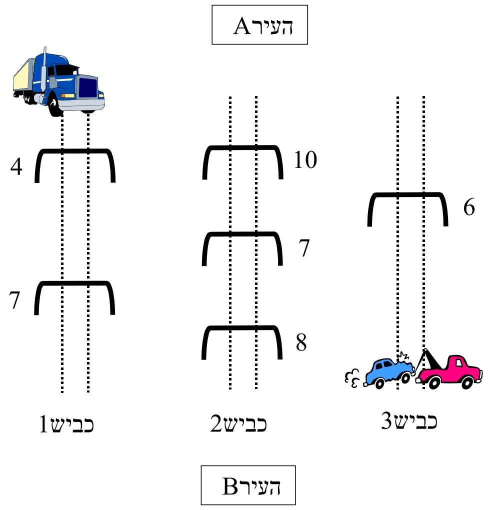
נתחו גם את יעילותו של האלגוריתם במקרה הגרוע בהנחה שתור הקדימויות ממומש על ידי ערימת מינימום.

שימו לב, האלגוריתם שלכם ישתמש ב- Q1,Q2 ובפעולות המוגדרות עליהם **כבקופסאות שחורות!**

4. כתבו את הפונקציה Delete(int ind) שמוחקת את האיבר הנמצא בתא ה- ind במערך המכיל ערימת מינימום. הפונקציה תתקן כמובן את הערימה. מה יעילות הפונקציה?

5. נתונות שתי ערים A,B וביניהן יש m כבישים מקבילים (לא נחתכים), הממוספרים מ- 1 ועד m. במהלך הזמן בונים על הכבישים גשרים, כאשר על כל כביש יכול להיות מספר כלשהו (לא מוגבל) של גשרים, והגובה של כל גשר יכול להיות מספר ממשי חיובי כלשהו.

דוגמה: במקרה זה יש m = 3 כבישים בין A ל- B, כאשר על כל כביש מצוירים הגשרים שנבנו עליו, וליד כל גשר רשום גובה הגשר. מספר הגשרים הכולל במקרה זה הוא n = 6.



עליכם לתכנן מבנה נתונים שתומך בפעולות הבאות:

- Init() – אתחול מבנה הנתונים כך שיכיל m כבישים ללא גשרים עליהם.
  יעילות הפעולה צריכה להיות O(m).
- AddBridge(float h, int r) – הוספת גשר שגובהו h על כביש מספר r.
  היעילות צריכה להיות O(log m) במקרה הגרוע.
- WhichRoad(float heigt) – הפונקציה תחזיר מספר של כביש שבו יכולה לנסוע משאית שגובהה height (המשאית תיסע מתחת לגשרים ואסור לה כמובן להיתקל באף אחד מהגשרים שעל הכביש הזה. לכן גובה המשאית צריך להיות קטן ממש מגובה כל הגשרים שעל הכביש). אם יש כמה כבישים שבהם המשאית יכולה לעבור, יוחזר מספרו של אחד מהם. אם המשאית אינה יכולה לעבור באף כביש תחזיר הפונקציה 0.
  יעילות הפעולה צריכה להיות Θ(1) במקרה הגרוע.
- Print(int r) – הפונקציה מדפיסה את הגבהים של כל הגשרים שנמצאים על כביש מספר r.
  יעילות הפונקציה צריכה להיות ליניארית במספר הגשרים שעל הכביש.

תארו את מבנה הנתונים וכתבו אלגוריתם לכל אחת מהפעולות. הסבירו מדוע יעילות כל אחת מהפעולות היא כנדרש.

6. נתון קובץ שבו מספר **סוגי** התווים השונים הוא n (הקובץ יכול להכיל כמובן כל תו כמה פעמים). נניח שבנינו עץ הופמן לצורך קידוד הקובץ הנתון. כמה צלעות יהיו בעץ שבנינו כפונקציה של n? (שימו לב, אתם מתבקשים לתת תשובה מדויקת ולא במונחים של Θ). **הוכיחו** תשובתכם.

7. א. מצאו קוד הופמן עבור השכיחויות הבאות: a: 1, b:1, c: 2, d: 3, e: 5, f: 8. מצאו גם את משקלו של הקוד.

