פתרון ממ"ן 11

<u>שאלה 1</u>

<u>טיוטה</u>

הרעיון בשאלה זו הוא לכתוב שגרה שהיא גרסה של אלגוריתם החיפוש הבינארי (עמוד 17 במדריך הלמידה). במקום לערוך השוואה של ערכים, ולבדוק האם האיבר שאנו בודקים קטן מהערך המבוקש או גדול ממנו, נבדוק האם המקום שאנו בודקים במערך נמצא לפני ℓ או אחריו.

תשובה

בפתרון לשאלה זו נציג אלגוריתם רקורסיבי, שהוא גרסה של אלגוריתם חיפוש בינארי, המקבל כקלט מערך A, עבורו מובטח שקיים ערך $1 \leq \ell \leq n$ כמתואר בשאלה, והמוצא את ℓ.

בכל שלב ברקורסיה, האלגוריתם מחזיק בשני מצביעים $left$ ו $right$ לשני מיקומים במערך. לכל אורך הריצה מתקיים $left \leq right$ והאלגוריתם עוצר כאשר $left = right$. בכל שלב, אם $left = right$, האלגוריתם מחזיר את $left$. אחרת, האלגוריתם בודק את הערך המופיע במערך במיקום $middle$ שהוא אמצע התת־מערך המוגבל על ידי $left$ משמאל ו $right$ מימין. אם $A[middle] < A[middle+1]$, האלגוריתם ממשיך את החיפוש בחצי הימני של התת־מערך. אחרת, האלגוריתם ממשיך את החיפוש בחציו השמאלי. האלגוריתם מתואר באופן פורמלי בפסאודוקוד הבא.

```
recSummit(A, left, right)

if left = right
    return left
middle ← ⌊(left+right)/2⌋
if A[middle] < A[middle + 1]
    then return recSummit(A, middle + 1, right)
return recSummit(A, left, middle)
```

```
summit(A[1, ..., n])

return recSummit(A, 1, n)
```

הטענה המרכזית שאנחנו רוצים להוכיח על האלגוריתם היא הטענה הבאה [1].

משפט 1 (נכונות האלגוריתם) בהנתן מערך $A[1,\ldots,n]$, אם קיים $1 \leq \ell \leq n$ כך שלכל $1 \leq j < \ell$ מתקיים $A[j] < A[j+1]$ וגם לכל $\ell \leq j < n$ מתקיים $A[j] > A[j+1]$ אז האלגוריתם מוצא את ℓ.

הטענה השנייה שנרצה להוכיח היא הטענה הבאה.

משפט 2 (סיבוכיות האלגוריתם) זמן הריצה של האלגוריתם במקרה הגרוע ביותר הוא $\Theta(\log n)$.

נניח כי קיים ℓ כמתואר במשפט 1. תנאי העצירה של האלגוריתם הוא $left = right$. נרצה להראות כי בסיום הריצה מתקיים כי $left = right = \ell$.

לצורך כך, נסמן ב $left_i, right_i$ את הערכים של המשתנים $left, right$ בתחילת הקריאה הרקורסיבית ה־i בהתאמה, ונוכיח את הטענה הבאה[2].

טענה 3 לכל i, בקריאה הרקורסיבית ה־i ל recSummit מתקיים כי $1 \leq left_i \leq \ell \leq right_i \leq n$.

הוכחה נוכיח הטענה באינדוקציה על i.

$i = 1$ - בקריאה הראשונה לאלגוריתם recSummit מתקיים כי

$$1 = left_1 \leq \ell \leq right_1 = n \ .$$

$i \to i+1$ - נניח נכונות הטענה עבור i, ונניח כי מתבצעת קריאה נוספת ל recSummit. לפי הנחת האינדוקציה, $1 \leq left_i \leq \ell \leq right_i \leq n$. אילו מתקיים $left_i = right_i$, אז מהתנאי בשורה 1 נובע כי לא תתבצע קריאה נוספת ל recSummit. לכן $left_i < right_i$, ועל כן

$$1 \leq left_i \leq middle = \left\lfloor \frac{left_i + right_i}{2} \right\rfloor \leq right_i - 1 < right_i \leq n \ .$$

בפרט נובע כי $middle < n$, ועל כן $middle + 1 \leq n$.

<u>מקרה 1: נניח כי $A[middle] < A[middle+1]$</u>

אז מההנחה על ℓ נובע כי $middle < \ell$, ועל כן

$$1 \leq left_i \leq left_{i+1} = middle + 1 \leq \ell \leq right_i = right_{i+1} \leq n \ .$$

[1] לפעמים טענה כזו נקראת באופן כללי "נכונות האלגוריתם". חשוב לשים לב שהביטוי "נכונות האלגוריתם" כוונתו שהאלגוריתם מחשב את הפונקציה שרצינו שיחשב. בכל מקרה בו מוכיחים נכונות, יש לפרט במדויק מה הטענה שאנחנו מעוניינים להוכיח. לא מספיק לכתוב "הוכחת נכונות".

[2] טענות כדוגמת טענה 3, הטוענות כי תכונה מסויימת מתקיימת בכל קריאה רקורסיבית או בכל שלב (איטרציה) בריצה נקראות שמורות לולאה. כטענה מתמטית אין שום דבר המייחד אותן, והוכחתן היא כהוכחת כל טענה מתמטית אחרת. המונח שמורת לולאה הוא מונח קונספטואלי בלבד, ואין צורך של ממש להזכיר אותו בהוכחה.

מקרה 2: נניח כי $A[middle] > A[middle+1]$

אז מההנחה על ℓ נובע כי $middle \geq \ell$, ועל כן

$$1 \leq left_i = left_{i+1} \leq \ell \leq middle = right_{i+1} \leq right_i \leq n \ .$$

$\square$

הטענה הבאה תסייע לנו לחסום את מספר הקריאות הרקורסיביות שמבצע האלגוריתם. יש לשים לב כי הטענה מתקיימת בין אם z מופיע במערך ובין אם לאו. בהוכחה של טענה זו איננו מניחים ש z מופיע במערך.

טענה 4 לכל i, בקריאה הרקורסיבית ה־i ל recSummit מתקיים כי $right_i - left_i \leq \frac{n}{2^{i-1}}$.

הוכחה נוכיח את הטענה באינדוקציה על i.

$i = 1$ ־ בקריאה הראשונה לאלגוריתם recSummit מתקיים כי $left_1 = 1$ ו $right_1 = n$. על כן $right_1 - left_1 \leq n = \frac{n}{2^{1-1}}$.

$i \to i+1$ ־ נניח נכונות הטענה עבור i, ונניח כי מתבצעת קריאה נוספת ל recSummit. לפי הטענה הקודמת, $left_i \leq right_i$. אילו מתקיים $left_i = right_i$, אז מהתנאי בשורה 1 נובע כי לא תתבצע קריאה נוספת ל recSummit. לכן $left_i < right_i$, ועל כן

$$left_i \leq middle = \left\lfloor \frac{left_i + right_i}{2} \right\rfloor \leq right_i - 1 < right_i \ .$$

מקרה 1: נניח כי $A[middle] < A[middle+1]$

אז $left_{i+1} = middle + 1 \leq right_i = right_{i+1}$, וכמו כן,

$$right_{i+1} - left_{i+1} = right_i - (middle+1) \leq right_i - \frac{left_i + right_i}{2} = \frac{1}{2}(right_i - left_i) \ ,$$

ולפי הנחת האינדוקציה נובע כי $right_{i+1} - left_{i+1} \leq \frac{n}{2^i}$.

מקרה 2: נניח כי $A[middle] > A[middle+1]$

אז $left_{i+1} = left_i \leq middle = right_{i+1}$, וכמו כן,

$$right_{i+1} - left_{i+1} = middle - left_i \leq \frac{left_i + right_i}{2} - left_i = \frac{1}{2}(right_i - left_i) \ ,$$

ולפי הנחת האינדוקציה נובע כי $right_{i+1} - left_{i+1} \leq \frac{n}{2^i}$. $\square$

כעת נוכל להוכיח את משפטים 2,1.

הוכחת משפט 1 מכיוון שבתחילת הריצה, $right_1 - left_1 = n$, ניתן להסיק מטענה 4 כי לאחר לכל היותר

$\log n + 2$ איטרציות האלגוריתם עוצר. לכל i, בקריאה הרקורסיבית ה־i מתקיים $left_i \leq \ell \leq right_i$.

