**תיאור המבנה:**

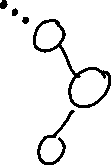
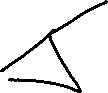
המבנה שלנו בנוי משלושה עצי AVL : עץ הקורסים, עץ השיעורים הנצפים ועץ השיעורים הלא נצפים

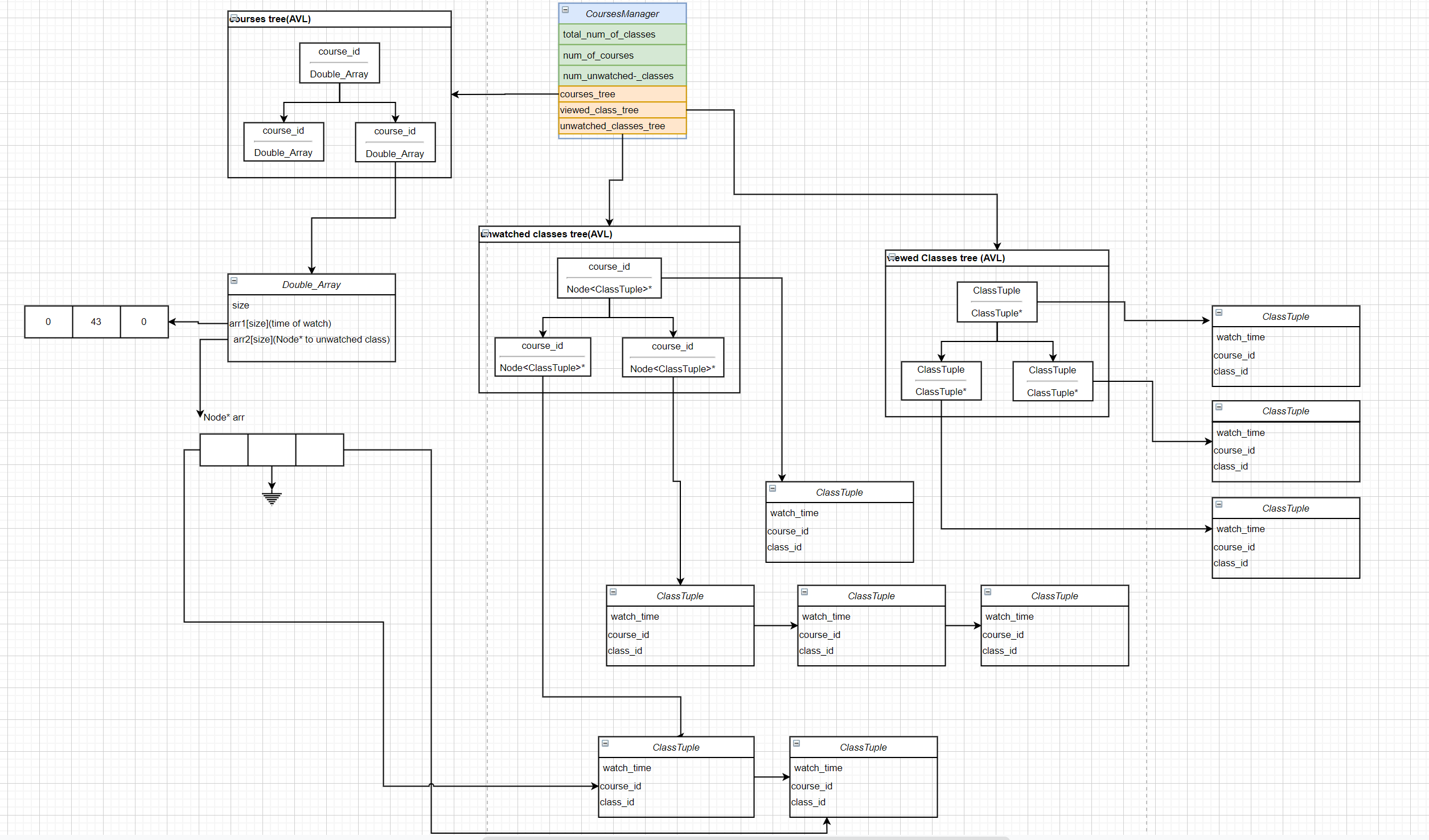
1. עץ הקורסים: זהו עץ אשר המפתח שלו הוא מספר הקורס והמידע שלו הוא "מערך כפול" בגודל של מספר השיעורים של אותו קורס. המכיל את זמן צפיה שלא אותו שיעור ומציע לאיבר ברשימת השיעורים הלא נצפים.
   1. "המערך הכפול": זהו מבנה שמכיל בתוכו שני מערכים בגודל מספר הכיתות. מערך של int שמציין את זמן הצפיה של כל שיעור. ומערך שני של מצביעים לאיברים ברשימה מקושרת שהיא רשימת השיעורים הלא נצפים
   2. רשימת השיעורים הלא נפצים: זהו רשימה שהאיברים בה הם מבנה שיצרנו שקוראים לו CLASSTUPLE. כל איבר ברשימה זאת מציין שיעור שלא נצפה.
   3. ClassTuple: מבנה שמכיל בתוכו שלושה משתנים, זמן הצפיה של אותו שיעור, מספר השיעור ומספר הקורס שאותו שיעור שייך אליו. מוגדרות אליו פעולות גדול, קטן ושווה לפי ההגדרה של איזה שיעור צריך לשים קודם במערך השיעורים הכי נצפים(יותר גדול מי שיש לו זמן צפיה יותר גדול, במקרה של שיוויון, זה בעל מספר הקורס הקטן יותר, ובמקרה של שיוויון שני זה בעל מספר השיעור הקטן יותר)
2. עץ השיעורים הנצפים: זהו עץ שאיברים הם ClassTupleים כאשר בהכרח זמן הצפיה של כל שיעור בו גדול ממש מ0
3. עץ השיעורים הלא נצפים: זהו עץ שמחזיק את רשימות השיעורים הלא נצפים. המפתח של כל איבר בעץ הוא מספר הקורס בו קיימים שיעורים לא נצפים והמידע הוא מצביע לרשימת השיעורים הלא נצפים שלא אותו קורס. נמצאים בו בהכרח קורסים שרשימת השיעורים הריקים בו לא ריקה.

