על מנת לענות על דרישות הסיבוכיות, נשתמש בשני עצי AVL – אחד עבור הקורסים, ואחד עבור **כל** השיעורים של כלל הקורסים.

על מנת להשתמש בעצי AVL כשבשני העצים nodes מסוגים שונים, מימשנו מחלקת AVLTree גנרית לפי הדגשים שנלמדו בהרצאה ובתרגול.

לכל אורך התרגיל, נתייחס למס' הקורסים הכולל כ-n, ולמס' השיעורים **הכולל** במבנה כ-m. סה"כ מדובר בשני עצי AVL, כאשר גודל עץ הקורסים הוא n וגודל עץ השיעורים הוא m. סה"כ סיבוכיות מקום O(n+m) (בינתיים).

-course\_id

-zero\_watches

-classes\_pointers\_array[][]

*בעץ הקורסים*, האיברים יהיו מטיפוס שייקרא course\_node:

* מזהה הקורס הייחודי (int course\_id)
* רשימה מקושרת של השיעורים חסרי הצפיות בקורס (twList zero\_watches)
* מערך באורך מס' השיעורים **של הקורס**.

נסמן את אורכו ב-Ci לקורס i, עבור 1 ≤ i ≤ n.מתקיים כי 1 ≤ Ci ≤ m, ולכל n > 1 מתקיים 1 ≤ Ci < m.

* מערך באורך Ci של מצביעים לNodes- של עץ השיעורים – בתא ה-i יהיה מצביע לnode של השיעור שה-id שלו הוא i.

-class\_id

-course\_id

-\*parent

-time

-classes\_pointers\_array[][]

*בעץ השיעורים*, האיברים יהיו מטיפוס שייקרא class\_node:

}

* מזהה הקורס אליו שייך השיעור נשים לב כי אף אחד משני השדות אלה אינו ייחודי לשיעור מסויים,
* מזהה השיעור בקורס אליו הוא שייך אך יחד הזוג הסדור (course\_id, class\_id) הוא ייחודי לכל שיעור.
* מצביע לאיבר האב בעץ
* זמן בדק' של צפייה בשיעור

**מימוש הפונקציות:**

***: void\* Init()***

הפונקציה קוראת לבנאי של מבנה AVLTree גנרי שיצרנו על מנת ליצור שני עצי AVL ריקים, אחד שהnodes שלו הם מסוג courseNode, והשני מסוג classNode.

סיבוכיות זמן O(1).

***StatusType AddCourse (void \*DS, int courseID, int numOfClasses):***

הפונקציה קוראת לבנאי של courseNode (בהנחה שהפרמטרים תקינים, כלומר המבנה תקין, הקורס עוד לא קיים ומס' השיעורים חיובי).

תיווצר רשימה מקושרת דו כיוונית של מספרים שלמים מ-0 עד numOfClasses שבה יהיו כל השיעורים שבהם לא צפו עדיין. סיבוכיות זמן יצירת הרשימה היא ליניארית כמספר ה-nodes, כלומר O(numOfClasses).

בכל יצירת קורס מוקצה מערך בגודל numOfClasses של מצביעים לאיברים מסוג classNode, כאשר באינדקס ה-i יופיעו בהתאמה מצביע לשיעור ה-i, וכן מצביע לאיבר ה-i ברשימה המקושרת.

המערך יאותחל בלולאה, ולכן סיבוכיות זמן האיתחול היא ליניארית, כלומר O(numOfClasses).

לאחר שנוצר הקורס, הוא יוכנס לעץ הקורסים הממויין (תוך שמירה על המיון), ועל כן סיבוכיות זמן ההכנסה היא כפי שנלמד בהרצאה O(log(m)).

לפיכך, סך כל סיבוכיות הזמן של הפונקציה היא O(logn)+2O(numOfClasses)=O(logn+numOfClasses), כנדרש.

***StatusType​ RemoveCourse(​void \*D​S, ​int​ courseID) :***

הפונקציה מבצעת חיפוש בעץ הקורסים לפי courseID סיבוכיות הזמן של פעולת החיפוש היא log(n), כאשר n הוא מספר הקורסים במערכת. במידה ונמצא הקורס המבוקש, ניגש לכל אחד מהclassNodes שבמערך המצביעים של הקורס, נוציא אותו מהעץ ונמחק אותו. כל הוצאה מתבצעת ב-O(log(M)), כאשר M הוא מספר השיעורים בכל המבנה, ומחיקה מתבצעת ב-O(1).

