***מבנה הנתונים מכיל:***

***1 .******coursesData*** *– עץ AVL\_v2 (****הסבר על העץ - תחת הערות****), ממויין לפי מספר הקורס מהקטן לגדול, כאשר כל תא בעץ מכיל:*

*א. מספר קורס.*

*ב. מערך של מצביעים לרשימה מקושרת (ראו 2), כל אינדקס מייצג מספר כיתה (המערך בגודל מספר הכיתות שסופקו).*

*ג. גובה הצומת הנוכחית.*

*ד. בן שמאלי.*

*ה. בן ימני.*

*ו. מצביע לאבא (שורש – NULL).*

*בנוסף, המבנה מכיל מצביע smallest\_node שמצביע לצומת הקטן ביותר בעץ.*

***2. timeList*** *– רשימה מקושרת דו כיוונית, מתחילה עם תא head בעל ערך 0, מסודרת ע"פי הערך המספרי בתא (בעקבות צורת ההוספה של תא לרשימה) מהקטן לגדול, כאשר כל תא בה מכיל:*

*א. מספר, המייצג את זמן הצפייה הנוכחי.*

*ב. עץ AVL של קורסים שיש בהם כיתות, נקבע לפי המספר (המייצג זמן) שבתא הרשימה (ראו 3).*

*ג. מצביע לאיבר הבא ברשימה.*

*ד. מצביע לאיבר הקודם ברשימה.*

***3. currTimeCourses*** *– עץ AVL\_v2 ,ממויין לפי מספר הקורס מהקטן לגדול, לכל תא ברשימה המקושרת יש עץ (עם מידע שונה) כנ"ל, כאשר כל תא בעץ מכיל:*

*א. מספר קורס.*

*ב. עץ AVL של כיתות שצפו בהן, ע"פי מספר הקורס וכמות הזמן שהתא אליו הם מחוברים מייצג. (כלומר, לכל כיתה יש רק תא אחד שמתאים לה ברשימה המקושרת), (ראו 4).*

*ג. גובה הצומת הנוכחית.*

*ד. בן שמאלי.*

*ה. בן ימני.*

*ו. מצביע לאבא (שורש – NULL).*

*בנוסף, המבנה מכיל מצביע smallest\_node שמצביע לצומת הקטן ביותר בעץ.*

***4.******currTimeClasses*** *– עץ AVL\_v2 , ממויין לפי מספר הכיתה מהקטן לגדול, לכל צומת בעץ currTimeCourses יש עץ (עם מידע שונה) כנ"ל, כאשר כל תא בעץ מכיל:*

*א. מספר כיתה ( המס' נע בין 0 למקס' כיתות שהוכנסו ע"י היוזר), בהתאם לזמנים.*

*ב. גובה הצומת הנוכחית.*

*ג. בן ימני.*

*ד. בן שמאלי.*

*ה. מצביע לאבא (שורש – NULL).*

*בנוסף, המבנה מכיל מצביע smallest\_node שמצביע לצומת הקטן ביותר בעץ.*

***5.******max\_time\_cell*** *– פוינטר לאיבר ברשימה המקושרת בעל הערך (זמן) הגבוה ביותר.*

***הערות:***

* *כאשר אנו כותבים על עץ AVL "ממויין", הכוונה כמובן לפי סדר inorder.*
* *כשכתבנו בקובץ את המילה "כיתה", זהו תרגום חופשי לclass והתכוונו לשיעור.*

***עץ AVL\_v2*** *הוא עץ AVL " משודרג" כך שבנוסף לעץ הAVL הרגיל, קיים מצביע בשם smallest\_node שמחזיק כתובת אל הצומת הקטן ביותר בעץ, ולכל תא בעץ יש מצביע parent שמחובר אל הצומת האבא של התא.*