מימוש עץ הAVL:

מימוש העץ AVL הוא עם שימוש באלגוריתמים בראינו בהרצאה, אם מעט שינוים.

* הוספת/הוצאת איבר: לאחר הוספת/הוצאת איבר לפי האלגוריתם הנלמד בכיתה, "נרד" במורד העץ עד לאיבר הכי גדול ועד לאיבר הכי קטן ונשמור מצביע אליהם. גובה עץ AVL הוא לכל היותר לוג מספר האיברים בו ולכן עדכון ושימרת מצביעים אלו לאחר הוספת/הוצאת איבר אינה פוגעת בסיבוכיות הזמן של פעולות אלו שכן הן בסיביוכיות זמן של לוג מספר האיברים
* הוספת פונקציות סיור inorder החל מצומת מסוימת בעץ (שימוש להוצאת השיעורים הכי נצפים): הפונקציה מקבלת את מספר האיברים שתרצה מהעץ ומצביע לצומת כלשהי בעץ (אנו ניתן לה את הצומת הכי גדולה או הכי קטנה) הפונקציה עושה סיור inorder כפי שראינו בכיתה החל מהצומת שניתנה לה ועוצרת במידה ועברה במספר האיברים המבוקש. אם אחרי סיור בכלל תת העץ שניתן לה לא עברנו בכמות הצמתים הדרושה היא "עולה" לאב של הצומת הנתונה ועושה את סיור הinorder בבן השני של אותו אב. אם גם זה לא מספיק היא "עולה" לאב של האב ופועלת באותו עיקרון בצורה רקורסיבית. היא תעבור במהלך הסיור רק בכמות האיברים הנדרשת ולכן סיבוכיות הזמן שלה היא כזו של כמות האיברים המבוקשת.