ב. הוכיחו או הפריכו את הטענה הבאה:
נתון קוד הופמן עבור סדרה של שכיחויות ותווים. יהיו a,b שני תווים כך שאורך מילת הקוד של a קטן ממש מאורך מילת הקוד של b. אז בהכרח השכיחות של a קטנה ממש מהשכיחות של b.

ג. הוכיחו או הפריכו את הטענה הבאה:
נתון קוד הופמן עבור סדרה של שכיחויות ותווים. יהיו a,b שני תווים עם אותו אורך של מילת קוד. אז השכיחות של a שווה לשכיחות של b.

פתרונות נבחרים

<u>שאלה 1</u>:
נתבונן בערמת מקסימום שנשמרת כרגיל במערך והיא: (n, n−1, n−2, …, 2, 1).
זו ערמה חוקית מכיוון שכל קדקוד גדול משני ילדיו. פעולת DeleteMax תוריד את השורש n ותשים במקומו את 1 שהוא האיבר האחרון במערך (ולכן גם העלה הימני ביותר ברמה האחרונה). עתה פעולת FixHeap(0) (שהיא חלק מפעולת DeleteMax) תבצע החלפה מהשורש עד לרמת העלים, ולכן מספר הפעולות יהיה $\Omega(\log n)$.

<u>שאלה 2</u>:
עבור $n \geq 4$

$$T(n) = \sqrt{n} \cdot T(\sqrt{n}) + \Theta(\sqrt{n}) + \Theta(2n\log(\sqrt{n}))$$

ועבור $n < 4$

$$T(n) = \Theta(1)$$

<u>הסבר:</u>
שלב (i) הינו רק סימון ואינו לוקח זמן.

שלב (ii) נותן את תנאי העצירה ($n < 4$), ומתאר את הרקורסיה: יש למיין ברקורסיה כל A_i. יש $\sqrt{n}$ תתי מערכים A_i וגודלו של כל אחד הוא $\sqrt{n}$. מכאן המחובר הראשון $\sqrt{n} \cdot T(\sqrt{n})$ בנוסחה.

חלק a בשלב (iii) בונה תור מינימום בגודל $\sqrt{n}$ בעזרת האלגוריתם של Floyd. מכאן המחובר השני $\Theta(\sqrt{n})$ בנוסחה.

בחלק b בשלב (iii) אנו מבצעים n איטרציות שבכל אחת מתבצעות פעולות DeleteMin ו- Insert על תור בגודל $\sqrt{n}$. מכאן המחובר השלישי $\Theta(2n\log(\sqrt{n}))$ בנוסחה.

פישוט הנוסחה נותן

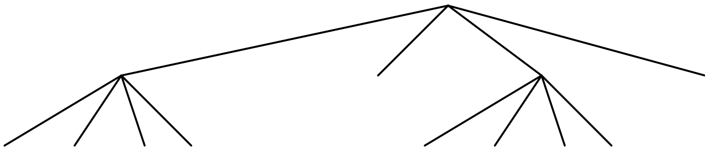
$$T(n) = \sqrt{n} \cdot T(\sqrt{n}) + \Theta(n\log n)$$

ובהתעלמות מהקבועים:

$$T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n\log n$$

עץ הרקורסיה:

רמה	גודל בעיה בכל צומת	מס' צמתים	עלות הרמה
0	n	1	n log
1	$n^{1/2}$	$n^{1/2}$	$n^{1/2}(n^{1/2}\log n^{1/2})$
2	$n^{1/4}$	$n^{3/4}$	$n^{3/4}(n^{1/4}\log n^{1/4})$
log log n	4	n/4	n



כאשר את מספר הרמה האחרונה h = log log n - 1 חישבנו ע"י הנוסחה $n^{(1/2)^h} = 4$.
סה"כ עלות כל הרמות (כלומר זמן הריצה של האלגוריתם) היא לכן

$$\sum_{h=0}^{\log\log n-1} n\log(n^{(1/2)^h}) = n\log n \sum_{h=0}^{\log\log n-1}(1/2)^h = \Theta(n\log n)$$

כאשר השוויון האחרון נובע מכך שה- $\sum$ היא סידרה הנדסית שסכומה גדול מ- 1 וקטן מ- 2.