בפרט, בקריאה האחרונה מתקיים $left_i \leq \ell \leq right_i = left_i$ ועל כן $left = \ell$. $\square$

הוכחת משפט 2 מכיוון שבכל קריאה רקורסיבית האלגוריתם מבצע מספר קבוע של פעולות, ומכיוון שלאחר לכל היותר $\log n + 2$ איטרציות האלגוריתם עוצר, נובע כי זמן הריצה של האלגוריתם במקרה הגרוע ביותר הוא $O(\log n)$.

עבור המערך A המוגדר על ידי

$$A[1] = 1,\ A[2] = n,\ A[3] = n-1,\ A[4] = n-2, \ldots, A[n] = 2 \quad ,$$

ניתן להוכיח באינדוקציה כי לכל $i \leq \log n - 1$ מתקיים כי $left_i = 1$. מכיוון שבאיטרציה האחרונה מתקיים $left = 2$ (לפי משפט 1), נובע כי מספר האיטרציות שמבצע האלגוריתם הוא לפחות $\log n - 1$, ועל כן זמן הריצה של האלגוריתם במקרה הגרוע ביותר הוא $\Omega(\log n)$.

מכאן כי זמן הריצה של האלגוריתם במקרה הגרוע ביותר הוא $\Theta(\log n)$. $\square$

$\checkmark$

שאלה 2

טיוטה
מומלץ לתת את הדעת לעובדה שמספר טענות בשאלה עושות שימוש באותם נימוקים בדיוק. בכתיבת ההוכחה כדאי להכליל את הטיעונים ולכתוב טענות עזר.

תשובה

נוכיח תחילה מספר טענות כלליות. תהאנה $f, g : \mathbb{N} \to (0, +\infty)$.

למה 5 אם $f = o(g)$ אז $f = O(g)$ וגם $f \neq \Omega(g)$.

הוכחה נניח כי $f = o(g)$. מהגדרה נובע כי קיים $n_0 \in \mathbb{N}$ כך שלכל $n \geq n_0$, $f(n) \leq g(n)$. לכן $f = O(g)$.

יהיו $c > 0$ ו $n_0 \in \mathbb{N}$. מהגדרת $o(g)$ קיים $n_1 \in \mathbb{N}$ כך שלכל $n \geq n_1$ מתקיים $f(n) \leq 2cg(n)$. על כן עבור $n = \max\{n_0, n_1\}$ מתקיים כי $n \geq n_0$ וגם $f(n) < cg(n)$. לכן $f \neq \Omega(g)$. $\square$

באופן דומה נוכיח את הלמה הבאה.

למה 6 אם $f = \omega(g)$ אז $f = \Omega(g)$ וגם $f \neq O(g)$.

א. נתבונן ב $c_1 = 1, c_2 = 2$ וב $n_0 = 10^{10}$. יהי $n \geq n_0$ אז

$$f(n) = n^2 \leq n^2 + 10^{10}n = g(n) = n^2 + n_0 \cdot n \leq n^2 + n^2 = 2f(n) \quad ,$$

ועל כן $f = \Theta(g)$ ממשפט 3.1 בספר נובע כי $f = O(g)$ וגם $f = \Omega(g)$. מלמות 5, 6 נובע כי $f \neq o(g)$ וגם $f \neq \omega(g)$.

ב. נוכיח כי $f = o(g)$. לצורך כך, יהי $c > 0$. על פי מדריך הלמידה, (עמוד 34 למעלה) נקבל כי $\log^{12} n = o(n^{0.1})$. על כן קיים $n_0 \in \mathbb{N}$ עבורו לכל $n \geq n_0$ מתקיים כי $\log^{12} n < cn^{0.1}$. נכפול ב n ונקבל כי לכל $n \geq n_0$ מתקיים $f(n) = n \log^{12} n < cn^{1.1} = c \cdot g(n)$. על כן $f = o(g)$, ולפי למה 5 נובע כי $f = O(g)$. כמו כן נובע כי $f \neq \Omega(g)$. על פי משפט 3.1 נובע כי $f \neq \Theta(g)$ ועל פי למה 6 נובע כי $f \neq \omega(g)$.