תרשים סכמתי של המבנה במצב ממוצע:

גם לשני אלו יש מצביע ל"מערך כפול" אבל לא משורטט בשביל לשמור על תרשים ברור יותר

* + - * ע
* **עע**

**Init()**

האלגוריתם:

1. נאתחל את מבנה הנתונים ריק- כלומר שלושה עצים ריקים(כלומר המצביע לעצים), ושלושה int-ים.
2. נחזיר מצביע המקרה של הצלחה וnull במקרה של כישלון

סיבוכיות זמן האלגוריתם:

1. O(1)- מספר סופי וקבוע של פעולות
2. O(1)- מספר סופי וקבוע של פעולות

לכן **בסה"כ** סיבוכיות האלגוריתם O(1)

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

האלגוריתם:

1. נוודא את תקינות הקלט ואם לא תקין נחזיר ערך מתאים
2. נוודא שהקורס אינו קיים במבנה שלנו על ידי בדיקה אם המספר קורס מופיע בעץ הקורסים אם מופיע נחזיר שגיאה
3. נוסיף לעץ הקורסים צומת בעלת מפתח זהה למספר קורס וערך של מצביע למבנה "המערך הכפול" שיצרנו.

* הסבר על מבנה "המערך הכפול" שיצרנו:

ביצירת המצביע אנו יוצרים שני מערכים בגודל מספר הכיתות. מאתחלים את המערך הראשון להכיל זמן צפיה לכל שיעור להיות 0, יוצרים רשימה של כל השיעורים שיש בקורס (רשימת השיעורים הלא נצפים) ושומרים מצביע לכל איבר ברשימה במערך השני.

1. הוספת צומת לעץ השיעורים הלא נצפים עם מפתח של מספר הקורס וערך של התחלת רשימת השיעורים הלא נצפים.
2. עדכון משתנים נוספים של המבנה כגון מספר השיעורים הכולל, מספר השיעורים הלא נצפים.

סיבוכיות זמן האלגוריתם:

1. בדיקה של משתנים o(1)
2. בדיקת ערך קיים בעץ AVL ראינו בהרצאה o(log(n)) כאשר n-מספר הקורסים הקיימים כבר במבנה
3. הוספת צומת לעץ AVL כנ"ל o(log(n)) יצירת "המערך הכפול" o(m) כאשר m=מספר השיעורים שיש בקורס, משום שאנחנו מקצים ומאתחלים מערך בגודל m ויוצרים רשימה בגודל m .
4. הוספת צומת לעץ AVL כנ"ל o(log(n))
5. עדכון של מספר משתנים סופי o(1)

* **סה"כ:** o(m+log(n)) כאשר n=מספר הקורסים הקיימים במבנה, m=מספר השיעורים שיש לקורס המוסף.

סיבוכיות מקום האלגוריתם:

אין קריאות רקורסיביות, פרט לקריאות הרקורסיביות של העצים בסיבוכיות של לכל היותר עומק העץ הכי גדול, שבמקרה הזה לכל היותר בגודל n לכן סיבוכיות מקום של הרקורסיה היא לכל היותר (log(n)) רק הקצה של שני מערכים בגודל m ויצירה של רשימה מקושרת באורך m לכן

* **סה"כ:** O(m+log(n)) כאשר m= מספר השיעורים בקורס המוסף ןn= מספר הקורסים במבנה

**StatusType RemoveCourse(void \*DS, int courseID)**

האלגוריתם:

1. נוודא את תקינות הקלט ואם הוא לא תקין נחזיר ערך מתאים
2. נמצא את הקורס בעץ הקורסים אם הוא לא נמצא נחזיר ערך מתאים
3. במידה ומצאנו מקודם את הצומת, ניגש למערך השיעורים של הקורס שנמצא בצומת הזאת, עבור כל שיעור שנצפה נשחרר אותו מעץ השיעורים הנפצים.
4. עבור כל שיעור שלא נצפה כבר, נשחרר את האיבר ברשימת השיעורים הלא נצפים, שאליו יש מצביע במערך זה.
5. נשחרר את "המערך הכפול".
6. במידה והיו שיעורים שלא נצפו נשחרר את הצומת של הקורס שהייה בה מצביע לרשימת השיעורים שלא נצפו של אותו קורס.
7. נחזיר הצלחה.

