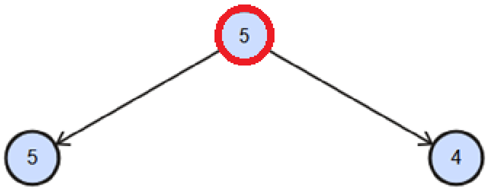
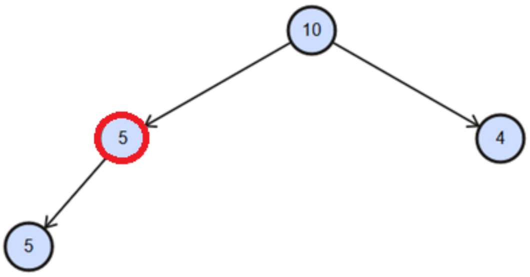
מבני נתונים ומבוא לאלגוריתמים – ממ"ן 14

שאלה 1

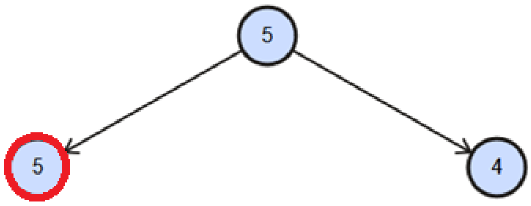
א. **ערימה:**
לא בהכרח, נציג דוגמה נגדית:
בהינתן הערימה הבאה:



נכניס 10 לערימה:



עתה נוציא את 10. לאחר מכן האיבר האחרון עובר לראש הערימה ומבצעים עליו heapify, ונקבל:



נשים לב כי עתה 5 אדום בתחתית הערימה, ואילו בהתחלה 5 שחור היה.
לכן מבנה הערימה לא נשאר אותו הדבר לאחר הכנסה והסרת 10.
יש לציין ש־5 אדום ושחור הם אולם בעלי מפתח זהה, אך שניהם שייכים לאובייקט שונה, כלומר הם לא מייצגים את אותו איבר ולכן סדר האיברים בערימה שונה.

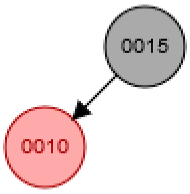
הערה: לאחר שרטוט וכתיבת הדוגמה, ראיתי שכאשר האיבר החדש שמכניסים הוא בן שמאלי ומקסימלי (כך שיהיה בראש הערימה לאחר סידורה) ואז מסירים אותו, הערמה תשתנה גם אם מסתכלים על ערכי המפתחות בלבד, ולא מתחשבים ב"אובייקט" שהם שייכים אליו. כך שאני לא רוצה שישתמע מהתשובה שלי שכל מה שיכול להשתנות זה סדר בין מפתחות זהים.

טבלת גיבוב:
המבנה יישאר זהה.
נניח שהכנסנו איבר בעל מפתח x לטבלת גיבוב. יש שתי אפשרויות:
1. התא שהאיבר נכנס אליו ריק, כלומר x יהיה האיבר הראשון ברשימה המקושרת.
לאחר שנוציא את x מהתא, הוא יחזור להיות ריק כפי שהיה קודם. אין השפעה על איברים אחרים בטבלה, ולכן הטבלה תחזור למצבה הקודם במדויק.

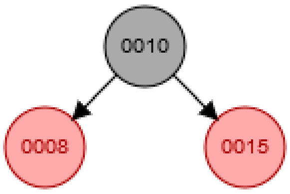
2. התא שהאיבר נכנס אליו לא ריק, כלומר קיימים איברים ברשימה המקושרת.
לפי אופן פעולת השרשור בטבלת גיבוב, האיבר x יהיה האיבר הראשון ברשימה המקושרת, והאיבר הבא יהיה האיבר שהיה ראשון לפני הכנסת x. ברגע שנמחק את x, הפעולה שתתבצע היא להגדיר את ה־head של התא להצביע ל־next(x), כלומר לאיבר שהיה ראש הרשימה לפני שהכנסנו את x. ולכן המבנה יישאר זהה.

עץ חיפוש בינארי:
המבנה יישאר זהה. כידוע, בהכנסת איבר לעץ חיפוש בינארי, האיבר החדש תמיד יהיה עלה. הסרת עלה מעץ חיפוש בינארי היא טריוויאלית: פשוט מוחקים את העלה.
מפני שלעלים אין בנים (הגדרה), העץ נשאר עץ חיפוש בינארי ולכן אין צורך לבצע פעולות נוספות.
במילים אחרות, לא מבוצעות פעולות על העץ מלבד הוספה והסרה של אותו עלה, ולכן העץ נשאר כשהיה.