ג. נתבונן ב $c_1 = c_2 = 1$ וב $n_0 = 1$. יהי $n \geq n_0$ אז

$$f(n) = 5^{\log n} = \left(2^{\log 5}\right)^{\log n} = 2^{\log 5 \cdot \log n} = \left(2^{\log n}\right)^{\log 5} = n^{\log 5} = g(n) \quad ,$$

ומכאן $f(n) \leq g(n) \leq f(n)$. נובע כי $f = \Theta(g)$ ועל כן ממשפט 3.1 בספר $f = O(g)$ וגם $f = \Omega(g)$. מלמות 5, 6 נובע כי $f \neq o(g)$ וגם $f \neq \omega(g)$.

ד. נוכיח כי $f = o(g)$. לצורך כך, יהי $c > 0$. על פי מדריך הלמידה, (עמוד 34 למעלה) נקבל כי $\log n = o(n^{10})$. על כן קיים $n_1 \in \mathbb{N}$ עבורו לכל $n \geq n_1$ מתקיים כי $\log n < cn^{10}$. נסמן $n_0 = \max\{10^{10}, n_1\}$. נקבל כי לכל $n \geq n_0$ מתקיים $f(n) = \log n < cn^{10} = c \cdot g(n)$. על כן $f = o(g)$, ולפי למה 5 נובע כי $f = O(g)$. כמו כן נובע כי $f \neq \Omega(g)$. על פי משפט 3.1 נובע כי $f \neq \Theta(g)$ ועל פי למה 6 נובע כי $f \neq \omega(g)$.

$\checkmark$

שאלה 3

טיוטה

בגישה הראשונה לפתרון השאלה, נראה מאד טבעי להוכיח ישירות באינדוקציה את נכונות האלגוריתם. ליתר דיוק, הטענה אותה היינו רוצים להוכיח היא:

לכל מספר טבעי n, לכל מערך A באורך n, אם ב A יש איבר המופיע יותר מ $n/2$ פעמים, אז האלגוריתם מחזיר את האיבר הזה.

אם ננסה להוכיח את הטענה הזו ישירות, ההוכחה "נתקעת" בשלב המעבר של האינדוקציה, במקרה בו n איזוגי.

דרך אחת לקבל תחושה כיצד האלגוריתם עובד היא לחקור מעט את ההתנהגות של אלגוריתם פשוט יותר, שאינו מחליף איברים כאשר המונה מתאפס. אם האיבר הנבדק שווה למועמד, מוסיפים 1 למונה, ואחרת מורידים 1 מהמונה. כדאי לשחק מעט עם האלגוריתם הפשוט, ולגלות שבכל איטרציה מתקיים כי ערכו של המונה שווה להפרש בין מספר האיברים השווים ל $A[1]$ לבין מספר האיברים השונים מ $A[1]$ עד לאיטרציה זו.

טיוטה

לאחר ש"משחקים" עם האלגוריתם, ומריצים אותו על מספר קלטים, ניתן להיווכח בשתי עובדות חשובות.

1. אם $counter$ נשאר חיובי לאורך כל ההרצה, אז האיבר הראשון במערך בהכרח חוזר יותר מ $n/2$ פעמים, ואכן האלגוריתם מחזיר אותו.

2. האלגוריתם הוא, במובן מסוים, חסר זכרון. ליתר דיוק, אם באיטרציה כלשהי, $counter$ מתאפס, אז מהאיטרציה הבאה האלגוריתם מתחיל "מחדש", והמשך הריצה שלו אינו תלוי בתחילתה.

המשמעות המעשית של התכונה השנייה היא שאין כל דבר מיוחד באיבר הראשון בתכונה הראשונה. למעשה, עבור כל איבר j, אם בתחילת האיטרציה ה j מתקיים $counter = 0$ ובמהלך כל האיטרציות $j, j+1, ...k$ עבור $j < k$ כלשהו, $counter$ לא מתאפס, אז האיבר $A[j]$ מופיע יותר מ $\frac{k-j}{2}$ פעמים בתת מערך $A[j, j+1, .., k]$.

מהשילוב של שתי התכונות הללו נובעת תכונה שלישית, מעט עדינה יותר. אם באיטרציה j כלשהי $counter$ התאפס, אז האיבר הראשון המערך מופיע בדיוק $j/2$ פעמים בתת מערך $A[1, ..., j]$. מהתכונה השנייה שכתבנו, נובע שמכאן מתחיל האלגוריתם "מחדש".

האלגוריתם עובר על כל המערך, ועל כל איבר במערך מבצע מספר קבוע של פעולות. על כן זמן הריצה של האלגוריתם הוא $\Theta(n)$.