סיבוכיות זמן האלגוריתם:

1. O(1) -בדיקת מספר משתנים סופית
2. O(log(n)) כאשר n מספר הקורסים שיש במבנה- חיפוש בעץ AVL
3. O(m\*log(M)) כאשר m מספר ההרצאות של הקורס וM מספר ההרצאות הכולל במערכת- המקרה הכי גרוע יהיה כאשר כל אחד מהשיעורים שבמנה יהיה בעל זמן צפיה חיובי ממש כלשהו. ולכן לכל שיעור בקורס שנרצה להוציא נאלץ להוציא את האיבר שלו מעץ השיעורים הנצפים וזה יהיה בסיבוכיות של O(logM) מכיוון שזוהי הוצאה מעץ AVL שבמקרה הכי גרוע יש לו M איברים.
4. O(m)-לכל היותר m שיעורים שלא נצפו לכן שחרור של רשימה בגודל m
5. O(m)- שחרור של 2 מערכים בגודל m.
6. O(logn)-הוצאה מעץ(AVL) השיעורים הלא נצפים אשר לכל היותר בגודל n
7. O(1)- פעולה אחת

logM>=logn מכיוון שמספר ההרצאות במערכת בהכרח גדול יותר ממספר הקורסים ולכן

* **בסה"כ**: הסיבוכיות זמן תהיה בסדר גודל של O(mlogM) כנדרש

סיבוכיות מקום של האלגוריתם:

לא נדרשת הקצאה של מקום או פעולה רקוסיבית פרט לפעולות רקורסיביות של העץ כמו של הוצאה ובמקרה הכי גרוע זו תיהיה הוצאה מעץ השיעורים הנצפים כאשר כל השיעורים במבנה נצפו כבר. לכן סיבוכיות מקום: O(log(M)) כאשר M=מספר השיעורים הכולל במבנה

**StatusType WatchClass(void \*DS, int courseID, int classID, int time)**

האלגוריתם:

1. נוודא את תקינות הקלט ואם לא תקין נחזיר ערך מתאים
2. נוודא שהקורס קיים במבנה. אם לא נחזיר ערך מתאים
3. נוודא שלקורס הנ"ל יש מספר שיעור כזה על ידי בדיקת כמות מספר הכיתות. אם אין נחזיר ערך מתאים
4. נמצא את הקורס בעץ הקורסים, ונוסיף לזמן הצפיה שלו את הזמן המבוקש.
5. במידה וצפו בו לפני כן, נוציא את הצומת המתאימה לשיעור מעץ השיעורים הנצפים
6. במידה ולא צפו בו לפני כן, נמחק את באיבר ברשימת השיעורים הלא נצפים של אותו קורס בעזרת המצביע לאיבר זה השמור במערך השני.
   1. אם זה היה השיעור הלא נצפה האחרון, נוציא את הצומת של הקורס מעץ הכיתות הלא נצפות
7. נכניס לעץ הכיתות הנצפות את הכיתה הזאת לאחר עדכון זמן הצפייה בה.
8. נחזיר הצלחה (במידה ולא הייתה שגיאת הקצאה)

סיבכיות זמן האלגוריתם:

