דף סיכום בחינה

ציון	ניקוד מירבי	מספר שאלה
4.00	9.00	1.1
9.00	9.00	1.2
9.00	9.00	1.3
16.00	16.00	2
16.00	16.00	3.1
16.00	16.00	3.2
16.00	16.00	3.3
9.00	9.00	4

ציון בחינה סופי : 95.00

הבחינה הבדוקה בעמודים הבאים

שאלה 1 - עצי AVL

...

סעיף א

...

נוכיח באינדוקציה על גובה העץ.

...

מקרה בסיס $(\mathrm{h}=0)$:

נראה שזהו עץ בעל צומת יחיד, השורש.

גובה עץ זה הוא 0 **ועומק עץ זה הוא** 0 **גם כן, לכן נקבל** $0 \geq 0 \longrightarrow d\left(root\right) \geq \lceil \frac{h}{2} \rceil$.

...

הנחת האינדוקציה :

יהי AVL - T_k **בגובה** h_k.

נניח שעבור תתי העצים ששורשיהם הם הבנים של שורש העץ T_k

מתקיימים $d\left(x\right)_{left} \geq \lceil \frac{h_{left}}{2} \rceil$, $d\left(x\right)_{right} \geq \lceil \frac{h_{right}}{2} \rceil$ **עבור כל עלה** x **של העץ** T_k.

נוכיח שמתקיים $d\left(x\right)_k \geq \lceil \frac{h_k}{2} \rceil$ **עבור כל עלה** x **של העץ** T_k.

...

בסיס האינדוקציה$(\mathrm{h}>1)$:

מהנחת האינדוקציה, נקבל ש- $h_k - 2 \leq h_{right}, h_{left} \leq h_k - 1$ **(הפרש הגבהים בין הבנים הוא לכל היותר 1).**

מקרה 1 $(h_{right} \mid h_{left} = h_k - 1)$: **(לשם נוחיות, נגדיר** $p = right \mid left$**)**

מהנחת האינדוקציה, נקבל $d\left(x\right)_p \geq \lceil \frac{h_p}{2} \rceil = \lceil \frac{h_k - 1}{2} \rceil$.

כלומר, $d\left(x\right)_k = d\left(x\right)_p + 1 \geq \lceil \frac{h_p}{2} \rceil + 1 = \lceil \frac{h_k - 1}{2} \rceil + 1$.

כך שמתקיים $d\left(x\right)_k \geq \lceil \frac{h_k - 1}{2} \rceil + 1 \longrightarrow d\left(x\right)_k \geq \lceil \frac{h_k}{2} \rceil - \lceil \frac{1}{2} \rceil + 1 \longrightarrow d\left(x\right)_k \geq \lceil \frac{h_k}{2} \rceil$.

מקרה 2 $(h_{right} \mid h_{left} = h_k - 2)$: **(לשם נוחיות, נגדיר** $p = right \mid left$**)**

מהנחת האינדוקציה, נקבל $d\left(x\right)_p \geq \lceil \frac{h_p}{2} \rceil = \lceil \frac{h_k - 2}{2} \rceil$.

כלומר, $d\left(x\right)_k = d\left(x\right)_p + 1 \geq \lceil \frac{h_p}{2} \rceil + 1 = \lceil \frac{h_k - 2}{2} \rceil + 1$.

כך שמתקיים $d\left(x\right)_k \geq \lceil \frac{h_k - 2}{2} \rceil + 1 \longrightarrow d\left(x\right)_k \geq \lceil \frac{h_k}{2} \rceil - \lceil \frac{2}{2} \rceil + 1 \longrightarrow d\left(x\right)_k \geq \lceil \frac{h_k}{2} \rceil$.

...

נראה שהחסם הדוק עבור משפחת עצי AVL **מינימאלים.**

בעצים אלו המרחק בין השורש לעלה בעל העומק המינימאלי (x) **הוא הנמוך ביותר ולכן מתקיים** $d\left(x\right) = \lceil \frac{h}{2} \rceil$.

...

חסר הוכחה

סעיף ב

...

נראה שמחיקה של צומת מסויים בעץ AVL **מינימאלי דורשת לכל הפחות** $log\left(n\right)$ **סיבובים.**

...