שאלה 3:

תאור עילי של אלגוריתם OrderedLayers(Tree T):

נסרוק את העץ רמה רמה. אחרי סריקת הצמתים בעומק i-1, תור אחד יהיה ריק והתור השני יכיל את כל הצמתים בעומק i. (בתחילת האלגוריתם אחד התורים יכיל את שורש העץ - הצומת היחיד בעומק 0, והתור השני יהיה ריק). באיטרציה הבאה נוציא ונדפיס את האברים מהתור המלא עד שיתרוקן, ולכן יודפס תוכן הצמתים ברמה זאת בסדר עולה. באותו מעבר נכניס לתור השני את כל הילדים של הצמתים בעומק i כלומר את כל הצמתים בעומק i+1. תפקידי התורים יתחלפו לסירוגין (נוציא אברים מאחד מהם ונכניס לשני), עד ששניהם יתרוקנו.

תאור מפורט של אלגוריתם OrderedLayers(Tree T)

1. ניצור שני תורי עדיפות מינימום ריקים Q1 Q2
2. נכניס ל Q1 את השורש של T
3. כל זמן ש Q1 איננו ריק בצע:
   א. כל זמן ש Q1 איננו ריק בצע:
      1. השמט (DeleteMin) זוג (key,node) מ Q1
      2. הדפס את key
      3. אם ל node יש בן שמאלי, הכנס אותו ל Q2
      4. אם ל node יש בן ימני, הכנס אותו ל Q2
   ב. כל זמן ש Q2 איננו ריק בצע:
      1. השמט (DeleteMin) זוג (key,node) מ Q2
      2. הדפס את key
      3. אם ל node יש בן שמאלי, הכנס אותו ל Q1
      4. אם ל node יש בן ימני, הכנס אותו ל Q1

<u>הערות:</u>

1) בפתרון המוצג כאן אבר בתור הוא מצביע לצומת בעץ, והעדיפות היא המספר בצומת.
2) בתיאור האלגוריתם "הכנס את הצומת n לתור Q" משמעותו Q.Insert(key,data) כאשר data הוא מצביע לצומת n ו key הוא המספר בצומת הזה.

וריאציות אחרות לאלגוריתם: תנאי הלולאה החיצונית יכול להיות "כל זמן שאחד התורים איננו ריק". במקום הלולאה הפנימית השנייה אפשר להעביר את תוכן Q2 ל Q1 (למשל בלולאה כל עוד Q2 לא ריק, שמשמיטה מינימום מ Q2 ומכניסה ל Q1). פתרון אפשרי אחר הוא להשתמש באחד התורים ברמת הצומת בתור העדיפות, ובאופן כזה לזהות מתי הושלם טיפול ברמה מסוימת.

יעילות:

עבור כל צומת בעץ מתבצעת פעולת הכנסה Insert לתור עדיפויות, ופעולת השמטה DeleteMin אחת. כל אחת מהפעולות האלה תיקח זמן $\Theta(\log m)$ כש- m הוא מספר האברים בתור בשלב זה. מכיוון שיש n אברים, והתור לא יכיל יותר מ n אברים, זמן הריצה הוא $O(n \log n)$.
במקרה הגרוע, העץ הנתון הוא שלם, ובמעבר על הרמה הנמוכה ביותר התור יכיל את כל העלים, שהם כמחצית מספר הצמתים בעץ n/2, כך שהחסם הוא הדוק, וזמן הריצה הוא $\Theta(n \log n)$.

<u>שאלה 4</u>:

ניתן את האלגוריתם בלבד לתיקון ערימת מינימום לאחר פעולת delete(int i) שמורידה את האיבר שנמצא במקום ה- i.
האלגוריתם הוא שילוב של האלגוריתמים ל- Insert ו- FixHeap.

1) נשים במקום ה- i את האיבר האחרון במערך שנסמנו ב- x, ונעדכן את מס' אברי הערמה.
2)
   i. אם x גדול מההורה שלו וקטן מילדיו אז זוהי ערימת מינימום חוקית וסיימנו.
   ii. אחרת, אם x קטן מההורה שלו אז עלינו לתקן את הערמה רק כלפי מעלה (כי x במקרה זה קטן מילדיו, כי כבר ההורה שלו היה קטן מילדיו).
   התיקון הוא בדומה ל- Insert: כל עוד x קטן מההורה שלו נחליף ביניהם.
   iii. אחרת, x גדול מאחד מילדיו לפחות. עלינו לתקן את הערמה רק כלפי מטה (כי x במקרה זה גדול מההורה שלו). התיקון זהה ל- FixHeap(i).