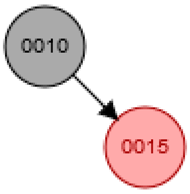
עץ אדום שחור:
המבנה לא בהכרח יישאר זהה.
נציג דוגמה נגדית:
בהינתן עא"ש הבא:



נוסיף לו את האיבר 8. הוא קטן מ־10, לכן נכניס אותו כבן שמאלי של 10.
כעת יש שני איברים אדומים ברצף. שניהם בנים שמאליים, ולכן ניתן לתקן ע"י רוטציה ימנית על 10, נקבל:



עתה נסיר את 8. 8 הוא עלה, ולכן ניתן פשוט למחוק אותו, ונישאר עם העץ הבא:



קיבלנו עץ שונה משהיה בהתחלה, כלומר מצאנו דוגמה בה הכנסה והסרה של איבר משנה את מבנה העץ.

ב. **ערימה**:
סדר ההכנסה משנה. נציג דוגמה:
נניח שהערמה שלנו בעלת איבר אחד, והוא 10.
נכניס לה את 5, אז הוא יהיה בן שמאלי של 10. נכניס את 6, הוא יהיה בן ימני של 10.
עתה נכניס לערמה קודם את 6, אז 6 יהיה בן שמאלי של 10, נכניס את 5 והוא יהיה הבן הימני הפעם.
כלומר סדר שונה של הכנסת האיברים הניבה עצים שונים.

טבלת גיבוב:

סדר ההכנסה משנה.
נניח שאנו מכניסים את x,y, ושניהם מגובבים לאותו תא בטבלה.
אם נכניס את x קודם, ואחריו את y, אז בסוף הרשימה המקושרת תהיה כך ש־y הוא הראש ואחריו x.
אם נכניס את y קודם, ואחריו את x, הפעם x יהיה ראש הרשימה ואחריו y.
לכן סדר ההכנסה משנה.

עץ חיפוש בינארי:
סדר ההכנסה משנה.
נניח שהעץ ריק, נכניס את האיבר 8. אז כעת הוא שורש העץ ואין לו בנים.
נכניס את האיבר 9, אז כאמור הכנסה של איבר לעץ חיפוש בינארי תמיד תהיה כעלה. אז 9 גדול מ־8, ולכן הוא יהיה הבן הימני של 8.
כלומר נקבל עץ ש־8 הוא השורש, ו־9 בן ימני שלו.

אם נכניס קודם את 9 ואז את 8, אז 9 יהיה השורש ו־8 יהיה הבן השמאלי שלו (שכן $8 < 9$), כלומר נקבל שני עצים שונים.

עץ אדום שחור:
סדר ההכנסה משנה.
אותה דוגמה בדיוק כמו של עץ חיפוש בינארי שלעיל, תניב את אותו עץ (רק שהצומת שהוא בן יהיה צבוע באדום).

שאלה 2

ראשית, נכניס את קבוצת k הצבעים הנתונה לטבלת גיבוב T בגודל k עם מיעון ישיר. נוכל לעשות זאת כי נוכל לייצג כל צבע במספר שונה ($1 \dots k$).
כל תא בטבלה יקבל ערך בוליאני (מאותחל ל"שקר"), כך שלכל מסלול נוכל לסמן באמצעות הטבלה את הצבעים שנמצאים בו.
כמו כן, נשים לב כי אורך מסלול מגוון הוא לכל היותר k.
נאתחל מונה count ל־0, שיספור את מספר הצמתים.

בשלב הראשון, נעבור רקורסיבית על תת העץ השמאלי בסדר **סופי**, החל מהשורש.
תחילה נבדוק את תנאיי העצירה:
אם x=nil, נחזיר 0.
אם T[key(color[x])]=true, כלומר אם הצבע של הצומת x קיים בטבלת הגיבוב, הרי שלא ניתן להתקדם לצומת הזה שכן יחד איתו המסלול הנוכחי לא מגוון, אז נחזיר 0.
אם לא (המסלול עדיין חוקי), אז נשנה את שדה הצבע בטבלה להיות true, נקדם את count באחד, ונבדוק האם count=k. אם כן, יש מסלול מגוון באורך מקסימלי ולכן ניתן להחזיר k.
אם לא, כלומר: $count \neq k$, נמשיך רקורסיבית לבניו של x באותו אופן, ונשמור את התוצאה שהם מחזירים, כלומר את אורך המסלול המגוון המקסימלי שמתחיל מהם.

כששתי הקריאות לבניו של x חוזרות, נקרא לשגרה של השלב השני של האלגוריתם, שהיא מעבר על תת העץ הימני. בקריאה נעביר מצביע לטבלת הגיבוב במצבה הנוכחי, ואת המונה count, שמכיל את אורך המסלול.
השגרה תחזיר את אורך המסלול המגוון הארוך ביותר שכולל את כל הצמתים עד כה בתת העץ השמאלי. שזה המסלול המגוון הארוך ביותר בעץ שכולל את צומת x ועובר בשורש.

כעת נסיר את צבע צומת x מהטבלה, נוריד 1 מ־count, ונחזיר לאב את הערך המקסימלי שמצאנו.
בכך בעצם אנו מסירים את הצומת הנוכחי מהמסלול.
בסוף נחזיר לקורא את אורך המסלול המגוון הארוך ביותר שמצאנו בעץ.

שלב שני (חיפוש בתת עץ הימני):
יש לנו טבלת גיבוב T ואת מספר הצמתים במסלול שהיא מייצגת.
הצומת הראשון הוא תמיד הבן הימני של שורש העץ. גם כאן אנו נעבור על תת העץ הימני בסדר סופי, והאלגוריתם דומה לשלב א':
תנאיי העצירה:
אם x=nil, נחזיר 0.

אם T[key(color[x])]=true, כלומר אם הצבע של הצומת x קיים בטבלת הגיבוב, הרי שלא ניתן להתקדם לצומת הזה שכן יחד איתו המסלול הנוכחי לא מגוון, אז נחזיר 0.
אם לא (המסלול עדיין חוקי), אז נשנה את שדה הצבע בטבלה להיות true, נקדם את count באחד, ונבדוק האם count=k. אם כן, יש מסלול מגוון באורך מקסימלי ולכן ניתן להחזיר k.
אחרת, נמשיך רקורסיבית לבניו של באותו אופן, ונשמור את התוצאה שהם מחזירים, כלומר את אורך המסלול המגוון המקסימלי שמתחיל מהם.

עד כה השגרה דומה לשגרה של תת העץ השמאלי. ההבדל הוא שכעת נחשב את הערך המקסימלי מבין: אורך המסלול הנוכחי, אורך המסלול הארוך ביותר מהבן השמאלי ואורך המסלול הארוך ביותר מהבן הימני.
נסיר את צבע צומת x מהטבלה, נוריד 1 מ־count, ונחזיר לאב את הערך המקסימלי שמצאנו.

נכונות:

נשים לב כי אנו עוברים על כל מסלול מגוון בעץ, שכן השגרה של תת העץ השמאלי מתחילה בשורש, כך שהשורש חלק מהמסלול כנדרש. לאחר מכן אנו נעבור על **כל** תת העץ השמאלי, אלא אם כן נגיע לצומת שממנו לא ניתן להגיע לשורש במסלול מגוון.

נשים לב כי לכל צומת בעץ, מפני שאין בו מעגלים (מהגדרת עצים), יש רק מסלול אחד לשורש, והוא לקרוא שוב ושוב ל־parent(x) עד שנגיע לשורש. זה בדיוק מה שאנו עושים, ולכן אנו לא מפספסים אף מסלול אפשרי שמתחיל בצומת כלשהו בתת העץ השמאלי.

בשגרה של תת העץ הימני, אנו כבר מקבלים מסלול חוקי שמכיל בתוכו את השורש, לכן אנו חופשיים לנסות כל מסלול בתת העץ עד שנגיע להפרה.

כאשר אנו חוזרים מהקריאה לתת העץ הימני, טבלת הגיבוב חוזרת לקדמותה שכן לפני חזרה מצומת אנו מסירים את הצבע מטבלת הגיבוב.

נשים לב כי במקרה שהמסלול הארוך ביותר לא מכיל אף צומת בתת העץ השמאלי, אנו עדיין נמצא אותו שכן לאחר מיצוי של האופציות של השגרה הראשונה, אנו מבצעים קריאה לתת העץ הימני רק עם השורש, כנדרש.

סיבוכיות:

הכנסה לטבלת גיבוב וחיפוש מתבצעים בזמן קבוע (שכן משתמשים במיעון ישיר).

יצירת הטבלה בעלת k איברים (כמספר הצבעים) תתבצע בזמן k.

במקרה הגרוע, $k \geq n$, וכל הצבעים של כל הצמתים שונים.

נניח שבתת העץ השמאלי יש j איברים, אז בתת עץ הימני יש $n-j$ איברים, כאשר $0 \leq j \leq n$.

אז בשלב הראשון אנו עוברים על כל האיברים בתת עץ השמאלי בסדר תוכי, ובכל צומת אנו מבצעים עבודה קבועה (מלבד הקריאה לתת עץ הימני), ולכן המעבר עצמו לוקח $\Theta(j)$ זמן.

לכל צומת, אנו קוראים לשגרה של תת העץ הימני, והיא במקרה הגרוע תעבור על כל הצמתים בו, ולכן נבצע j קריאות לתת העץ הימני, וכל קריאה תעבור במקרה הגרוע על כל $n-j$ האיברים בתת העץ הימני, ולכן נבצע $j(n-j) = jn - j{}^\wedge 2$. נשים לב שכאשר העץ מאוזן, יש כ־$\frac{n}{2}$ איברים בכל צד של העץ, ולכן נבצע $\frac{n}{2} * \frac{n}{2} = \frac{n^2}{4}$, כלומר זמן הריצה הוא ריבועי: $\Theta(n^2)$.

נוסיף לזה את זמן יצירה ואתחול טבלת הגיבוב של k הצבעים, ונקבל שזמן הריצה הכולל הוא: $\Theta(k + n^2)$.

שאלה 3

על מנת לתקן את העץ לאחר הסרת x, נבצע את התהליך הבא:
נסיר את תת העץ המושרש ב־x מהעץ.
נחבר את z לאב של y (אם אין לו אבא, כלומר שהוא השורש ואז נשים את z כשורש העץ) ונצבע אותו בשחור. כעת y מנותק מהעץ, נשמור אותו בצד (נוסיף אותו לאחר מכן בחזרה לעץ).
נשים לב כי יש לנו כעת עץ אדום שחור חוקי.
מפני שהתחלנו עם עץ אדום שחור חוקי, אז אורך כל המסלולים השחורים החל מהאיבר y זהה.
מפני ש־x,y (לפני השינוי) היו אדומים, הרי שכל תת העץ המושרש בהם מכיל את אותו מספר שחורים.
לכן כאשר הסרנו את y, שמנו במקומו את z ושינינו את צבעו לשחור, הרי ששמרנו על אותו מספר שחורים בכל המסלולים בעץ.

כל שנותר לנו כעת הוא להוסיף את y לעץ ע"י שגרת RB-INSERT (כפי שמופיעה בספר הלימוד עמ' 236). מפני שהשגרה יודעת להכניס איבר לעץ אדום שחור ולשמור על תכונותיו בסיום, הרי שהאלגוריתם עובד כרצוי.

כלומר, בתחילה הסרנו את x ו־y מהעץ אבל שמרנו על תקינותו, ולאחר מכן הוספנו את y.
נראה שהסרת x ו־y שומרת על כל תכונות עץ אדום שחור:

1. כל צומת הוא אדום או שחור (לא ביצענו צביעה אחרת).
2. השורש הוא שחור, שכן או שלא נוגעים בשורש, או ש־z הופך להיות השורש והרי שצבענו אותו בשחור.
3. כל העלים שחורים, שכן האיבר היחיד שצבענו הוא z וצבענו אותו בשחור.
4. אנו יודעים שבניו של z "תקניים", ולכן שחורים (כי z המקורי אדום), וכן שאביו שחור. אנו צבענו את z בשחור, ולכן לא יכול להיות שיצרנו רצף מסוים של שני אדומים.
5. אורך המסלולים השחורים נשאר זהה, שכאמור לעיל מחקנו צומת שחור (y) ובתמורה צבענו את z בשחור, ולכן סך הצמתים השחורים נשאר זהה.

הוכחנו כי האלגוריתם פועל נכון.

סיבוכיות:
הסרת x,y והצבת z כבן של אבא של y (או שורש אם לא קיים) מתבצעת בזמן קבוע.
הכנסת y לעץ ע"י שגרת RB-INSERT מתבצעת כידוע בזמן $\Theta(\lg n)$, ולכן סך הסיבוכיות היא: $\Theta(\lg n)$.

שאלה 4

נשתמש בעץ אדום שחור מורחב T (כפי שמופיע בספר הלימוד), ובנוסף לשדה size, נוסיף לכל צומת שדה sum שיכיל את סכום כל האיברים בתת העץ שלו (כולל).
כמו כן, נשתמש בעץ אדום שחור מורחב **נוסף** QT, כמו שמופיע בספר (עץ ערכי מיקום), כלומר לכל צומת יש שדה size שמכיל את מספר הצמתים המורשים בו (כולל). בעץ זה, המפתחות יהיו זמן ההכנסה, אבל נשמור בשדה נוסף (val) את המפתח k.
כמו כן, נשמור מצביע בעץ Q לצומת המתאים בעץ QT.

נשים לב כי תחזוקת השדות הנוספים מחושבת רק באמצעות צומת ושני בניו, ואז לפי משפט 14.1 ניתן לתחזק את השדות הנ"ל מבלי להשפיע אסימפטוטית על זמן הריצה.

INSERT:

הכנסה של המפתח k רגילה לעץ T, ועדכון השדה sum, שמכיל את סכום שדות sum של בניו והערך k.
הכנסת הזמן לעץ QT, עדכון השדה val=k ושדה size כפי שמופיע בספר הלימוד.
יצירת מצביע בצומת בעץ T לצומת בעץ QT.
הכנסה לעץ אדום שחור היא $\Theta(\lg n)$ וכאמור ממשפט 14.1 עדכון השדות לא משפיע על זמן הריצה.
כמו כן, יצירת המצביע לוקחת זמן קבוע.
לכן הסיבוכיות: $\lg n + \lg n + 1 = \Theta(\lg n)$.

DELETE:

מחיקה מעץ אדום שחור רגילה. באמצעות המצביע בעץ T מוחקים את הצומת המתאים ב־QT, מוחקים את הצומת ב־T, ומעדכנים את השדות הנוספים (שכאמור ממשפט 14.1 לא משפיעים על זמן הריצה).
מחיקה מעץ אדום שחור מתבצעת כידוע ב־$\Theta(\lg n)$, לכן הסיבוכיות: $\lg n + \lg n = \Theta(\lg n)$

PAIR-SUM:

נשתמש בעץ T. נמצא ונשמור מצביעים לאיבר המינימלי והאיבר המקסימלי בעץ (העלה השמאלי ביותר והעלה הימני ביותר בהתאמה). כל פעולה כזו לוקחת $\Theta(\lg n)$ ולכן סה"כ $\lg n + \lg n = \Theta(\lg n)$.
נסמן את המצביעים $left, right$, המפתחות בשדה value.
כל עוד $left \neq right$, נבצע: (חשוב: הכוונה שהמצביעים לא מצביעים לאותו איבר, לא שערך המפתחות לא שווה)
אם $left.value + right.value > s$ אז: $right = pred[right]$ (כלומר האיבר הקודם ל־right).
אחרת אם $left.value + right.value < s$ אז: $left = succ[left]$ (כלומר האיבר העוקב ל־left).
אחרת ($left.value + right.value = s$) החזר את $left, rigt$, כלומר מצאנו זוג איברים שעונה על הדרישה.

בדרך זו נעבור על כל האיברים בעץ, באמצעות השגרות של מציאת איבר עוקב וקודם כפי שמופיעות בספר הלימוד.
נשים לב כי במקרה הגרוע נעבור על כל איברי העץ ומפני ששגרות עוקב/קודם מתבצעות ב־$\Theta(\lg n)$, נראה כי זמן הריצה הוא $\Theta(n \lg n)$ במקרה הגרוע, אך לא כך הדבר: לפי שאלה 12.2-7 זמן הריצה הוא $\Theta(n)$ במקרה הגרוע, בהתאם לנדרש בשאלה.

AVE:

נמצא את האיבר במיקום i_1 בעץ T ע"י $OS - SELECT(root[T], i_1)$, כפי שמופיעה בספר הלימוד בעמוד 254. נבצע פעולה דומה עבור האיבר במיקום i_2.
נסמן את האיברים l, r.
אנו יודעים שהערך $root[T].sum$ מכיל את סכום כל האיברים בעץ T.
אנו רוצים להחסיר מסכום זה את כל האיברים הקטנים מהאיבר left, וכמו כן להחסיר את כל האיברים הגדולים מהאיבר right. לכן $sum = root[T].sum - (left[l].sum + right[r].sum)$ זה בעצם הסכום של כל האיברים בין l, r, כפי שאנחנו רוצים.
לחישוב הממוצע, כל שעלינו לעשות הוא לחלק את הסכום במספר האיברים: $\frac{sum}{i_2 - i_1 + 1}$, וזה הממוצע שמבוקש.
נשים לב כי כל החישובים מתבצעים בזמן קבוע, ועיקר העבודה הוא במציאת הצמתים l, r.
כפי שמוכח בספר הלימוד, הסיבוכיות של $OS - SELECT$ היא $\Theta(\lg n)$, אנו משתמשים בה פעמיים, ולכן סך הסיבוכיות: $\lg n + \lg n = \Theta(\lg n)$.

OLD:

נקרא לשגרה $OS - SELECT(root[QT], m)$, זמן הריצה הוא $\Theta(\lg n)$ כנדרש. אנו שומרים בעץ QT את המפתחות לפי זמן ההכנסה, ולכן סדר האיברים בעץ QT הוא בעצם סדר ההכנסה של המפתחות לעץ T, וזה מה שאנחנו רוצים למצוא.

שאלה 5

נשתמש בעץ אדום שחור כמבנה הנתונים שלנו, כאשר המפתח יהיה זמן הרשמה לקורס (t).
כל רשומה תכיל את השדות הבאים:
שם סטודנט, זמן (t), מצב לימודים (status), היכול להכיל את הערכים 1/0 (תקין/לא תקין).

כמו כן, נוסיף לכל איבר x שדה: max-valid-time, היכיל את הזמן המקסימלי שסטודנט נרשם בתת העץ המושרש ב־x (כולל x), כאשר נציב בשדה 0 אם לא קיימת תמונה כזו.
נעדכן את השדה הנ"ל ע"י חישוב:

$$\begin{cases} m[x] = \max\left(m\left[left[x]\right],\ m[right[x],\ t[x]\right) & status[x] = 1 \\ m[x] = \max\left(m\left[left[x]\right],\ m[right[x]]\right) & status[x] = 0 \end{cases}$$

(קיצרתי את שם השדה החדש ל־m בחישוב לצורך בהירות)
כלומר, ערך השדה יהיה התאריך המקסימלי מבין: השדה x (אם מצב הלימודים שלו תקין), שדה m של הבן השמאלי של x או שדה m של הבן הימני של x.
לפי משפט 14.1, מפני שערך השדה מחושב רק באמצעותו ובאמצעות בניו הישירים, הרי שניתן לתחזק את שדה זה בהכנסה ומחיקה מבלי להשפיע אסימפטוטית על זמן הריצה.
כלומר, אם מכניסים או מוחקים איבר שסטטוס הלימוד שבו תקין, יש לעדכן את השדה הנ"ל בהתאם ולפי הצורך.

הכנסת סטודנט:
הכנסה רגילה לעץ אדום שחור, עם עדכון השדה max-valid-time בהתאם, שכידוע מתבצעת בזמן $\Theta(\lg n)$.

מחיקת סטודנט:
מחיקה רגילה מעץ אדום שחור, עם עדכון השדה max-valid-time בהתאם, שכידוע מתבצעת בזמן $\Theta(\lg n)$.

סטודנט הבא אחרי x:
נשתמש בשגרה successor בעץ חיפוש בינארי (ובפרט בעא"ש). מפני שהמפתחות הם זמן הרישום, מציאת האיבר הבא אחרי סטודנט x יביא את מועד ההרשמה העוקב, כנדרש.
כידוע, זמן ריצת השגרה $\Theta(\lg n)$.

הרשמה תקינה הבאה אחרי זמן t:
תחילה נבדוק האם: $\max - \text{valid} - \text{time}[root[T]] > t$. אם לא, אזי שלא קיים איבר כזה בעץ, ונחזיר NULL.

אחרת, נבצע קריאה לשורש העץ x עם הערך t.
אם יש ל־x בן שמאלי ו: $\max - \text{valid} - \text{time}[left[x]] > t$, הרי שהאיבר המבוקש נמצא בתת העץ השמאלי של x, אז נבצע קריאה רקורסיבית לבן השמאלי של x.

אחרת, נבדוק האם $status[x] = 1$ וגם $t[x] > t$. אם כן, הרי ש־x הוא איבר המבוקש.
אחרת, האיבר נמצא בתת העץ הימני של x, כך שנבצע קריאה רקורסיבית לבן הימני של x.

בכל קריאה אנו מבצעים עבודה בזמן קבוע ולכל היותר יורדים רמה אחת בעץ, ומפני שגובה עא"ש הוא $\Theta(\lg n)$, הרי שהסיבוכיות של השגרה היא $\Theta(\lg n)$.