*השורש (הצומת העליונה): יחזיק NULL במצביע parent.*

*במקרה של צומת בודד: smallest\_node יצביע אל הצומת הנ"ל.*

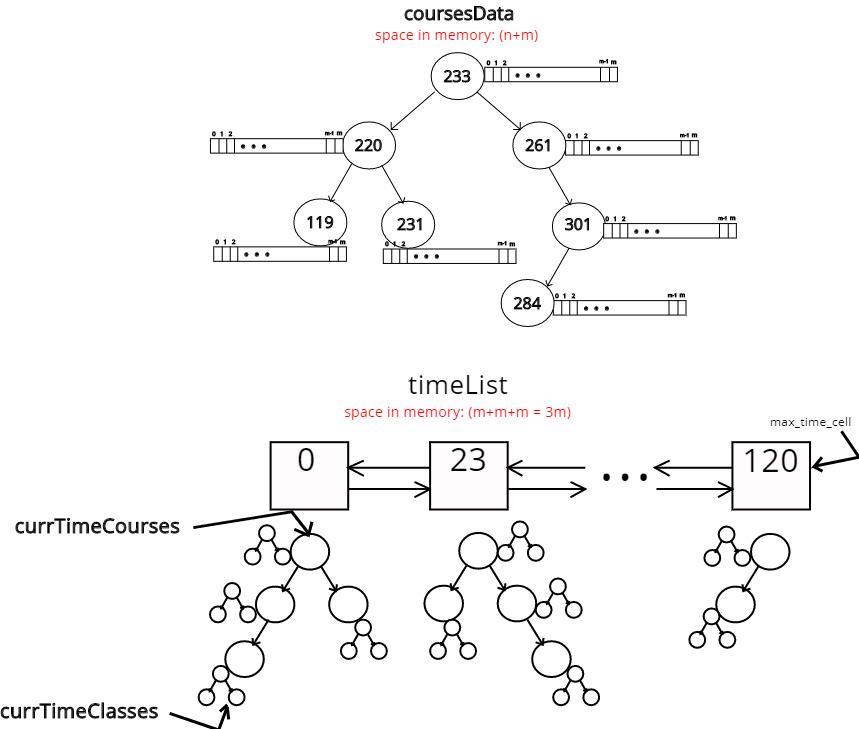
*בגלגולים: נדאג לעדכן בנוסף לleft וright את מצביע parent, העדכון הוא חיבור וניתוק של עד 3 מצביעים ולכן אין שינוי בסיבוכיות הזמן.*

*בעת הכנסה של איבר: נבצע הכנסה כפי שלמדנו ונעדכן גם את parent של הצומת החדשה.*

*במקרה של הסרה: נבצע הסרה כפי שלמדנו, נעביר (במקרה הצורך) את הparent של הנמחק לparent של הבנים שלו.*

*סיבוכיות מקום: נשארת O(n) כאשר n מס' הצמתים בעץ - הוספנו רק מצביע לכל צומת.*

***איור להמחשה:***

*******כמות הקורסים: n, כמות השיעורים במערכת: m. (הסבר על הסיבוכיות תחת "סיבוכיות")*

***מימושים לפונקציות:***

***- Init() .1***

***א.*** *צור מצביע למבנה ריק (עץ AVL + Linked list) mainData, שמכיל:*

*1. courseData (עץ AVL ריק).*

*2. timeList רשימה מקושרת.*

*3. max\_time\_cell – כרגע NULL (עד לשלב ג).*

***ב.*** *צור תא 0 ברשימה המקושרת timeList.*

***ג.*** *עדכן את פוינטר max\_time\_cell כך שיצביע לתא 0 ברשימה המקושרת.*

*-* ***AddCourse(void\* DS, int courseID, int numOfClasses) .2***

***א.*** *צור צומת חדשה עם הערך courseID והכנס את הצומת לעץ courseData.*

***ב.*** *הכנס לצומת הנ"ל, בערך המתאים, מערך של פוינטרים שכולם מצביעים לתא 0 של הרשימה המקושרת.*

***ג.*** *אם לא קיים עץ כזה, צור בתא 0 ברשימה המקושרת עץ AVL חדש (בערך currTimeCourses). הכנס אל העץ צומת חדשה courseID (שתכיל את עץ הכיתות – שלב ד).*