תשובה

הטענה המרכזית שאנחנו רוצים להוכיח על האלגוריתם היא הטענה הבאה.

משפט 7 (נכונות האלגוריתם) בהנתן מערך $A[1, \ldots, n]$, אם קיים ערך x המופיע במערך יותר מ $n/2$ פעמים, אז האלגוריתם מחזיר את x.

הטענה השנייה שנרצה להוכיח היא הטענה הבאה.

משפט 8 (סיבוכיות האלגוריתם) זמן הריצה של האלגוריתם במקרה הגרוע ביותר הוא $\Theta(n)$.

לצורך הוכחת משפט 7 נניח כי קיים ערך x המופיע במערך יותר מ $n/2$ פעמים. נסמן ב $counter_i$ את הערך של המשתנה $counter$ בסיום האיטרציה ה i. לשם פשטות הטיעונים נסמן גם $counter_0 = 0$. כמו כן נסמן ב $candidate_i$ את הערך של המשתנה $candidate$ בסיום האיטרציה ה i. נוכיח תחילה את הטענה הבאה.

למה 9 יהיו $1 \le j < k \le n$, ונניח כי $counter_{j-1} = 0$ וכי $counter_i > 0$ לכל $j \le i < k$. אז $A[j]$ מופיע $\frac{counter_k+k-j+1}{2}$ פעמים בתת מערך $A[j, j+1, ..., k]$[3] ויותר על כן, $candidate_i = A[j]$ לכל $j \le i \le k$. בפרט, אם $k = n$, האלגוריתם מחזיר את $A[j]$.

הוכחה נוכיח את הטענה באינדוקציה על $k - j \ge 1$. עבור המקרה $k - j = 1$, מכיוון ש $counter_{j-1} = 0$, כלומר בתחילת האיטרציה ה j מתקיים $counter = 0$, נובע כי התנאי בשורה 5 מתקיים, ועל כן $counter_j = 1$ ו $candidate_j = A[j]$. לכן באיטרציה ה $k = j + 1$ התנאי בשורה 5 אינו מתקיים. אם $A[k] = A[j]$ אז $counter_k = counter_j + 1 = 2$, ואכן, $2 = \frac{2+1+1}{2} = \frac{counter_k+k-j+1}{2}$. אחרת, $A[k] \neq A[j]$ ועל כן $counter_k = counter_j - 1 = 0$, ואכן, $2 = \frac{2+1+1}{2} = \frac{counter_k+k-j+1}{2}$.

נניח נכונות הטענה עבור $k - j$ ונוכיח עבור $k - j + 1$. על כן, נניח כי $counter_{j-1} = 0$ וכי $counter_i > 0$ לכל $j \le i < k+1$. מהנחת האינדוקציה, $candidate_i = A[j]$ לכל $j \le i \le k$. כעת, מכיוון ש $counter_k > 0$, התנאי בשורה 5 אינו מתקיים באיטרציה ה $k+1$, ועל כן $candidate_{k+1} = A[j]$. כמו כן $A[j]$ מופיע $\frac{counter_k+k-j+1}{2}$ פעמים בתת מערך $A[j, j+1, ..., k]$. מכיוון ש $counter_k > 0$, באיטרציה ה $k+1$ התנאי בשורה 5 אינו מתקיים. אם $A[k+1] = A[j] = candidate_k$ אז $counter_{k+1} = counter_k + 1$ ואכן, מספר הפעמים ש $A[j]$ מופיע בתת מערך $A[j, j+1, ..., k+1]$ הוא

$$\frac{counter_k+k-j+1}{2} + 1 = \frac{counter_k+k-j+1+2}{2} = \frac{(counter_k+1)+(k+1)-j+1}{2} = \frac{counter_{k+1}+(k+1)-j+1}{2} .$$

אם $A[k+1] \neq A[j] = candidate_k$ אז $counter_{k+1} = counter_k - 1$ ואכן, מספר הפעמים ש $A[j]$ מופיע בתת מערך $A[j, j+1, ..., k+1]$ הוא

$$\frac{counter_k+k-j+1}{2} = \frac{counter_{k+1}+1+k-j+1}{2} = \frac{counter_{k+1}+(k+1)-j+1}{2} .$$