יעילות האלגוריתם היא O(logn) כי במקרה הגרוע מתקנים כל הדרך מעלה לשורש או כל הדרך משורש לעלה, וגובה הערימה הוא $\Theta(\log n)$.

<u>שאלה 5</u>:

מבנה הנתונים: אנו נשמור לכל כביש את כל הגבהים של הגשרים שנבנו עליו, וכן את גובהו של הגשר המינימלי עליו. בנוסף, נחזיק ערימת מקסימום שבה נשמור לכל כביש את גובהו של הגשר המינימלי עליו. הדבר יאפשר לנו בקלות למצוא מיהו הכביש שגובה הגשר המינימלי עליו גבוה ככל האפשר. ביתר דיוק מבנה הנתונים יכיל את המרכיבים הבאים:

1. ערימת מקסימום שתישמר במערך Heap[m], כאשר בכל איבר בערימה יישמרו השדות הבאים:
   א. rnum – מספר כביש.
   ב. min – גובה גשר מינימלי על כביש מספר rnum.
   שדה המפתח בערימה יהיה השדה min ואילו השדה rnum יהיה שדה נתונים.
2. מערך Roads[m], כאשר בתא i במערך יישמרו השדות הבאים:
   א. index - מספר התא במערך Heap שבו נשמר כביש מספר i.
   ב. List – רשימה מקושרת חד-כיוונית של הגשרים על כביש i.

הנחה: נניח שהתאים של המערכים ממוספרים מ- 1 עד m ולא מ- 0 עד m–1 כמקובל ב- C.

אתחול $Init()$:

<u>אלגוריתם</u>: עבור $i = 1,\dots,m$:

$Heap[i].rnum = i$.

$Heap[i].min = \infty$.

$Roads[i].List = NULL$.

$Roads[i].index = i$.

<u>יעילות האלגוריתם</u>: היא כמובן $\Theta(m)$ כנדרש.

<u>הערה</u>: אתחלנו את השדה min להכיל מספר גדול מאוד ∞, שפירושו שכל משאית יכולה לעבור על הכביש כי אין עליו גשרים. לחילופין, אפשר לאתחל את השדה min להכיל ערך 0 שאינו ערך חוקי של גובה של גשר, אבל אז יש לדאוג לעדכן את הפעולות הרגילות המוגדרות על ערימת מקסימום כך שיתייחסו אל הערך 0 כאל ערך הגדול מכל ערך אחר.

:AddBridge(float hb, int r)

<u>הרעיון</u>: נוסיף גשר חדש שגובהו hb לכביש מספר r בערימה ונתקן כמובן את הערימה.

<u>אלגוריתם</u>:

1. נוסיף את hb כאיבר חדש לראש הרשימה $Roads[r].List$.
2. יהי $j = Roads[r].index$ מספר התא שבו נמצא כביש מספר r בערימת המקסימום.
3. אם $hb < Heap[j].min$ אז:
   א. נעדכן $Heap[j].min = hb$.
   ב. נתקן את הערימה על ידי קריאה ל- $FixHeap(j)$.

<u>הערה</u>: הפונקציה $FixHeap(int\ j)$ תתקן את תת-הערימה שהשורש שלה נמצא בתא מספר j במערך Heap בדיוק כפי שלמדנו בכיתה, אולם כל פעם שיש החלפה בין שני תאים בערימה, היא תיגש גם למערך Roads ותחליף בין הערכים המתאימים. כלומר, אם מחליפים בערימה את התאים שמכילים את כבישים מספר s ו- t, אז נחליף בין $Roads[s].index$ ל- $Roads[t].index$.

<u>יעילות</u>: הכנסת איבר לראש רשימה לוקחת $\Theta(1)$ פעולות. לכן, יעילות האלגוריתם נשלטת על ידי היעילות של FixHeap. מכיוון שהערימה שלנו מכילה m איברים, הרי גובהה $\Theta(\log m)$, ולכן היעילות תהיה $O(\log m)$.

:WhichRoad(float hr)

<u>אלגוריתם</u>:

1. יהי $max = Heap[1].min$.
2. אם $hr < max$ אז נחזיר את $Heap[1].rnum$.
3. אחרת נחזיר 0.

<u>יעילות</u>: הפעולה היא כמובן $\Theta(1)$.

:Print(int r)

<u>אלגוריתם</u>: נדפיס את כל איברי הרשימה $Roads[r].List$.

<u>יעילות</u>: הפעולה היא כמובן ליניארית במספר הכבישים על כביש מספר r.

<u>הערה:</u>

a) שימו לב שאחרי בנית הערמה בשלב ה- Init, הפעולה היחידה המעדכנת אותה היא הקטנת ערך של איבר הנמצא בה (אין שימוש בפעולות העדכון הרגילות של ערמה Insert ו- DeleteMax).

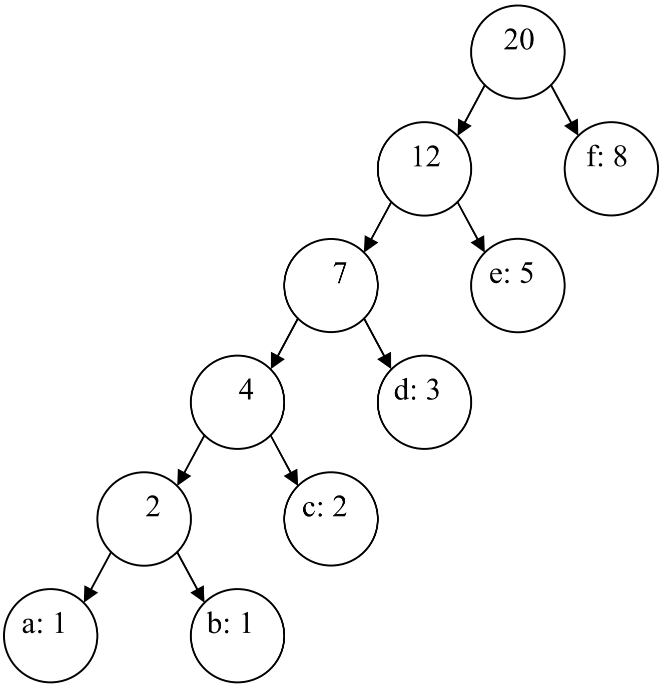
b) נקרא לפעולה המוזכרת בסעיף הקודם DecreaseValue. נסו להראות שאם בנית הערמה לוקחת $O(n)$ זמן, אז אחת הפעולות Max או DecreaseValue חייבות לקחת $\Omega(\log n)$ זמן.

<u>שאלה 6</u>

הוכחנו (תרגיל 4, שאלה 4א) שבעץ בינארי מספר העלים גדול ב- 1 ממספר הצמתים בעלי דרגה 2. מכיוון שעץ הפמן הוא מלא ו- n התווים הם בעלים, נקבל שמספר הצמתים הפנימיים בעץ הוא n-1. עץ בעל 2n-1 צמתים מכיל 2n-2 צלעות.

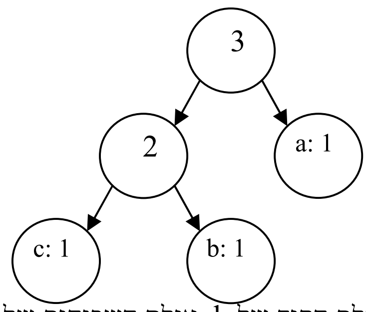
<u>שאלה 7</u>

א. עץ הפמן שמתאים לשכיחויות: a: 1, b:1, c: 2, d: 3, e: 5, f: 8 הוא:



משקל הקוד: $45 = 8*1 + 5*2 + 3*3 + 2*4 + 1*5 + 1*5$.

ב. הטענה כמובן אינה נכונה. דרושה דוגמה נגדית.
נתבונן בתווים a: 1, b:1, c:1.
הנה עץ הפמן מתאים:



אכן אורך מילת הקוד של a קטן ממש מאורך מילת הקוד של b, אולם השכיחות של a שווה לזו של b.

ג. הטענה אינה נכונה. דוגמה נגדית היא a: 1, b:2. בעץ הפמן שמתקבל לשני התווים יש אותו אורך של מילת קוד, אולם השכיחות של a אינה שווה לזו של b.