נוכיח עבור המקרה של מחיקת העלה בעל העומק המינימאלי בעץ, בו נצטרך את מספר הסיבובים המקסימאלי ובכך ההוכחה תקפה עבור שאר הצמתים בעץ (עבור כל עלה אחר נצטרך מספר סיבובים נמוך יותר).

...

יהי AVL - T_n עץ מינימאלי בגובה $h_n = log\,(n)$.

נסמן ב $X_1, \cdots, X_p$ את המסלול משורש העץ (X_1) לעלה בעל העומק המינימאלי (X_p) .

מחיקה של העלה X_p בעץ AVL מינימאלי תוביל למצב שבו כלל הצמתים במסלול יוצאים מן מהאיזון כך ששורש העץ X_1 הוא בצומת הגבוה ביותר במסלול שבו מופר האיזון.

לפי פתרון סעיף א, נראה שעומק העלה X_p הוא $\lceil \frac{log(n)}{2} \rceil = \lceil \frac{h}{2} \rceil$.

לכן עלינו לבצע לכל הפחות $\lceil \frac{log(n)}{2} \rceil$ סיבובים (לכל צומת נדרשת פעולת סיבוב יחידה או פעולת סיבוב כפולה).

נראה שעבור $c \geq 2$ ו- $n_0 = 2$, נקבל $\lceil \frac{log(n)}{2} \rceil = \Omega\,(log\,(n))$.

אז עבור משפחת עצי AVL מינימאלים נקבל שהחסם התחתון למספר הסיבובים במחיקת עלה בעל עומק מינימאלי הוא כ- $log\,(n)$ סיבובים.

...

סעיף ג

...

נראה שתוספת של 2 מצביעים לשדות המחלקה של צומת העץ תתמוך בפעולת $Find\,(x, k)$ שמחזירה את k האיברים הראשונים שגדולים מהמפתח x בעץ בזמן $O\,(log\,(n) + k)$.

...

נוסיף למחלקת $node$ שדה בשם $successor$ המצביע על הצומת הבא בסדר $Inorder$ (עוקב)

ושדה בשם $predecessor$ המצביע על הצומת שלפניו בסדר $Inorder$ (קודם).

(עץ AVL הוא עץ חיפוש בינארי, לכן עבור סדר $Inorder$ נקבל סדרה מונוטונית עולה).

פעולת $Find\,(x, k)$ תכלול בתוכה פעולת $Search\,(x)$.

(פעולת חיפוש המחפשת לאחר המפתח x בעץ - זמן הריצה של פעולת חיפוש בעץ AVL הוא $O\,(log\,(n))$)

במקרה שבו לא נמצא את המפתח x בעץ $\leftarrow$ הפעולה $Find\,(x, k)$ תחזיר $null$.

במקרה שבו נמצא את המפתח x בעץ, נבצע לולאה עם k איטראציות לכל היותר $\leftarrow$

בכל איטראציה, נאחסן את המפתח שנתון בצומת הנוכחי ונעבור לעוקב הבא של הצומת הנוכחי.

(כאשר מספר האיברים הראשונים שגדולים מהמפתח x קטן מ- k, נצא מהלולאה לאחר שנעבור על כל איברים אלו)

נראה שזמן הריצה של פעולת $Search\,(x)$ הוא $O\,(log\,(n))$.

וגם כן נראה שזמן הריצה של לולאה בעלת k איטראציות הוא $O\,(k)$.

משילוב זמני הריצה של הפעולות (חיפוש ולולאה), נקבל שזמן הריצה של פעולת $Find\,(x, k)$ הוא $O\,(log\,(n) + k)$.

...

בפעולת $Insert$, בנוסף לפעולת ההכנסה $(O\,(log\,(n)))$, נקרא לפעולות $Successor$ ו- $Predecessor$.

(זמני הריצה של הפעולות $Successor$ ו- $Predecessor$ הם $O\,(log\,(n))$).

ולאחר מכן, נשמור את הפלטים בשדות $successor$ ו- $predecessor$ בהתאמה.

מחיבור זמני הריצה של כלל הפעולות, נקבל שזמן הריצה הוא $O\,(log\,(n))$.

...

בפעולת $Delete\left(node\right)$, בנוסף לפעולת המחיקה($O\left(log\left(n\right)\right)$), נעדכן את השדות של העוקב/הקודם של הצומת הנמחק, כך שהקודם של $node$ יוגדר כקודם של העוקב של $node$ $(node.successor.predecessor = node.predecessor)$ והעוקב של $node$ יוגדר כעוקב של הקודם של $node$ $(node.predecessor.successor = node.successor)$.

נראה שזמני ריצה של הגדרות אלו הם $O\left(1\right)$.

כך שמחיבור זמני הריצה של כלל הפעולות, נקבל שזמן הריצה הוא $O\left(log\left(n\right)\right)$.

...

שאלה 2 - קבוצות

...

נציע מבנה נתונים שמשלב עצי AVL שונים למימוש ה- ADT הבא.

עץ T_x שבו כל צומת מכיל שדה ששומר את הערך של האיבר הראשון בזוג המפתח (x)

ושדה המצביע של שורש העץ T_y (עבור כל מפתח x שונה, נקבל עבורו עץ T_y שונה)

כל צומת בעצי T_y מכיל שדה השומר את האיבר השני בזוג המפתח (y)

ושדה נוסף השומר את הערך של הגובה $(height)$.

...

פעולות $Init$ -

בפעולה זו נגדיר $root_x = null$.

...

פעולת $insert\left(x, y, height\right)$ -

נחלק למקרים :

מקרה 1 (המפתח x קיים בעץ T_x) - נבצע פעולת חיפוש של המפתח x בעץ T_xעם n איברים.

ולאחר מכן, נבצע פעולת הכנסה של המפתח y (והערך $height$) בעץ T_y עם $\leq n$ איברים.

מקרה 2 (המפתח x לא קיים בעץ T_x) - נבצע פעולת הכנסה של המפתח x בעץ T_xעם n איברים.

ולאחר מכן, נבצע פעולת הכנסה של המפתח y (והערך $height$) בעץ T_y (בתור שורש).

...

פעולת $delete\left(x, y\right)$ -

נבצע פעולת חיפוש של המפתח x בעץ T_x $\longleftarrow$ במקרה שבו המפתח x לא קיים בעץ T_x, נזרוק שגיאה.

במקרה שבו המפתח x אכן קיים בעץ T_x,

נבצע פעולת חיפוש של המפתח y בעץ T_y $\longleftarrow$ במקרה שבו המפתח y לא קיים בעץ T_y, נזרוק שגיאה.

במקרה שבו המפתח y אכן קיים בעץ T_y נחלק למקרים :

מקרה 1 (עץ T_y מכיל צומת אחת בלבד) - נמחק את הצומת ששומר את הערך x בעץ T_x.

מקרה 2 (עץ T_y מכיל יותר מצומת אחת) - נמחק את הצומת ששומר את הערך y בעץ T_y.

במידה ואנו מוחקים את שורש העץ T_y, נעדכן את המצביע של צומת x בהתאם לכך (הצבעה אל השורש החדש של העץ T_y).

...

פעולת $find\left(x,y\right)$ -
נבצע פעולת חיפוש של המפתח x בעץ T_x $\longleftarrow$ במקרה שבו המפתח x לא קיים בעץ T_x, נחזיר $null$.
במקרה שבו המפתח x אכן קיים בעץ T_x,
נבצע פעולת חיפוש של המפתח y בעץ T_y $\longleftarrow$ במקרה שבו המפתח y לא קיים בעץ T_y, נחזיר $null$.
במקרה שבו המפתח y אכן קיים בעץ T_y, נחזיר את הערך $height$ של הצומת בעץ T_y.

...

פעולת $PrintAll\left(x\right)$ -
נבצע פעולת חיפוש של המפתח x בעץ T_x $\longleftarrow$ במקרה שבו המפתח x לא קיים בעץ T_x, לא נדפיס דבר.
במקרה שבו המפתח x אכן קיים בעץ T_x, נבצע הדפסת $Inorder$ של העץ T_y.

...

ניתוח זמני ריצה -

1. אתחול -

פעולת האתחול דורשת זמן קבוע, לכן נקבל $O\left(1\right)$.

2. הכנסה -

עבור שני המקרים, נקבל שעבור עץ T_x עם n איברים, זמן הריצה של פעולת החיפוש/הכנסה הוא $O\left(log\left(n\right)\right)$.
ועבור עץ T_y עם $\leq n$ איברים, נקבל שזמן הריצה של פעולת ההכנסה הוא $O\left(log\left(n\right)\right)$.
מחיבור זמני הריצה של כלל הפעולות, נקבל שזמן הריצה הוא $O\left(log\left(n\right)\right)$ עבור פעולת ההכנסה.