***ד.*** *צור עץ AVL חדש עם כל המספרים מ0 עד numOfClasses-1, והכנס אותו לצומת מסעיף ג.*

***- RemoveCourse(void\* DS, int courseID) .3***

***א.*** *גש לצומת המתאים (בעל ערך courseID) בcourseData (אם לא קיים כזה החזר FAILURE), והחל בסריקה של מערך הפונטרים שבצומת הנ"ל.*

***ב.*** *אם הכתובת שבתא במערך היא NULL המשך לתא הבא במערך. אחרת, גש לכתובת שבתא במערך.*

***ג.*** *גש לצומת courseID שבעץ currTimeCourses (שבתא הרשימה המקושרת), וסרוק את העץ currTimeClasses כך שבכל כיתה שנתקלים בה – עדכן את התא שבאינדקס המתאים במערך המצביעים להיות NULL (מסמנים שטיפלנו כבר בכיתה זו).*

***ד.*** *מחק את הצומת (הקורס) שבעץ currTimeCourses (שניגשנו אליו בסעיף ג).*

***ה.*** *בדוק אם הערך (העץ) currTimeCourses שבתא הרשימה המקושרת ריק, אם כן – בדוק האם max\_time\_cell מצביע לתא הרשימה המקושרת הנוכחית.*

***ה.1.*** *אם הוא אכן מצביע לתא, בדוק אם זהו התא 0 שברשימה, אם כן אל תעשה כלום. אחרת – עדכן את max\_time\_cell להצביע לתא הקודם לו ברשימה.*

***ה.2*** *אם התא אינו תא 0, מחק את התא מהרשימה המקושרת (תוך שמירה על האיברים הקודמים והבאים ברשימה).*

***ו.*** *מחק את הצומת (שניגשנו אליה בסעיף א) מcourseData. (סעיף זה יקרה לאחר שסיימנו לסרוק את מערך הפוינטרים בצומת).*

***- WatchClass(void\* DS, int courseID, int classID, int time) .4***

***א.*** *גש לצומת המתאים (courseID) בcourseData ואז גש לכתובת (תא ברשימה המקושרת) שבאינדקס classID במערך הפוינטרים שבצומת, שמור את ערך הצומת במשתנה (curr\_class\_time).*

***ב.*** *גש לצומת courseID שבcurrTimeCourses ואז גש לצומת classID (שבcurrTimeClasses, בצומת courseID אליה ניגשנו) ומחק את הצומת הנ"ל.*

***ב.1.*** *אם העץ currTimeClasses נשאר ריק (אין כיתות מהקורס בזמן הנוכחי), מחק את הצומת courseID מהעץcurrTimeCourses שבתא.*

***ב.1.1*** *אם העץ currTimeCourses נשאר ריק וגם כתובת התא שמחזיק את העץ הנ"ל היא זו שמאוכסנת בmax\_time\_cell, עדכן את הערך של התא להיות ערכו + time, והכנס בחזרה לתא את courseID כצומת לcurrTimeCourses והכנס את classID ולcurrTimeClasses שבצומת.* [מטפל במקרה של תא מקסימום].

***ב.2.1*** *אחרת (העץ נשאר ריק), אם לא מדובר בתא 0, מחק את התא מהרשימה המקושרת.*

***ג.*** *בדוק החל מהתא ברשימה המקושרת (אליו ניגשנו בסעיף א), האם (curr\_class\_time + time) גדול או שווה לתא הבא אחריו, עד שמוצאים תא שערכו שווה/גדול מ (curr\_class\_time + time) או עד לNULL.*

***ג.1.*** *אם ערך התא שווה ל(curr\_class\_time + time), גש לעץ currTimeCourses שבתא. אם יש בעץ צומת courseID, הכנס אליה את classID. אם אין, הכנס צומת courseID לעץ currTimeCourses, ולתוכה הכנס לcurrTimeClasses צומת classID.*

***ג.2.*** *אחרת (ערך התא גדול מ(curr\_class\_time + time) או שאין כזה), צור תא חדש עם הערך (curr\_class\_time + time) ברשימה המקושרת, וחברו לרשימה (תוך שמירה על הסדר). צור עץ בcurrTimeCourses והכנס אליו את הצומת courseID, ולתוכה הכנס לcurrTimeClasses צומת classID.*

***ג.3.*** *בדוק האם ערך התא שבmax\_time\_cell קטן מהערך של התא שיצרנו, אם כן – עדכן את max\_time\_cell להצביע לתא שיצרנו.*

***ד.*** *עדכן את המצביע במערך הפוינטרים (מסעיף א) להצביע לתא המתאים ברשימה המקושרת (עם ערך הזמן החדש של הכיתה).*

***- TimeViewed(void\* DS, int courseID, int classID, int \*timeViewed) .5***

***א.*** *גש לצומת המתאים (courseID) בcourseData ואז גש לכתובת (תא ברשימה המקושרת) שבאינדקס classID במערך הפוינטרים שבצומת.*

***ב.*** *עדכן את timeViewed להיות שווה לערך התא (ברשימה המקושרת) אליו מצביע הפוינטר שבמערך אליו ניגשנו בסעיף א.*

***- GetMostViewedClasses(void\* DS, int numOfClasses, int \*courses, int \*classes) .6***

***א.*** *גש לתא (ברשימה המקושרת) שנמצא בפוינטר max\_time\_cell, ואז גש לעץ currTimeCourses שבתא. צור משתנה i=0 (שיזכור כמה איברים כבר עודכנו) ומשתנה פוינטר curr\_cell = max\_time\_cell.*

***ב.*** *התחל לולאת while שתרוץ כל עוד i<numOfClasses.*

***ג.*** *גש לצומת הקטן ביותר בעץ currTimeCourses שבתא curr\_cell, ושמור במשתנה curr\_course את הקורס הנוכחי (נזכיר שהגישה מתבצעת מידית מהמצביע smallest\_node) ובצע סיור על next\_b של הצמתים (עד לNULL).*

***ד.*** *עבור כל צומת מcurrTimeCourses בסיור inorder החל מהצומת הקטן ביותר, גש לצומת הקטן ביותר בעץ currTimeClasses שבתוכו ובצע סיור על next\_b של הצמתים (עד לNULL).*

***ד.1.*** *הכנס את curr\_course לcourses[i] ואת ערך הצומת שבסיור לclasses[i].*

***ד.2.*** *הוסף 1 למשתנה i ובדוק האם i==numOfClasses.*

***ד.3****. אם התנאי מתקיים – החזר SUCCESS.*

***ה.*** *(אם הגענו לשלב הזה, סימן שעברנו על כל הכיתות שמתאימות לזמן המקסימאלי ועלינו להמשיך לזמן הגדול הבא אחריו). התקדם לתא הבא (אחד אחורה) ברשימה המקושרת (curr\_cell = curr\_cell->previous).*

***ו.*** *אם התא curr\_cell הוא NULL (עברנו את תא 0), החזר Failure.*

***- Quit(void\*\* DS) .7***

***א.*** *הפוך את DS לNULL כך שיקראו הדיסטרקטורים של העצים והרשימה המקושרת.*

***נכונות:***

***- Init() .1*** *– מאתחל את מבנה הנתונים שלנו עם הערכים הנחוצים לנו במבנה.*

*-* ***AddCourse(void\* DS, int courseID, int numOfClasses) .2*** *– מוסיף את הקורס המבוקש לעץ הקורסים הראשי שלנו ויוצר מערך בו כל אינדקס מייצג את השיעורים הקיימים בקורס. כמו כן, דואג לעדכן כי כרגע דקות הצפייה בכלל השיעורים של הקורס הנוכחי הם 0.* ***(שימו לב, תא 0 תמיד ישאר ברשימה המקושרת, הוא אינו נמחק!).***