סיבוכיות זמן: חיפוש log(n), סה"כ m מחיקות, כאשר m הוא מספר השיעורים של הקורס אותו רוצים למחוק, כשכל אחת עולה O(log(m), לכן סה"כ O(m·log(M)).

O(log(n)+O(m·log(M)=O(log(n)+m·log(M))O(m·log(M))

***StatusType​ WatchClass(​void \*​DS, i​nt​ courseID, i​nt​ classID, ​int ​time) :***

הפונקציה מבצעת חיפוש בעץ הקורסים (log(n) כאשר n הוא מספר הקורסים הכולל שבמבנה). אם נמצא הקורס המבוקש, פונים למצביע שנמצא בתא ה-classID במערך המצביעים, וקוראים לפונקציית עזר שבנינו בשם replaceClass:

הפונקציה מקבלת מצביע לclassNode אותו אנחנו רוצים להעתיק, ויוצרת העתק שלו הנבדל ממנו בזמן בלבד – הזמן בclassNode החדש הוא סכום הזמן של הclassNode המקורי והזמן אותו נתבקשנו להוסיף (time). לאחר יצירת ה-classNode החדש, הפונקציה מוחקת מהעץ את הclassNode המקורי ומכניסה את הclassNode החדש. חיפוש בעץ, מחיקה מהעץ והוספת איבר לעץ הן כולן פעולות שסיבוכיות הזמן שלהן O(log(n)) כאשר n הוא מספר האיברים בעץ (במקרה שלנו, המספר הכולל של השיעורים שקיימים במבנה).

בנוסף, לאחר שזמן הצפייה בשיעור גדול מ-0, כלומר לאחר בצפייה הראשונה, יימחק הnode של השיעור מהרשימה המקושרת. מכיוון ששמרנו מצביע לכל שיעור במערך, סיבוכיות זמן הפעולה היא O(1).

לפיכך, סך כל סיבוכיות הזמן של הפעולה היא O(3log(n))+O(1)=O(log(n)), כנדרש.

***StatusType​ TimeViewed(​void \*​DS, ​int​ courseID, ​int​ classID, ​int \*​timeViewed) :***

הפונקציה מבצעת חיפוש בעץ הקורסים (log(n) כאשר n הוא מספר הקורסים הכולל שבמבנה). אם נמצא הקורס המבוקש, פונים למצביע שנמצא בתא ה-classID במערך המצביעים, ושולפים את time.

סיבוכיות זמן: חיפוש הקורס המבוקש - log(n).

***StatusType​ GetMostViewedClasses(​void \*​DS, ​int ​numOfClasses, ​int \*​courses, ​int \*​classes) :***

לשם מימוש הפונקציה, נתחיל בהסבר על פונקציה חדשה שיצרנו הנקראת ReversedInOrder. הפונקציה מבצעת בעץ השיעורים חיפוש חדש שיצרנו הנקרא reversedInOrder, שהוא חיפוש בסדר בדיוק הפוך מסדר InOrder, כלומר חיפוש **בסדר יורד** של ערכים. למרות שמה, הפונקציה הזו אינה פונקציית חיפוש פשוטה.חתימתה:

int reverseInOrder(int m, void (\*function)(avlNode<T> \*)) const

כאשר m הוא מספר הצמתים עליהם נרצה להפעיל את function בסדר יורד.

ערך ההחזרה של reversedInOrder הוא מספר שלם המייצג את ההפרש שבין מספר הצמתים בעץ עליהם רצינו להפעיל את function לבין מספר הצמתים שעליהם function הופעלה בפועל. Function לא תעבוד על צמתים ריקים, לדוג', ולפעמים m עלול להיות גדול ממספר הצמתים בעץ.

בחזרה ל-GetMostViewedClasses: נשתמש בפונקציה reversedInOrder עם פונקציה שמעתיקה את מס' הקורס ומספר ההרצאה למערכים הנתונים (אותם נעביר by reference. נבדוק את ערך ההחזרה מהפונקציה reversedInOrder. אם היא 0, סיימנו. כלומר מילאנו את המערכים הנתונים ב-numOfClasses תאים בדיוק. אחרת, יש להוסיף למערכים הרצאות שזמן הצפייה בהם הוא עדיין 0, לפי סדר ידוע מראש:

?????

***void ​Quit(​void \*\*​DS) :***