3. מחיקה -

במקרים בהם לא נמצא את המפתח המבוקש בעץ T_x או בעץ T_y, אנו עוברים על כל גובה העץ.
לכן, נקבל זמן ריצה של $O\left(log\left(n\right)\right)$.
במקרים בהם אכן נמצא את המפתח המבוקש, נמחק צומת בעץ T_x או בעץ T_y .
נקבל שזמן הריצה ב- 2 מחיקות אלו הוא $O\left(log\left(n\right)\right)$.
נראה שעבור כל המקרים השונים, נקבל שזמן הריצה הוא $O\left(log\left(n\right)\right)$.

4. חיפוש -

פעולת חיפוש בעץ T_x או בעץ T_y עוברת לכל היותר על $log\left(n\right)$ צמתים (בעץ T_x וגם בעץ T_y),
לכן נקבל שזמן הריצה עבור פעולה זו הוא $O\left(log\left(n\right)\right)$.

5. הדפסה -

זמן הריצה עבור חיפוש המפתח x בעץ T_x הוא $O\left(log\left(n\right)\right)$
פעולת ההדפסה הינה פעולה רקורסיבית עם k קריאות פנימיות (כאשר k הוא מספר האיברים בעץ T_y),
לכן זמן הריצה של פעולת ההדפסה הינה $O\left(k\right)$.
מחיבור זמני הריצה של חיפוש האיבר והדפסת העץ, נקבל שזמן הריצה הכולל הוא $O\left(log\left(n\right)+k\right)$.

...

שאלה 3 - מיזוג/פיצול

...

סעיף א

...

נסמן את h_1 להיות הגובה של העץ T_1 ואת h_2 להיות הגובה של העץ T_2 ונחלק למקרים באופן הבא :

מקרה 1 $(h_1 > h_2)$ -

במסלול המתחיל משורש העץ T_1 ועד העלה הימני ביותר בעץ T_1, נחפש אחר האיבר הראשון (y) המקיים את $0 \leq h(y) - h(T_2) \leq 1$. כלומר, נחפש אחר המפתח הגדול ביותר הראשון

שבו גובהו שווה לגובה העץ T_2 או גדול ב־ 1 מגובה העץ T_2 .

תחילה נחליף את הצומת y ותת העץ שלו בצומת x . ולאחר מכן, נגדיר את הבן השמאלי של הצומת x את תת העץ של y

(נתון כי $maxT_1 < x < minT_2$, אז בהכרח מתקיים $y < x$)

ואת הבן הימני של הצומת x נגדיר להיות העץ T_2

(נתון כי $maxT_1 < x < minT_2$, ובפרט כל מפתח בעץ T_2 גדול מהמפתח x).

הוספת הצומת x עלולה להוציא את העץ החדש מן האיזון, לכן במקרה הגרוע ביותר נצטרך לבצע כ־ $2 * d(x)$ סיבובים

(מקרה שבו כל צומת במסלול משורש העץ החדש ועד לצומת x בעץ החדש, ידרוש סיבוב כפול).

כלומר, נקבל שמספר הסיבובים הנדרשים הוא $O(d(x))$.

בנוסף , נראה ש־ $d(x) = h(T_1) - h(x) \leq h(T_1) - h(T_2)$.

אז נקבל שמספר הסיבובים הנדרשים הוא $O(h(T_1) - h(T_2))$.

לסיכום נקבל שזמן החיפוש של המפתח y בעץ T_1 הוא $O(h(y))$.

אנו יודעים ש־ $h(y) \geq h(T_2)$, אז נקבל $O(h(T_2))$ עבור חיפוש המפתח y בעץ T_1 .

ועבור מספר הסיבובים הנדרשים לאיזון העץ החדש, נקבל $O(h(T_1) - h(T_2))$.

כך שמחיבור זמני הריצה של כלל הפעולות במקרה זה, נקבל $O(h(T_1) - h(T_2))$.

מקרה 2 $(h_1 \leq h_2)$ -

במקרה זה, נבצע אלגוריתם כמעט זהה למקרה הראשון, כאשר במקרה הנוכחי אנו מכניסים את העץ T_1 ואת המפתח x בצומת מסויים בעץ T_2 (הצומת הראשון (y) שבו מתרחש $0 \leq h(y) - h(T_1) \leq 1$ במסלול המתחיל

משורש העץ T_2 ועד לעלה השמאלי ביותר בעץ T_2).

...

סעיף ב

...

נגדיר רשימה מקושרת $LL_{\leq k}$ שתקבל מצביעים לצמתים שהמפתחות שלהם קטנים או שווים למפתח k

ורשימה מקושרת נוספת $LL_{>k}$ שתקבל מצביעים לצמתים שהמפתחות שלהם גדולים מהמפתח k .

לאחר הגדרת הרשימות המקושרות, נבצע סריקה של העץ T מהשורש אל העלה (בהתאם ל־ k שנקבל).

לכן זמן הריצה של הסריקה הוא $O(log(n))$ (גובה העץ).

במהלך הסריקה, נבדוק את המפתח בכל צומת שנעבור דרכו $\longleftarrow$

במקרה שבו המפתח של הצומת הנוכחי קטן או שווה למפתח k , נבצע הכנסה של הצומת אל $LL_{\leq k}$

ונעבור לבן הימני של הצומת הנוכחי להמשך הסריקה.

במקרה שבו המפתח של הצומת גדול מהמפתח k , נבצע הכנסה של הצומת אל $LL_{>k}$

ונעבור לבן השמאלי של הצומת הנוכחי להמשך הסריקה.

זמן ריצה של פעולות ההכנסה היא $O\left(log\left(n\right)\right)$ (מספר האיברים ברשימה אינו יכול להיות גבוה יותר מגובה העץ).
לאחר הסריקה, נרצה לבנות עץ AVL - T_1 באמצעות האיברים שברשימה המקושרת $LL_{\leq k}$
ועץ AVL - T_2 באמצעות האיברים שברשימה המקושרת $LL_{>k}$.
בניית העץ AVL - T_1 תתבצע ע"י שימוש בפונקציה שהגדרנו בסעיף א של השאלה.
תחילה נמזג את שני האיברים הראשונים ברשימה ע"י הפונקציה שהגדרנו ולאחר מכן נמזג את האיבר הבא ברשימה בעץ שנוצר וכך הלאה עד לסוף הרשימה.
בניית העץ AVL - T_2 תתבצע גם כן באמצעות הפונקציה שהגדרנו בסעיף א של השאלה.
עבור כל איבר ברשימה המקושרת, נפנה לבן הימני של האיבר ולא לאיבר עצמו שברשימה המקושרת (נרצה להרכיב עץ מאיברים בעלי מפתחות הגדולים ממש מהמפתח k).
וגם כאן, תחילה נמזג בין הבן השמאלי של האיבר הראשון ברשימה לבן השמאלי של האיבר השני במערך באמצעות הפונקציה שהגדרנו. ולאחר מכן נמזג את הבן השמאלי של האיבר הבא ברשימה המקושרת בעץ שנוצר וכך הלאה.
זמן הריצה של פעולות המיזוג היא $\sum_{i}^{j-2}log\left(\mid h_{i-1}-h_i\mid\right)=O\left(log\left(n\right)\right)$ כאשר j הוא אורך הרשימה המקושרת.
אז מסכום זמני הריצה של כלל הפעולות, נקבל שזמן הריצה הוא $O\left(log\left(n\right)\right)$.
...

סעיף ג

...
נשתמש ב- 2 עצי AVL על מנת לענות על דרישות זמני הריצה של הפונקציות.
עץ AVL - T_{black} השומר את האיברים בהם הצבע הוא שחור
ועץ AVL - T_{white} השומר את כל האיברים בהם הצבע הוא לבן.
...
פעולת $Add\left(x\right)$ -
נכניס את המפתח של איבר x $(x.key)$לעץ T_{black} במידה והאיבר הוא שחור.
אחרת, נכניס את המפתח של איבר x $(x.key)$ לעץ T_{white} .
זמן הריצה של פעולת ההכנסה עבור עץ AVL הוא $O\left(log\left(n\right)\right)$.
...
פעולת $Color\left(k\right)$ -
תחילה נחפש אחר המפתח k בעץ T_{black}.
במידה והאיבר לא נמצא בעץ T_{black}, נבצע את החיפוש בעץ T_{white} .
במידה והאיבר לא נמצא בעץ T_{white}, נחזיר $null$.
אחרת, נחזיר את הצבע המתאים לפי העץ שבו מצאנו את את המפתח k.
נקבל שזמן החיפוש בשני העצים הוא $O\left(log\left(n\right)\right)$.
...
פעולת $FlipColors\left(k\right)$ -
תחילה נשתמש בפונקציית הפיצול שהגדרנו בסעיף ב של השאלה, עבור העץ T_{black} ועבור העץ T_{white}.

ובכך נקבל 4 עצים חדשים -

1. עץ שחור המכיל מפתחות השווים או קטנים מ- k $\longleftarrow T_{black}^{\leq k}$.
2. עץ שחור המכיל מפתחות הגדולים מ- k $\longleftarrow T_{black}^{>k}$.
3. עץ לבן המכיל מפתחות השווים או קטנים מ- k $\longleftarrow T_{white}^{\leq k}$.
4. עץ לבן המכיל מפתחות הגדולים מ- k $\longleftarrow T_{white}^{>k}$.

זמן ריצה של פעולות פיצול אלו הוא $O\left(log\left(n\right)\right)$.

כעת נשתמש בפונקציית המיזוג שהגדרנו בסעיף א של השאלה.

באופן כזה, נקבל 2 עצים חדשים -

1. מיזוג של $T_{white}^{\leq k}$ ו- $T_{black}^{>k} \longleftarrow T_{black}$.
2. מיזוג של $T_{black}^{\leq k}$ ו- $T_{white}^{>k} \longleftarrow T_{white}$.

נראה שזמן הריצה של פעולות הפיצול הוא $O\left(log\left(n\right)\right)$.

נשים לב כי פעולת המיזוג מקבלת גם את הפרמטר - צומת x .

לכן, נציב את המפתח $\left\lfloor \frac{maxT_{black}+minT_{white}}{2} \right\rfloor$ עבור פונקצית המיזוג הראשונה

ואת המפתח $\left\lfloor \frac{maxT_{white}+minT_{black}}{2} \right\rfloor$ עבור פונקצית המיזוג השנייה.

לאחר המיזוגים, נמחק צמתים אלו משני העצים.

זמן הריצה עבור מחיקה הוא גם כן $O\left(log\left(n\right)\right)$.

אז מחיבור זמני ריצה של כלל הפעולות, נקבל $O\left(log\left(n\right)\right)$.

...

שאלה 4 - $B-trees$

...

פונקצית $search(current, k)$

1. נגדיר לולאה העוברת על כל הבנים של הצומת הנוכחי באמצעות המשתנה i .

1.א. נבדוק את התנאי $child[i].size-1 \geq k$.

1.א.א. התנאי מתקיים, אז נבצע קריאה רקורסיבית ל- $search(current.child[i], k)$ ונחזיר את פלט הפונקציה.

1.ב. תנאי (1.א) אינו מתקיים,

אז נבדוק את התנאים $k == size$ ו- $i < current.numberOfChildren$.

1.ב.א. התנאי מתקיים, אז נחזיר את המפתח ה- i של הצומת הנוכחי.

1.ג. תנאים (1.א) ו- (1.ב) אינם מתקיימים,

אז נבדוק את התנאים $current == leaf$ ו- $k < current.numberOfChildren$.

1.ג.א. התנאי מתקיים, אז נחזיר את המפתח ה- k של הצומת הנוכחי.

1.ד. נגדיר $k = k - size - 1$.

2. נחזיר $null$.

...

נבדוק עבור המקרה הגרוע ביותר בעץ כאשר k הוא המפתח האחרון בעלה הימני ביותר בעץ.

נראה שהמקרה הגרוע ביותר הוא כאשר גובה העץ הוא מקסימאלי ביחס ל- t הנתון.

במקרה כזה, נקבל שגובה העץ הוא $log_t\left(n\right)$ עבור n איברים בעץ.
גובה מקסימאלי בעץ $B-tree$ גורר לכך שמספר המפתחות בכל קודקוד הוא מינימאלי.
כלומר, מספר המפתחות בכל קודקוד הוא $t-1$ כך שמספר הבנים של כל קודקוד הוא t (מתכונת עץ $B-tree$).
אז בחיפוש אחר האיבר ה־ k, נצטרך לעבור על $t*h$ קודקודים לכל היותר.
כלומר, $O\left(t*h\right)=O\left(t*log_t\left(n\right)\right)=O\left(t*\frac{log(n)}{log(t)}\right)=O\left(\frac{t}{log(t)}*log\left(n\right)\right)$.
...