מבני נתונים - תרגיל רטוב 1

21 בדצמבר 2020

מבני הנתונים וטיפוסים בהם השתמשנו

$AVL\_Tree < KEY, VAL >$ - עץ AVL :

גנרי דו כיווני (הורה מצביע לילד וילד מצביע להורה) כפי שנלמד בהרצאה.
הצמתים ממוינים לפי ערך המפתח ומכילים את המידע .
הפעולות המוגדרות וסיבוכיות הזמן והמקום של המבנה כפי שנלמדו בכיתה.

$List < T >$ - רשימה מקושרת דו כיוונית.

אברי הרשימה הם טיפוס $ListNode < T >$, המכיל פוינטר חכם ל ערך T ,מצביע לאיבר לפני ברשימה ומצביע לאיבר אחרי ברשימה.
הפעולות המוגדרות וסיבוכיות הזמן והמקום של המבנה כפי שנלמדו בכיתה.

$Lecture$ -אוביקט המייצג הרצאה:

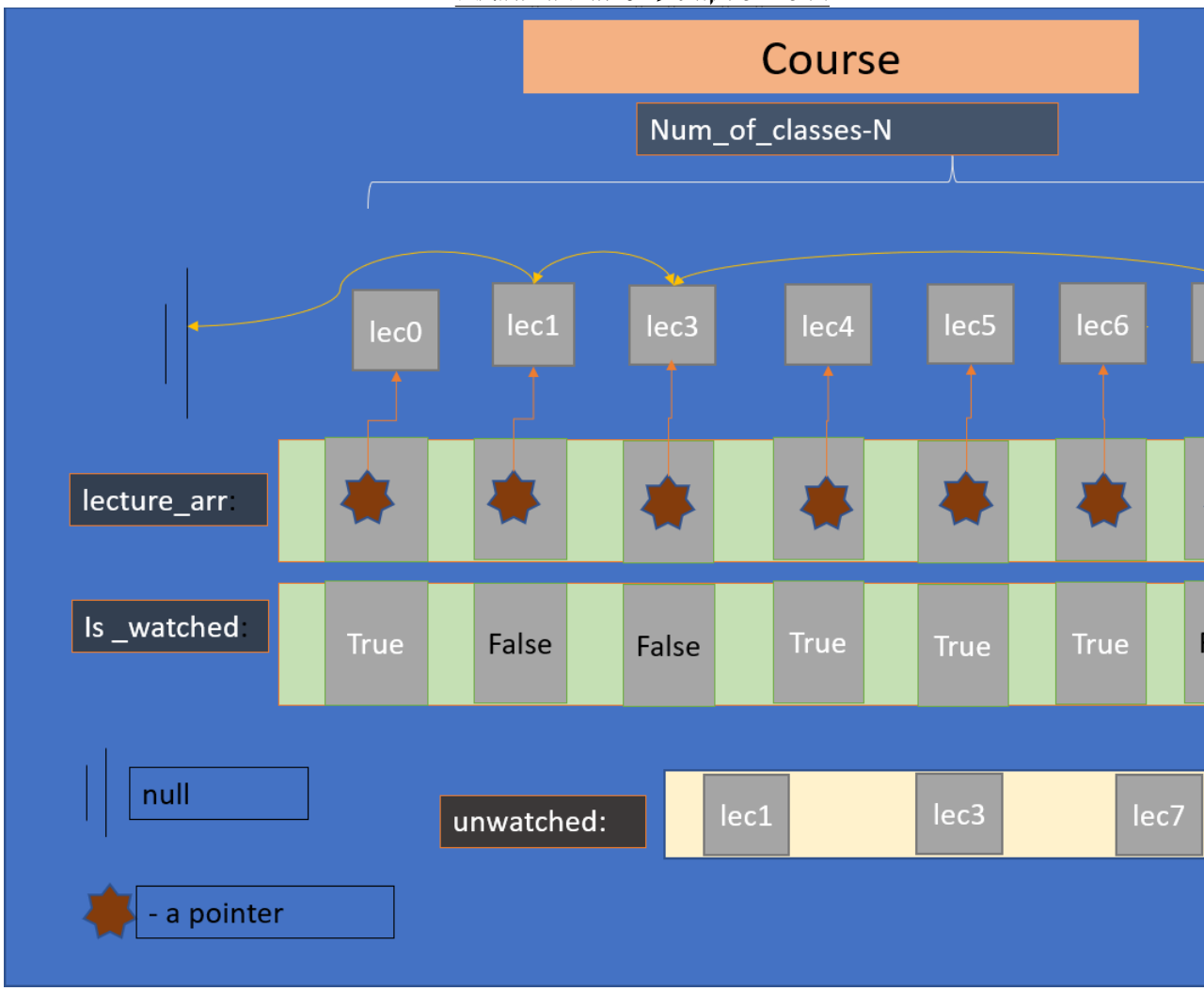
שדות:
$courseID$ -מספר טבעי חיובי (int) המייצג את מזהה הקורס שאליו שייכת ההרצאה
$lectureID$ - מספר טבעי אי שלילי (int) המייצג את מזהה ההרצאה
num_views - מספר טבעי אי שלילי המייצגג כמה צפיות יש להרצאה
לאובייקט הנ"ל מוגדר יחס סדר בעזרת אופרטורי השוואה(גדול , קטן , שווה) המוגדר באופן הבא:
1) לפי num_views - מספר צפיות בסדר ירוד (הרצאה l_1 בעלת מספר צפיות גדול יותר מאלה של l_2 תקיים $l_1 > l_2$).
אם מתקיים שוויון במספר הצפיות:
2) לפי $courseID$ - מזהה הקורס בסדר עולה (בהינתן שוויון במספר הצפיות ,הרצאה l_1 בעלת מזהה קורס קטן מזה של l_2 תקיים $l_1 > l_2$).
אם מתקיים שוויון גם במזהה הקורס:
3) לפי $lectureID$ - מזהה ההרצאה בסדר עולה.
במידה ומתקיים שוויון בכל שלושת השדות, נגיד ש l_1 שווה ל l_2.
פעולות מוגדרות :
אתחול: אתחול שדות המבנה.
השמת מספר קבוע של ערכים מסוג int .
הוספת זמן צפיה: עדכון שדה זמן הצפיה לפי פרמטר.
עדכון שדה מסוג int.

שליפת זמן צפיה: החזרת שדה זמן הצפיה.
קריאה של שדה מסוג int .
אופרטורי השוואה: השוואה בין הרצאות כפי שתוארה לעיל.
השוואה של intים .
סיבוכיות זמן של כלל הפעולות:
כל הפעולות הן בסיבוכיות זמן ומקום $O(1)$ (פעולות על מספר קבוע של intים).
סיבוכיות מקום:
$O(1)$, השדות של $Lecture$ הן מספר קבוע של קבועים מסוג int.
הפעולות לא מקצות זכרון חדש ולכן גם הן $O(1)$ כל אחת.

$Course$ - אוביקט המייצג קורס:

מטרה:
מטרת המבנה היא להכיל את המידע הרלוונטי לקורס מסוים והרצאותיו.
שדות:
$courseID$ - מספר טבעי חיובי (int) המייצג את מזהה הקורס.
$lecture\_arr$ - מערך של מצביעים ל$ListNode < Lecture >$. מערך זה כולל מצביעים לחוליות ברשימה של כל ההרצאות של הקורס .
$num\_of\_classes$- מספר טבעי חיובי המייצג את מספר ההרצאות שיש בקורס (האורך של $lecture\_arr$).
$unwatched$ - רשימה של עצמים מטיפוס $Lecture$. רשימה זו כוללת את כל ההרצאות של הקורס להן 0 צפיות ורק אותן.
מטרת השדה היא לאפשר מעבר רק על הרצאות שלא נצפו בלי לבצע פעולות על איברי ביניים.
$is\_watched$ - מערך של $bool$ באורך זהה לשל $lecture\_arr$ ועם אינדקסים תואמים - הערך באינקדס i הוא $true$ אם ההרצאה הi כבר נצפתה, ו$false$ אחרת.
מטרת השדה היא להוות מעקב נפרד אחרי ההרצאות שכבר נצפו . כך מתאפשר שחזור נכון של $unwatched$ בהעתקה או השמה של קורסים, ומוודא שבעת הריסת אובייקט $Course$ שחרור הזיכרון של $unwatched$ ו$lecture_arr$ לא יוביל לשחרור כפול של חוליות ברשימה.
פעולות מוגדרות:
אתחול (מפרמטרים או כהעתקה עמוקה):
אתחול השדות , חלק מהשדות הם מערכים ורשימות, כשכל רשימה היא בגודל לכל היותר m לכן האתחול נעשה ב$O(m)$ (כאשר מעתה $m = \text{num\_of\_classes}$, מספר ההרצאות בקורס).
אופרטור השוואה:
העתקה עמוקה של השדות, בדומה לאתחול סיבוכיות זמן היא $O(m)$.
שליפת שדות:($Getters$):
נעשה ב$O(1)$, השדות הפשוטים מועברים ב$O(1)$ וכך גם המורכבים(מערכים ורשימות) כיוון שמעבירים רפרנס או מצביע לשדה.
הריסה:
מעבר על מספר ההרצאות בקורס והריסתן אחת אחת $O(m)$.
סיבוכיות מקום:
גודל הרשימות והמערכים שזכרונם מנוהל על ידי קורס:
$|\text{lecture\_arr}| = m$, $|\text{unwatched}| \leq m$,$|\text{is\_watched}| \leq m$
שאר השדות הן $O(1)$.
מכאן ש סיבוכיות המקום של $Course$ היא $O(m)$.
נציין כי הפונקציות של הפעולות לא רקורסיביות ולא מקצות זכרון חדש ולכן סיבוכיות המקום שלהן היא $O(1)$.

תרשים של קורס עם 10הרצאות לדוגמא:



אוביקט $BoomDS$ ולו השדות הבאים:

$courses$- עץ AVL של $Course$, ממוין לפי מזהה הקורס.
$lectures$- עץ AVL של $Lecture$, ממוין לפי אופרטורי ההשוואה של מחלקת $Lecture$.
כלומר, מספר צפיות בסדר יורד, לאחר מכן מזהה קורס בסדר עולה, ולאחר מכן מזהה הרצאה בסדר עולה.

עץ זה מכיל רק את ההרצאות שלהן יותר מ 0 צפיות.
$most\_watched$ - מצביע לAVL_NODE מהעץ $lectures$ המהווה את הצומת הימני ביותר בעץ - כלומר את ההרצאה ה"גדולה" ביותר והראשונה בסדר ההדפסה.
$smallest\_id$ - מצביע לAVL_NODE מהעץ $courses$ המהווה את הצומת השמאלי ביותר בעץ - כלומר את הקורס ה"גדול" ביותר והראשון בסדר ההדפסה.

פעולות מוגדרות:

- $AddCourse$

מבצעת הוספה של קורס חדש לעץ $courses$, ומעדכנת את $smallest\_id$ לפי הצורך.

יצירת הקורס היא בסיבוכיות זמן $O(m)$ (אתחול קורס), כאשר m הוא מספר ההרצאות בקורס שמתווסף.

סיבוכיות הזמן של ההכנסת איבר חדש לעץ AVL היא $O(logn)$, כאשר n הוא מספר הקורסים בעץ בעת ההוספה.

לכן סה"כ סיבוכיות זמן היא $O(log(n)+m)$.

-$RemoveCourse$

מבצעת הסרה של קורס מהעץ $courses$, ושל כל ההרצאות בעץ $lectures$ ששייכות לקורס הזה. הסרה של הרצאה מהעץ היא בסיבוכיות זמן $O(logM)$ כאשר M הוא מספר ההרצאות על פני כל הקורסים בעת ההוספה.

עבור לא יותר מm הרצאות שמוסרות מהעץ, כאשר m הוא מספר ההרצאות בקורס המוסר, נקבל כי סיבוכיות הזמן של הסרת כל ההרצאות היא $O(mlogM)$. בנוסף, מתבצעת הסרה של הקורס מעץ הקורסים, סיבוכיות הזמן לכך היא $O(log(n))$ כאשר n הוא מספר הקורסים בעץ. כיוון שניתן להניח כי יש יותר הרצאות מקורסים, ההסרה של הקורס מהעץ $courses$ לא משפיעה על סיבוכיות הזמן.

לכן סיבוכיות הזמן של הפעולה היא $O(mlogM)$.

-$WatchClass$

מוסיף זמן צפייה להרצאה מסויימת. עדכון מיקום ההרצאה בעץ $lectures$ מתבצע על ידי הסרת ההרצאה אם היא כבר בעץ, הוספת הזמן הנדרש, והוספת ההרצאה מחדש לעץ עם הזמן המעודכן. הפעולות הכבדות הן הוצאה והכנסה מעץ AVL ולכן סיבוכיות הזמן של חלק זה היא $O(logM)$ כפי שנלמד.

בנוסף, הפונקציה דואגת לשמורה של $Course$ שהרצאה שלו ניצפת : מנתקת איבר ברשימת ההרצאות של הקורס מהחוליות, כדי שבמעבר על הרצאות שלא נצפו עד כה נעבור **רק** על הרצאות שלא נצפו. לשם כך נקראת הפעולה $removeFromUnwatched$ שמנתקת איבר מהרשימה $unwatched$ בשדה של הקורס ומשאירה אותו כ"צף"(ניתוק איבר מרשימה $O(1)$), בנוסף מכתיבה במערך ה$is_watched$ שההרצאה נצפתה (שינוי ערך במערך $O(1)$). סה"כ הדגאה לשמורה היא $O(1)$.

לכן הפעולה בסך הכל מתבצעת בסיבוכיות זמן של $O(logM)$.

ולכן גם $O(logM+t)$ כאשר t הוא מספר דקות הצפייה שהתווספו.

- $TimeViewed$

מתבצע חיפוש על עץ ההרצאות, עץ AVL שגודלו n כמספר ההרצאות ולכן החיפוש נעשה ב$O(logn)$, לאחר מכן מתבצעת פעולת גישה לשדה של ההרצאה ב$O(1)$.

לכן סיבוכיות הפעולה היא $O(logn)$.

-$GetMostViewedClasses$

תחילה, מתבצע סיור על העץ של ההרצאות שצפו בהן, סיור המתחיל מ $most\_watched$ בפעולות עוקב בדומה לסיור $inOrder$ הפוך - ראשית מתבצעת קריאה רקורסיבית עבור הבן הימני, לאחר מכן נכתב במערכים הנתונים המידע עבור הצומת הנוכחי, לאחר מכן מתבצעת קריאה רקורסיבית עבור הבן השמאלי ולבסוף עבור האב.

הפונקציה מקבלת ערכים בוליאנים עבור כל אחת מהקריאות האפשריות כדי לסמן לכל איטרציה האם היא נקראה על ידי בן או אב של צומת שכבר עברנו עליו ברקורסיה, על מנת למנוע מעבר כפול על צמתים.

כאשר מגיעים לעלה השמאלי ביותר של עץ ההרצאות, במידה ונותרו עוד הרצאות להדפסה מתוך הm :

מתחיל סיור על עץ הקורסים באופן דומה לסיור על עץ ההרצאות, כך שמתחילים מ$smallest_id$.
עם זאת, כיוון שכעת הסיור הוא מהצומת הקטן לגדול, תתבצע קודם קריאה רקורסיבית עבור הבן השמאלי. עבור הצומת הנוכחי,

עוברים על כל ההרצאות שלא נצפו עד כה בקורס זה (לשם כך השדה $unwatched$ ב$Course$, במהלך התוכנית אנו דואגים לשמור על רשימה זו כך שתכיל רק הרצאות שעוד לא נצפו) . לאחר תתבצע קריאה רקורסיבית עבור הבן השמאלי ולבסוף עבור האב, תוך שימוש בערכים בוליאנים כדי למנוע מעבר כפול על צמתים כמו בסיור על עץ ההרצאות.

הסיור עובר רק על הרצאות לפי הסדר שהוגדר להדפסה ורק על הרצאות שמוגדרות להדפסה (עד כדי קבועים בסיור ה$inOrder$ ההפוך) ולכן סיבוכיות היא $O(m)$ במקרה הגרוע.

- $Quit$

שחרור הזכרון מהעצים ואיפוס הפוינטרים.

שחרור עץ נעשה באמצעות סיור $PostOrder$ רקורסיבי על העצים.

עץ ההרצאות משוחרר ב$O(n)$ כשn מספר ההרצאות , כי כל איבר בעץ משתחרר ב $O(1)$.

עץ הקורסים משוחרר גם הוא ב $O(n)$, נוכיח זאת :

נסמן את מספר ההרצאות בקורס ה i ב n_i .

מתקיים : $\sum_{i=1}^{n} n_i = n$, שכן סך ההרצאות בכל הקורסים הוא סך ההרצאות.

שחרור צומת של הקורס ה i בעץ הקורסים היא $O(n_i)$ (שחרור של קורס נעשה ב$O(n_i)$) .

לכן שחרור סך הצמית הוא :

$$\sum_{i=1}^{n} O(n_i) = O(\sum_{i=1}^{n} n_i) = O(n)$$

קבלנו אזי ששחרור מבנה הנתונים הוא $O(n) + O(n) = O(n)$ ולכן גם $O(n+m)$ כיוון שידוע ש $n \geq m$.

<u>סיבוכיות מקום:</u>

המבנה מנהל זכרון של שני עצים, עץ ההרצאות ועץ הקורסים.

הזכרון הדרוש עבור עץ ההרצאות הוא $O(n)$ כאשר n הוא מספר ההרצאות.

הזכרון הדרוש עבור עץ הקורסים הוא $O(n)$ גם כן, משיקולים זהים לשיקולי סיבוכיות השחרור של העץ שצוינו לעיל .

($\sum_{i=1}^{n} O(n_i) = O(\sum_{i=1}^{n} n_i) = O(n)$).

ישנן פעולות של המבנה שהן רקורסיביות ולכן דורשות זכרון נוסף, אך עומק הרקורסיות תמיד קטן מ $O(n)$.

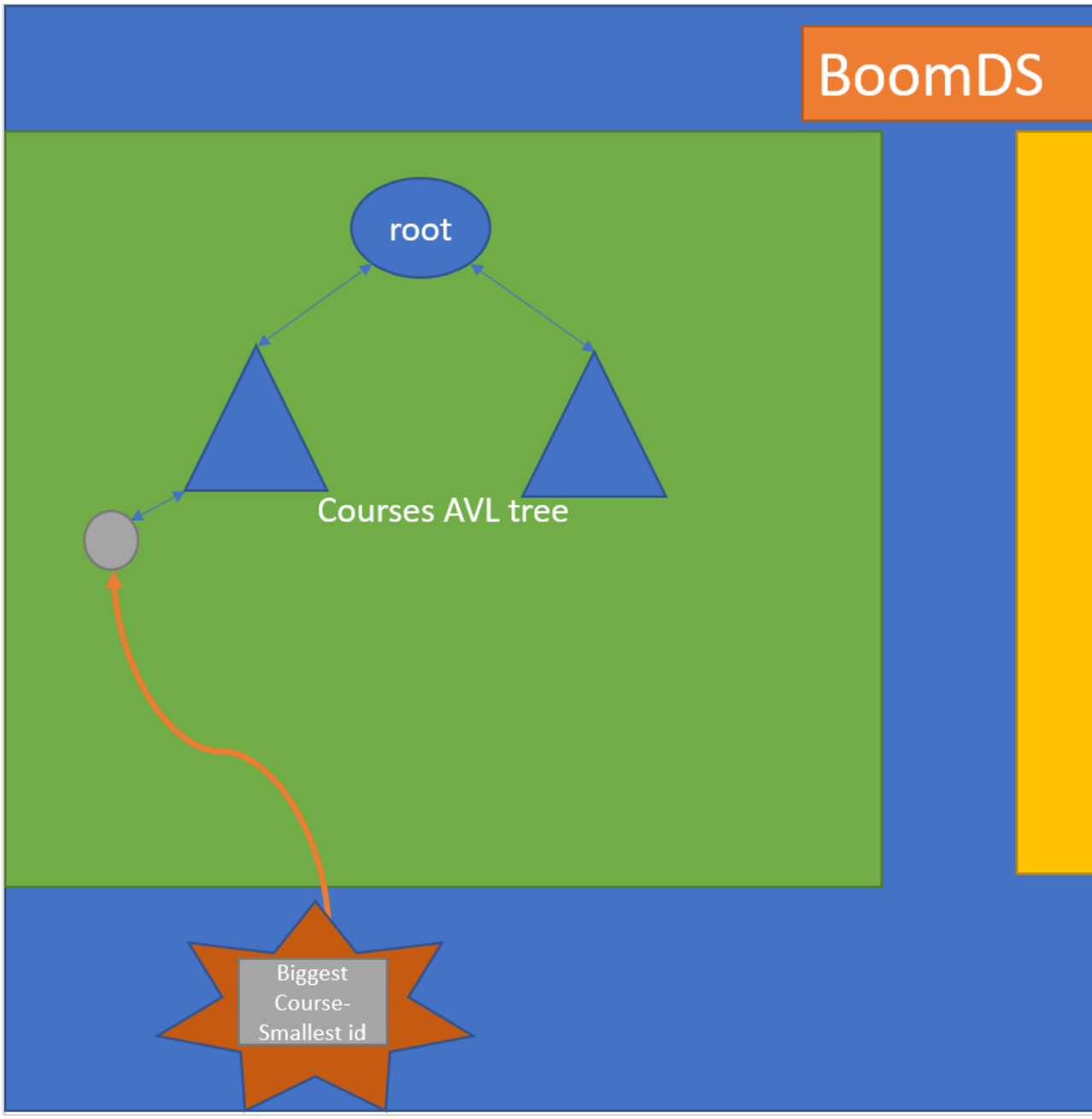
הפונקציות שפועלות באופן רקורסיבי :

חיפוש בעץ AVL, בין היתר לצורכי הוספה והסרה של איברים : הפונקציה נקראת עבור כל איבר במסלול החיפוש של האיבר המבוקש. במקרה הגרוע האיבר המבוקש לא נמצא בעץ ולכן עומק הרקורסיה יהיה כעומק העץ. כידוע עץ AVL מאוזן ולכן עומק העץ הוא $O(log(n))$.

$GetMostViewedClasses$: כולל רקורסיבי על עץ ההרצאות ועל עץ הקורסים כפי שתואר למעלה. בכל איטרציה יש 3 קריאות רקורסיביות אפשריות שכולן באותו העומק. עבור עץ ההרצאות, יש לכל היותר n איברים לעבור עליהם.

במקרה הגרוע ביותר כל אחד מהאיברים דורש הגדלת עומק הרקורסיה, אבל כיוון שיש לכל היותר n איברים עומק הרקורסיה יהיה לכל היותר n כלומר $O(n)$. עבור עץ הקורסים השיקולים זהים. ניתן להניח כי מספר הקורסים קטן מממספר ההרצאות הכולל ולכן גם עומק הרקורסיה של המעבר על עץ הקורסים הוא $O(n)$.

<u>תרשים של $BoomDS$ להמחשה:</u>


BoomDS
root
Courses AVL tree
Biggest
Course-
Smallest id