1. O(1) – בדיקה של מספר סופי של משתנים
2. O(log(n))- חיפוש הקורס בעץ(AVL) הקורסים. כאשר n=מספר הקורסים הקיימים כבר במבנה.
3. O(log(n))- חיפוש הקורס בעץ הקורסים שוב ובדיקה של גודל המערכים שלו בעזרת בדיקת משתנה הגודל של מבנה "המערך הכפול"
4. O(log(n))- שוב חיפוש הקורס בעץ הקורסים ושינוי ערך במערך אחד שנמצא ב"מערך הכפול" בO(1)
5. O(log(M)) בסיכוי הגרוע ביותר כל השיעורים במבנה נמצאים כבר בעץ השיעורים הנצפים ולכן נצטרך להכניס איבר בעץ AVL בגודל M. כאשר M= מספר השיעורים הכולל שנמצא במבנה.
6. O(1)- ניגשים למקום המתאים במערך השני של "המערך הכפול" ומוחקים את האיבר הבודד מהרשימה
   1. O(log(n))- במקרה הכי גרוע לכל אחד מהקורסים יש לפחות שיעור שלא נצפה ולכן עץ השיעורים הלא הצפים יכיל n איברים. וכן כפי שראינו בהרצאה הוצאה של איבר מעץ AVL היא בסיבוכיות הנ"ל. כאשר n=מספר הקורסים הכולל במבנה
7. O(log(M))במקרה הגרוע ביותר כל השיעורים נמצאים בעץ הכיתות הנצפות לכן הכנסה של איבר לעץ AVL שכזה היא זאת.
8. פעולה אחת

* **סה"כ** סיבוכיות זמן של O(log(M)) במקרה הגרוע (M>>n) כאשר M זה מספר השיעורים הכולל במערכת

סיבוכיות מקום האלגוריתם:

פה ההקצאה היחדיה היא של צומת בודדת בעץ השיעורים הנצפים ואין פעולות רקורסיביות פרט לאלו של פעולות העץ. העץ הכי גדול במקרה הגרוע ביותר יהיה עץ השיעורים הנצפים עם M איברים לכן סיבוכיות מקום של רקורסיה בו היא O(log(M))

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

האלגוריתם:

1. נבדוק את תקינות הפלט- אם הוא לא תקין נחזיר את הערך המתאים
2. נמצא את הקורס בעץ הקורסים- אם הוא לא נמצא נחזיר את הערך המתאים
3. ניגש לצומת המתאימה לקורס בעץ ונוציא מ"המערך הכפול" שלנו את זמן הצפיה השמור במקום התואם למספר השיעור במערך הראשון
4. נחזיר הצלחה(במידה ולא הייתה שגיאת הקצאה)

סיבוכיות זמן האלגוריתם:

1. O(1)- מספר סופי קבוע של פעולות יחידות
2. O(log(n)) כאשר n מספר הקורסים במערכת- חיפוש בעץ AVL בגודל n
3. O(log(n))- חיפוש בצומת בעץ(AVL) הקורסים וגישה למערך בO(1)
4. O(1)- מספר סופי קבוע של פעולות יחידות

* **סה"כ** הסיבוכיות זמן תהיה O(logn) כאשר n זה מספר הקורסים במערכת כנדרש.

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

האלגוריתם:

1. נבדוק את תקינות הקלט ונחזיר ערך מתאים
2. נבדוק אם קיימים במבנה שלנו מספיק שיעורים על ידי בדיקת המשתנה המתאים. אם לא קיימים נחזיר ערך מתאים.
3. נבצע מילוי של המערכים באמצעות פונקציה שעושה סיור inorder מהגדול לקטן בעץ שנותנים לה. ניתן לה את התת עץ שהשורש שלו הוא האיבר הכי גדול בעץ השיעורים הנצפים שלנו (מצביע לתת העץ הזה שמור לנו בעץ ומעודכן כפי שהוסבר מקודם ללא פגיעה בסיבוכיות הזמן של פעולות העץ)
4. במידה ולא היה מספיק שיעורים שנצפו, נמלא את המשך המערכים בעזרת פונקציה שעושה סיור inorder מהקטן לגדול בעץ שאיבריו הם מספר קורס ורשימה של שיעורים שנותנים לה. והפעם ניתן לה את תת העץ שהשורש שלו הוא הצומת הכי קטנה בעץ השיעורים הלא נצפים (כלומר הקורס בעל המזהה הכי נמוך מתוך כל הקורסים שקיימים להם שיעורים לא נצפים). (גם מצביע צומת הכי קטנה בעץ אנחנו תמיד שומרים ומעדכנים בלי לפגוע בסיבוכיות הזמן של פעולות העץ).
5. נחזיר הצלחה (במידה ולא הייתה בעיית הקצאה)

