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.

בנוסף לפונקציות הרגילות של עץ AVL, הוספנו פונקציית חיפוש נוספת: **reverseInOrder**.

זוהי פונקציית חיפוש בסדר הפוך מחיפוש inOrder, כלומר חיפוש **בסדר יורד** של ערכים.

למרות שמה, הפונקציה הזו אינה פונקציית חיפוש פשוטה כמו inOrder. 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 reverseInOrder 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', ובפרט m עלול להיות גדול ממספר הצמתים בעץ. ערך ההחזרה של הפונקציה הוא m-S, כאשר S מציין את מספר הצמתי בעץ עליהם הפונקציה function פעלה.

לכל אורך התרגיל, נתייחס למס' הקורסים הכולל כ-n, ולמס' השיעורים הכולל במבנה כ-m.

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

* מזהה הקורס הייחודי.
* מספר השיעורים בקורס.

-course\_id

- num\_of\_classes

-classes\_pointers\_array[][]

-zero\_views\_node\_pointers

-zero\_views\_classes[]

* מערך של מצביעים לשיעורים בעץ השיעורים. אורכו כמספר השיעורים בקורס. נסמן בחישובי הסיבוכיות (לכל אורך התרגיל) את מספר השיעורים של קורס i ב-Ci, עבור 1 ≤ i ≤ n. מתקיים כי 1 ≤ Ci ≤ m, ולכל n > 1 מתקיים 1 ≤ Ci < m. בעת יצירת הקורס כל המצביעים במערך הם nullptr, כלומר ה-nodes של השיעורים טרם נוצרו. (לוודא עם שיר)
* מערך באורך Ci של מצביעים לשיעורים ברשימת "אפס צפיות": זוהי רשימה מקושרת דו כיוונית שבה כל שיעור עם id=i שטרם צפו בו נמצא בה בחוליה ה-i ברשימה.
* רשימה מקושרת של השיעורים חסרי הצפיות בקורס.

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

-class\_id

-course\_id

-\*parent

-time

-classes\_pointers\_array[][]

}

* 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 שה-nodes שלו הם מסוג courseNode, והשני עם צמתים מסוג classNode.

*סיבוכיות זמן:* *O(1)* (יצירת 3 מצביעים)

***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).

בנוסף, 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(n)).

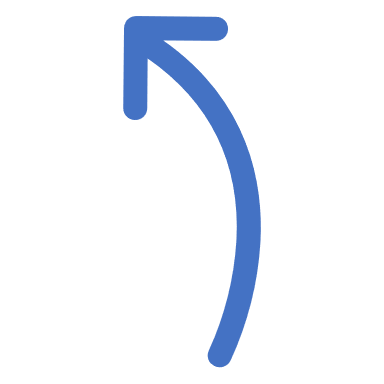
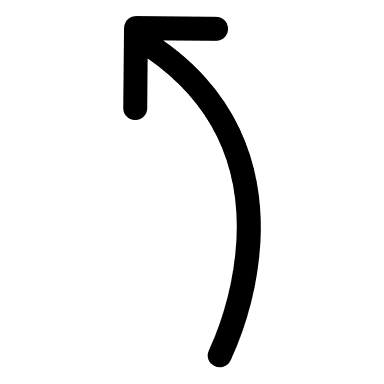
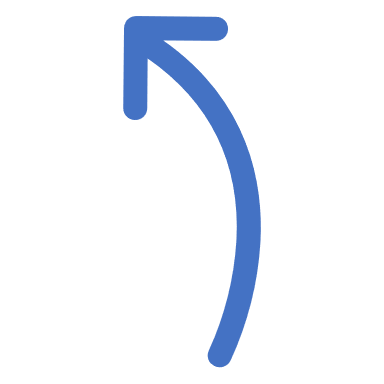
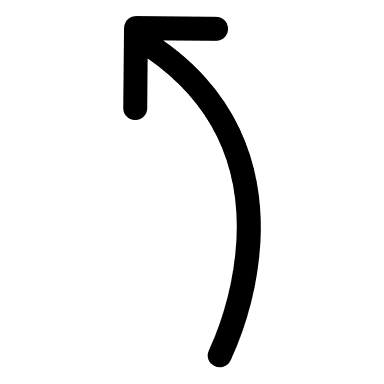
*סיבוכיות זמן:* *O(log(n)+numOfClasses)* (O(log(n))+2O(numOfClasses))

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

הפונקציה מבצעת חיפוש בעץ הקורסים לפי courseID (זמן log(n)). במידה ונמצא הקורס המבוקש, ניגש לכל אחד מהמצביעים שבמערכי המצביעים של הקורס, נוציא אותו מהעץ או מהרשימה ונמחק אותו. כל הוצאה מהעץ מתבצעת ב-O(log(m)), וכל מחיקה מתבצעת ב-O(1). כיוון שיש CID הרצאות למחוק, סך סיבוכיות הזמן של פעולות ההוצאה והמחיקה של כל ההרצאות מהקורס היא O(CID· log(m)).