***- RemoveCourse(void\* DS, int courseID) .3*** *– מוחק תחילה את כל השיעורים של הקורס, כולל הקורס עצמו מהמבנה שמחזיק את דקות הצפייה, תוך מחיקת תאים מהרשימה המקושרת שאין בהם צורך יותר (כדי לשמור על סיבוכיות המקום). לאחר מכן דואג למחוק את הקורס מעץ הקורסים הראשי.*

***- WatchClass(void\* DS, int courseID, int classID, int time) .4*** *– דואג למחוק את השיעור מהערך הקודם של זמן הצפייה שלו, ובמקרה הצורך גם מחיקת קורס ותא מהרשימה המקושרת אם צריך. לאחר מכן דואג להוסיף (ובמקרה הצורך ליצור) לתא עם הערך המתאים את השיעור. בסיום תהליך עדכון מבנה הזמנים, מעדכן גם את התא במערך המתאים שבעץ הקורסים הראשי להצביע לערך הזמן הנכון.*

***- TimeViewed(void\* DS, int courseID, int classID, int \*timeViewed) .5*** *– דואג לעדכן את המצביע שהכניסו לנו לזמן המתאים לשיעור שבקורס המבוקש.*

***- GetMostViewedClasses(void\* DS, int numOfClasses, int \*courses, int \*classes) .6*** *– מעדכן את המערכים הנתונים לנו בשיעורים הנצפים ביותר, הדבר נעשה ע"י גישה למצביע ששמרנו לערך הצפייה המקסימאלי במערכת, ותוך כדי גישה מיידית לשיעור הקטן ביותר שבקורס הקטן ביותר שבתא עם ערך הצפייה הגדול ביותר, וביצוע סיור inorder החל ממנו. הפונקציה מבצעת עדכון של המערכים לפי סדר לקסיקוגרפי כאשר סדר עדכון המערכים יהיה לפי:*

*1. מספר הקורס מהנמוך לגבוה.*

*2. מספר השיעור מהנמוך לגבוה.*

*ועוברת לזמנים קטנים יותר (מהגדול לקטן) לפי הצורך. כמו כן, הפונקציה דואגת לעצור כאשר הגענו למספר הקורסים שהיוזר ביקש לעדכן במערכים (או בשגיאה כלשהי כמובן..).*

***- Quit(void\*\* DS) .7***

*הפיכת מצביע לNULL גורמת ליציאה מהסקופ כך שהדיסטרקטורים של מבני הנתונים שלנו נקראים.*

***סיבוכיות:***

***נסמן: n – מספר הקורסים במערכת, m – מספר שיעורים בקורס מסוים, M – מספר השיעורים בכלל המערכת.***

*1 .זמן של* ***Init()*** *–*

1. *אתחול מבנה נתונים שכולל: עץ ריק – O(1).*
2. *רשימה מקושרת עם איבר בודד – O(1).*
3. *מצביע – O(1).*

*סה"כ:* ***O(1)****.*

*2 .זמן של* ***AddCourse(void\* DS, int courseID, int numOfClasses)***  *-*

1. *הכנסת צומת חדשה לעץ AVL עם לכל היותר n צמתים (קורה פעמיים) – O(2log(n)).*
2. *בניית עץ AVL עם מספרים מ0 עד m – O(m).*

*סה"כ: O(2log(n)+m) =* ***O(log(n)+m)****.*

*3 .זמן של* ***RemoveCourse(void\* DS, int courseID)*** *–*

*נסתכל על המקרה הגרוע ביותר:*

*יש במערכת n קורסים, כאשר לקורס שמעוניינים למחוק יש m שיעורים, ובמערכת יש סה"כ M שיעורים.*

1. *מציאת צומת בעץ AVL עם n צמתים: O(log(n)).*
2. *סריקת מערך עם m תאים, כמספר השיעורים בקורס, כאשר לכל שיעור בקורס, במקרה הגרוע ביותר, יש זמן צפייה שונה – O(m).*

*ב.1. ניגשים לתא במצביע שבמערך – O(1).*

*ב.2. מחפשים את צומת הקורס בעץ AVL עם לכל היותר n צמתים: O(log(n)).*

*ב.3. מבצעים מחיקה של השיעור מעץ AVL + שינוי המצביע במערך – O(1).*

*סה"כ:* ***O(mlog(n)).*** *מתקיים (ונתון) בהכרח כי , לכן בהכרח מדובר ב: O(mlog(M)).*

* *בכל מקרה אחר, סיבוכיות הזמן דווקא קטנה יותר: בזמן צפייה זהה לכל השיעורים בקורס יתקבל O(m) פעולות – m פעולות מחיקה + m-1 פעולות בסדר גודל O(1).*

*4.זמן של* ***WatchClass(void\* DS, int courseID, int classID, int time)***  *–*

*נתייחס למקרה הגרוע ביותר: מעלים את ערכו של t כך שכל הערכים מערך התא הנוכחי ועד לערך התא הנוכחי + t קיימים ברשימה המקושרת עבור קורסים שונים. בנוסף, כמעט כל השיעורים במערכת שייכים לקורס הנתון, ונמצאים בעץ יחד עם השיעור המבוקש.*

*א. מציאת צומת בעץ AVL עם n צמתים (קורה לכל היותר 3 פעמים) – O(3log(n)).*

*ב. מציאת שיעור בעץ AVL עם לכל היותר m שיעורים – O(log(m)).*

*ג. מחיקת/יצירת צומת/עץ ריק/תא ברשימה מקושרת – O(1).*

*ד. מעבר על t תאים עד למציאת המקום המתאים לשיעור ברשימה המקושרת – O(t).*

*סה"כ: O(3log(n)+log(m)+t), מתקיים כי M>m וגם M>n ולכן:*

*5.זמן של****TimeViewed(void\* DS, int courseID, int classID, int \*timeViewed)***  *–*

*א. מציאת צומת בעץ AVL עם n צמתים – O(log(n)).*

*ב. גישה לאיבר במערך + עדכון מצביע – O(1).*

*סה"כ:* ***O(log(n)).***

*6.זמן של* ***Quit(void\*\* DS) -***

***GetMostViewedClasses(void\* DS, int numOfClasses, int \*courses, int \*classes) –***

*\*\* כאן הערך m = numOfClasses. \*\**

*א. גישה למצביע + יצירת משתנים – O(1).*

*ב. ביצוע לולאה כאשר עוצרים לאחר m פעולות:*

*ב.1. גישה למצביע (פעמיים) וגישה לתא ברשימה מקושרת – O(1).*

*ב.2. עידכון מערכי הנתונים courses וclasses – O(1).*

*ב.3. ממשיכים בסיור inorder, סדר גודל של כמות האיברים שהדפסנו. – O(1).*

*סה"כ:* ***O(m)****.*

*7.זמן של*

*8 .מקום -*

*במקרה הגרוע ביותר, מתקיים שזמני הצפייה בכל השיעורים במערכת שונים, לכן, אנחנו משתמשים ב:*

*א. עץ AVL\_v2 עם n צמתים – O(n).*

*ב. בכל צומת בעץ יש מערך בגודל של מס' השיעורים בקורס - O(M).*

*ג. רשימה מקושרת עם לכל היותר M תאים שונים – O(M).*

*ג.1. במקרה כזה – יש לכל היותר קורס אחד שבו שיעור אחד – O(2M).*

*הסבר: בכל מצב, יש במבנה הנתונים של זמני הצפייה לכל היותר M שיעורים (כי אין כפילויות) ולהם יכולים להתאים לכל היותר M קורסים שונים.*

*סה"כ: O(4M+n) = O(M+n).*