סיבוכיות הזמן של האלגוריתם:

1. O(1) – פעולה אחת של בדיקת מצביע
2. O(1) – בדיקת משתנה יחיד
3. O(m) -מעבר על לכל היותר המספר המבוקש של השיעורים בתוך העץ מהגדול לקטן והעתקה של ערכם למערכים. כאשר m הוא מספר השיעורים המבוקשים
4. O(m) – במידה ולא היה אף שיעור נצפה אחד נעבור על רשימות השיעורים הריקים ונעתיק איבר איבר מהם למערכים המתאימים.
5. O(1) – פעולה אחת

**סה"כ:** O(log(m)) כאשר m זה מספר השיעורים המבוקשים

סיבוכיות מקום של האלגוריתם:

אין הקצאות והפעולה הרקורסיבית של פונקצית הinorder היא לכל היותר לוג(מספר השיעורים המבוקשים) במידה וכולם נמצאים בעץ השיעורים הנצפים, כי בעצם נצטרך לעלות רקורסיבית עד שסימנו להעביר את כל השיעורים למערכים המתאימים, הנמצאים באופן ממיון בעץ(AVL) השיעורים הנצפים

לכן:O(log(m))

**void Quit(void \*\*DS)**

האלגוריתם:

1. נבדוק את תקינות הקלט ונחזיר ערך מתאים
2. על כל צומת בעץ הקורסים אנו משחררים את "המערך הכפול" תוך שיחרור של כל איבר ברשימה שיש אליו מצביע במערך המתאים ואז משחררים את הצומת
3. משחררים את כל הצמתים שקיימים בעץ השיערים הנצפים
4. משחררים את כל הצמתים בעץ השיעורים הלא נצפים

סיבוכיות הזמן של האלגוריתם:

1. O(1)- בדיקת מצביע אחד
2. O(M+N)- שיחרור כל המערכים שגודלם הכולל M- סך כל השיעורים במבנה. ומחיקה של N- מספר הקורסים במבנה צמתים מהעץ בצורה רקורסיבית (מחק בן ימני, מחק בן שמאלי, ואז מחק את עצמך)
3. O(M)- מחיקת לכל היותר M-מספר השיעורים הכולל של איברים מהעץ בצורה רקורסיבית (באותו אופן כמו הקודם)
4. O(N)- מחיקת N-מספר הקורסים במבנה של צמתים מתוך עץ השיעורים הלא נצפים שממיון לפי מספר הקורס בו יש שיעורים לא נצפים.

* **סה"כ:** O(M+N) כאשר M=מספר השיעורים במבנה וN=מספר בקורסים במבנה.

סיבוכיות מקום של האלגוריתם:

אין הקצאה אלא רק פעולה רקורסיבית של מחיקת איברים מהעץ שעומקה היא לכל היותר גובה העץ. המקרה הכי גרוע העץ הכי גדול שיכול להיות הוא עץ השיעורים הנצפים בעל M(מספר השיעורים הכולל במערך) ולכן גובהו לכל היותר log(M).

לכן סיבוכיות מקום:O(logM) כאשר M-מספר השיעורים במבנה.

**סיבוכיות מקום כוללת:**

גודל העץ הקורסים ועץ השיעורים הלא נצפים הוא לכל היותר N-מספר הקורסים הכולל

גודל עץ השיעורים הנצפים הוא לכל היותר M-מספר השיעורים הכולל במבנה

אם נחבר את גודל כל המערכים שיש בעץ הקורסים נקבל מערך גדול בגודל של כל השיעורים במבנה כלומר M.

כמו כך סכום אורך כל הרשימות של השיעורים הלא נצפים הוא M.

כל סיבוכיות המקום של הפעולות הרקורסיוביות קטנות מסיבכות המקום שנדש על מנת לשמור את הנ"ל. לכן **סיבוכיות מקום כוללת** של המבנה:

O(M+N) כאשר M=מספר השיעורים הכולל במבנה וN מספר הקורסים הכולל המערכת.