□

הוכחת משפט 7

נסמן ב $0 = j_0 < j_1 < \ldots < j_\ell \le n$ את מספרי האיטרציות שבסופן $counter = 0$. מהלמה נובע כי לכל $1 \le i \le \ell$ מתקיים כי בתת מערך $A[j_{i-1}+1, j_{i-1}+2, \ldots, j_i]$ מתקיים כי $A[j_{i-1}]$ מופיע $\frac{j_i - j_{i-1}}{2}$ פעמים בדיוק. לכן x מופיע בתת מערך $A[j_{i-1}+1, j_{i-1}+2, \ldots, j_i]$ לכל היותר $\frac{j_i - j_{i-1}}{2}$ פעמים. נובע כי x מופיע לכל היותר $\frac{j_\ell}{2}$ פעמים בתת מערך $A[1, 2, \ldots, j_\ell]$. מכאן כי $j_\ell < n$, וכמו כן, x מופיע יותר מ $\frac{n - j_\ell}{2}$ פעמים בתת מערך $A[j_\ell+1, j_\ell+2, \ldots, n]$, ושוב על פי הלמה, נובע כי $candidate_n = x$, ומכאן כי האלגוריתם מחזיר את x. □

[3] הביטוי הזה נראה מוזר בתחילה, אבל האינטואיציה מאחוריו די פשוטה. אם $counter$ אינו מתאפס, אז הוא שווה בדיוק להפרש בין מספר הפעמים ש $A[j]$ מופיע, למספר האיברים בתת מערך שאינם שווים ל $A[j]$. אם מסדרים את המשוואה הזו, מקבלים את הביטוי עבור מספר הפעמים ש $A[j]$ מופיע בתת מערך.

הוכחת משפט 8 האלגוריתם עובר על כל אחד מאברי המערך, ועבור כל איבר מבצע לכל הפחות פעולה אחת, ולכל היותר 7 פעולות. על כן לכל מערך באורך n מתקיים כי זמן הריצה של האלגוריתם הוא לכל הפחות n ולכל היותר $7n$. על כן זמן הריצה הוא $\Theta(n)$. □

✓

<u>שאלה 4</u>

תשובה

נגדיר $f, g : \mathbb{N} \to \mathbb{N}$ באופן הבא. לכל $n \in \mathbb{N}$,

$$f(n) = \begin{cases} n! & n \in \mathbb{N}_{even} \\ (n-1)!+1 & n \in \mathbb{N}_{odd} \end{cases}, g(n) = \begin{cases} n! & n \in \mathbb{N}_{odd} \\ (n-1)!+1 & n \in \mathbb{N}_{even} \end{cases}.$$

נוכיח כי f עולה. יהי $n \in \mathbb{N}$.

<u>מקרה 1: נניח כי $n \in \mathbb{N}_{even}$</u>

אז $f(n+1) = ((n+1)-1)!+1 = n!+1 > n! = f(n)$.

<u>מקרה 2: נניח כי $n \in \mathbb{N}_{odd}$</u>

אז $f(n+1) = (n+1)! > (n-1)!+1 = f(n)$.

לכן f עולה. באופן דומה נוכיח כי g עולה. כעת נוכיח כי $f \neq O(g)$. לצורך כך, יהיו $c > 0$ ו $n_0 \in \mathbb{N}$. ידוע כי קיים מספר טבעי n_1 המקיים $n_1 \geq c$ (למשל $n_1 = \lceil c \rceil$). נסמן $m = \max\{n_0, n_1\} \in \mathbb{N}$ ונתבונן ב $n = 2m \in \mathbb{N}_{even}$. ראשית נשים לב כי $n > m \geq n_0$, וכמו כן $2m - 1 \geq m \geq n_1 \geq c$.

מהגדרת g מתקיים כי $g(n) = (n-1)!+1 = (2m-1)!+1$. מהגדרת f ומהטענות לעיל נובע כי

$$\begin{aligned} f(n) &= n! = (2m)! \\ &= 2m \cdot (2m-1)! \\ &= (2m-1)\cdot(2m-1)! + (2m-1)! \\ &\geq c \cdot (2m-1)! + c = c \cdot g(n) \end{aligned}$$

על כן $f \neq O(g)$. באופן דומה נוכיח כי $g \neq O(f)$. על כן הטענה איננה נכונה. ✓