מס' ההרצאות בקורס

בעל מזהה courseID

***סיבוכיות זמן:* *O( m*·*log(M ))* (O(log(n)+O(CID· log(m)=O(log(n)+CID**·** log(m))O(CID**·** log( m )))

מס' ההרצאות הכולל במבנה

*הערת הבהרה: לפי הסימונים שלנו לכל אורך התרגיל, m מסמן את מספר ההרצאות הכולל במבנה, ומס' ההרצאות של קורס מסויים עם מזהה ID הוא CID. נשים לב כי בדרישות הסיבוכיות שניתנו לתרגיל, ניתנו משתנים M ו-m, להם ניתנה משמעות זהה. על כן השוני הסמנטי בין ההסבר לבין התשובה הסופית, שכן m=CID ו-M=m.*

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

הפונקציה תבדוק קודם כל אם השיעור המבוקש נמצא ברשימה zero\_views של הקורס courseID.

אם כן, כלומר הצפייה שאנו נדרשים להוסיף היא הצפייה הראשונה של ההרצאה הזו, תימחק החולייה של השיעור מהרשימה המקושרת. ניצור classNode עם מזהה classID וזמן צפייה time, ונכניס אותו לעץ השיעורים. כל הפעולות מלבד ההכנסה לעץ הן פעולות בסיבוכיות קבועה (יש כאן 6 פעולות בדיוק) – בדיקת מצביע לרשימה מקושרת, עדכון שלושה מצביעים, מחיקת חוליה, יצירת ClassNode. סיבוכיות הכנסת איבר לעץ AVL בעל m איברים היא O(log(m)).

אחרת, קיים בעץ השיעורים classNode עבור השיעור classID של קורס courseID. נחפש את המצביע ל-classNode המבוקש וניצור העתק שלו הנבדל ממנו בזמן בלבד – הזמן ב-classNode החדש יהיה סכום הזמן של ה-classNode המקורי והזמן אותו נתבקשנו להוסיף (time). לאחר יצירת ה-classNode החדש, הפונקציה מוחקת מהעץ את ה-classNode המקורי ומכניסה את ה-classNode החדש. חיפוש, מחיקת והוספת איבר בעץ הן כולן פעולות שסיבוכיות הזמן שלהן היא O(log(m)). לפיכך, סך כל סיבוכיות הזמן של הפעולה היא O(3log(m)).

*סיבוכיות זמן:* *O( log(m ))* (max{O(log(m), O(3log(m))}=O(3log(m)))

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

הפונקציה מבצעת חיפוש בעץ הקורסים (log(n)). אם נמצא הקורס המבוקש, פונים למצביע שנמצא בתא ה-classID במערך המצביעים, ומכניסים את time ל-timeViewed.

*סיבוכיות זמן:* *O( log(n))* (חיפוש בעץ AVL בעל n צמתים)

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

נשתמש בפונקציה reversedInOrder עם פונקציה שמעתיקה את מס' הקורס ומספר ההרצאה למערכים הנתונים (אותם נעביר (by reference. נבדוק את ערך ההחזרה מהפונקציה reversedInOrder. (לבדוק – מה הסיבוכיות של reverseInOrder) אם הוא 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:

**???איך פתרנו את זה בסוף??? עשינו רשימה של קורסים שיש בהם עוד שיעורים חסרי צפיות???**

***void ​Quit(​void \*\*​DS) :***

הפונקציה מוחקת את כל מבנה הנתונים כך שייקרא ההורס של המערכת, ומחיקת המבנה נעשית כך:

ראשית נמחק עץ הקורסים: כל קורס מבצע מחיקה של המצביעים לעץ השיעורים ומחיקה של הרשימה ושל המצביעים לרשימה. לפיכך סך כמות האיברים שיש למחוק במקרה הגרוע היא 3Ci, כאשר Ci הוא מס' השיעורים בקורס ה-i שנמחק.

סך כל הקורסים במערכת הוא n, לכן סך כל סיבוכיות הזמן של המצביעים והרשימות של כלל הקורסים היא , אך נשים לב כי סכום כל ה-Ci הוא בדיוק m, כלומר כמות כל השיעורים במערכת. לפיכך סיבוכיות הזמן של מחיקת המצביעים והרשימות היא O(3m)=O(m).

בנוסף למצביעים ולרשימות, יש למחוק את העץ עצמו, בו יש n איברים מסוג courseNode, לכן יש O(n) פעולות נוספות שיש לבצע.

סיבוכיות זמן מחיקת עץ הקורסים – O(n+3m).

שנית, יש למחוק את עץ השיעורים. יש בו בדיוק m איברים מסוג classNode, אותם יש למחוק. כל classNode נמחק בסיבוכיות O(1), שכן הוא לא מחזיק מערך או רשימה, ולכן סיבוכיות מחיקת עץ השיעורים היא O(m).

*סיבוכיות זמן:* *O(n+m)* (O(n+3m)+O(m)=O(n+3m+m)=O(n+4m))

***סיבוכיות מקום כוללת של המבנה:***

סה"כ מדובר בשני עצי AVL, כאשר גודל עץ הקורסים הוא n וגודל עץ השיעורים הוא m.

בעץ הקורסים, כל איבר מחזיק בתוכו שני מערכים ורשימה מקושרת, גודל כל אחד כגודל מס' השיעורים בקורס.

בעץ השיעורים כל איבר הוא בגודל קבוע.

לפיכך סיבוכיות המקום הנדרשת עבור כל המבנה היא: