

מבני הנתונים שבהם השתמשנו:

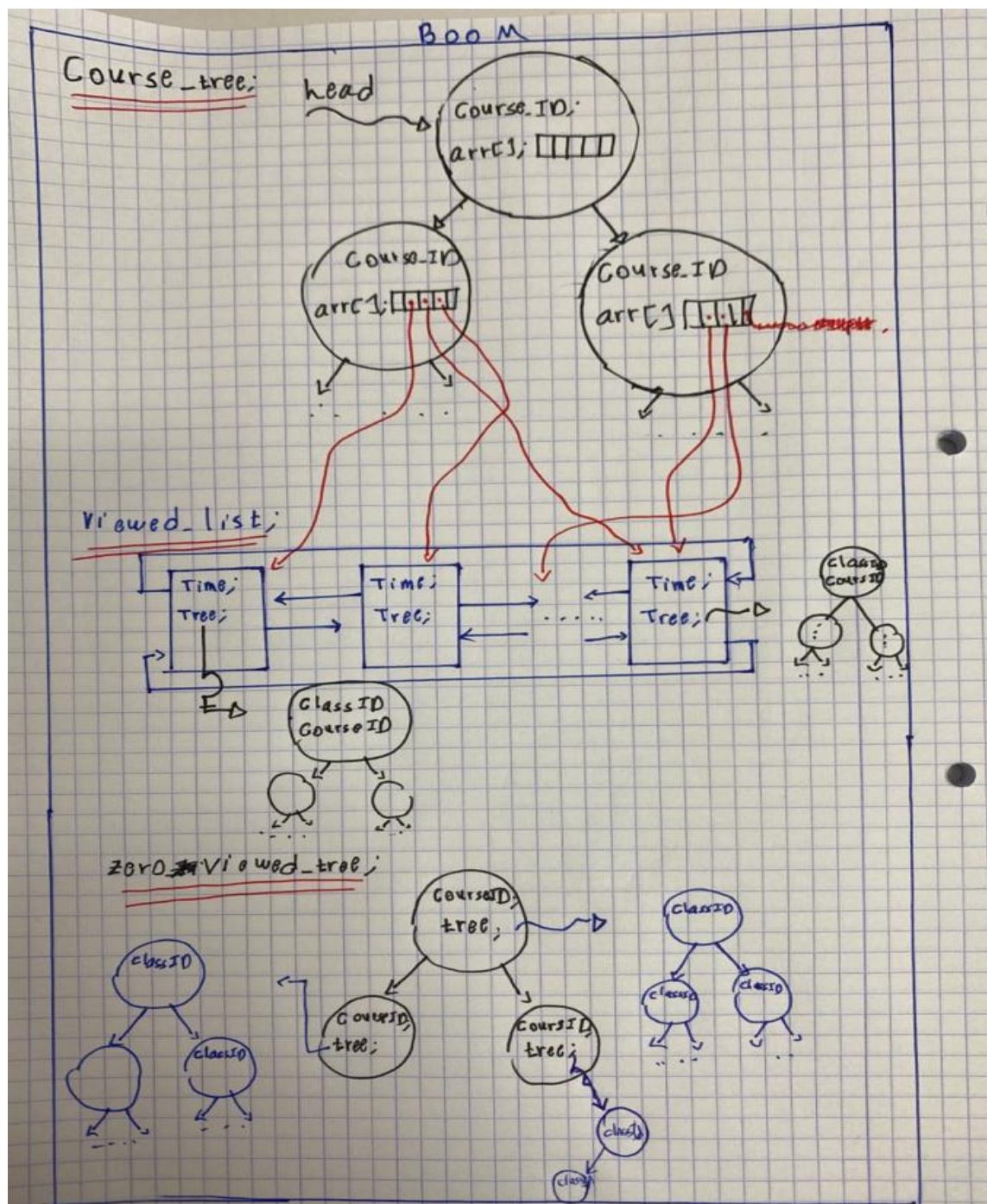
- AVLTree גנרי דומה למה שראינו בהרצאות, הוספנו לעץ שלנו מצביע שמצביע על האיבר הקטן ביותר בעץ
- AVLTreeNode שזאת מחלקה גנרית שיורשת מAVLTree ומוסיפה פונקציות ושדות שמאפשרות למחלקה הזו להיות Node בתוך העץ
- LinkedList מעגלית גנרית,
- כל איבר ברשימה LinkedList קראנו לו ListElement והאיבר הזה גם כן גנרי ומכיל שדה שנקרה Time מטיפוס int ומצביע גנרי (בקוד הצבענו לאיברים מטיפוס AVLTree)
- PointerNode שזה Node גנרי שמכיל בתוכו מספר הקורס ומערך של מצביעים (המצביעים גנריים)
- TreeNode שזה Node רגיל לא גנרי שמכיל בתוכו מספר הקורס ומספר השיעור שהו מציין

המבנה הראשי שבו השתמשנו במבנים אלו קראנו לו Boom, והו מכיל את השדות הבאות:

- Course_tree:
 - השתמשנו בשדה הזה כדי לאחסן את כל הקורסים והשיעורים
 - המבנה הוא מטיפוס AVLTree כך שכל איבר בעץ הוא מטיפוס PointerNode, וכל איבר במערך של PointerNode מצביע לאיבר מסוג ListElement
 - האיברים בעץ הזה מסמנים את הקורסים שהוספנו למערכת שלנו -Boom-
- Viewed_list:
 - השתמשנו בשדה הזה כדי לעקוב אחרי הזמן שנצפה בו כל שיעור בכל קורס
 - המבנה הוא מטיפוס LinkedList, כל איבר בLinkedList -שקראנו לו ListElement- מכיל בתוכו עץ מסוג AVLTree והעץ מכיל אברים מטיפוס TreeNode
 - כל איבר ברשימה שלנו -ListElement- מכיל בתוכו כל השיעורים מהקורסים השונים שצפו בהם הסטודנטים Time זמן
- Zero_viewed_tree:
 - השתמשנו בשדה הזה כדי לאחסן את כל השיעורים בכל הקורסים שאף סטודנט לא צפה בהם (TimeWatched=0)
 - המבנה הוא מטיפוס AVLTree וכל איבר בעץ הוא מטיפוס AVLTreeNode וכל איבר בעץ היורש הוא מטיפוס Node

- כל איבר בעץ שלנו מכיל עץ מטיפוס AVLTreeNode, העץ הזה מכיל שדה שנקרא `course_id` כלומר כל עץ מסמן קורס שונה והאיברים בעץ הזה הם השיעורים מהקורס שאף אחד לי צפה בהם

ציור למבנה הנתונים Boom:



מימוש הפעולות הנדרשות:

:Void* init()

דרישות סיבוכיות זמן: $O(1)$

אנחנו מאתחלים את שלושת השדות במבנה שלנו Boom

ראינו בהרצאה שאתחול של עץ ריק הוא בסיבוכיות זמן $O(1)$ ולכן השדות `course_tree` ו `zero_viewed_tree` מאותחלים שניהם בסיבוכיות $O(1)$ ובסה"כ נקבל סיבוכיות של $O(1)$

בנוסף ראינו בהרצאה שאתחול של רשימה ריקה הוא בסיבוכיות $O(1)$ ולכן אתחול השדה `viewed_list` הוא בסיבוכיות $O(1)$

בסה"כ נקבל סיבוכיות קבועה $O(1)$

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

דרישות סיבוכיות זמן: $O(m+\log(n))$ כאשר m הוא מספר השיעורים ו n הוא מספר הקורסים במערכת בזמן הפעולה

ראשית אנחנו מקצים `PointerNode` עם `course_id=courseID` ומערך בגודל m שכל אבריו מאותחלים ל `nullptr`

אחר כך מוסיפים את האיבר הזה לשדה `course_tree` שהוא מטיפוס `AVLTree`

בנוסף אנחנו מאתחלים עץ מטיפוס `AVLTreeNode` עם `course_id=coursed` ו m אברים לעץ מטיפוס `Node` שממוספרים מ 0 עד $m-1$ על ידי אלגוריתם שנספיר כעת ומוסיפים את העץ הזה לשדה הראשי `zero_viewed_tree`

הסבר לסיבוכיות:

- ראשית אתחול המבנה `PointerNode` עם ההשמה בשדה `course_id` היא בסיבוכיות זמן קבוע (כיוון שאלו פעולות קבועות שלא תלויות בקלט כלשהו) ואתחול המערך בגודל m כך שכל איבר מאותחל לערך מסוים כפי שראינו בהרצאה הוא בסיבוכיות של $O(m)$

○ בסה"כ נקבל עבוא האתחול הזה $O(m) + O(1) = O(m)$

- הוספת איבר לעץ `AVL` כפי שראינו בהרצאה הוא בסיבוכיות זמן של $O(\log(k))$ כאשר k הוא מספר האברים הנמצאים כעת בעץ

○ כיוון שכל איבר בעץ שלנו `AVLTree` מסמן קורס שונה שהוכנס אז העץ שלנו

מכיל n אברים -כאשר n הוא מספר הקורסים במערכת בזמן הפעולה- ולכן

הוספת האיבר תהיה בסיבוכיות זמן של $O(\log(n))$

- אתחול עץ מטיפוס AVLTreeNode עם m אברים
 - האלגוריתם: קודם כל אנחנו מאתחלים מערך מטיפוס int כך שכל איבר שווה לאינדקס שלו במערך גלומר $arr[i]=i$ ושוב ראינו בהרצאה שזה $O(m)$ כאשר m הוא אורך המערך
 - אנחנו קוראים לפונקציה לבנות את העץ שנקרא לה `orderedArrayToAVLTree` בפונקציה הזו:
 - מסתכלים על אמצע המערך, בונים Node כך שמספר הקורס הוא מספר הקורס שקבלנו בפונקציה הראשית ומספר השיעור הוא האינדקס האיבר שעומדים עליו שזה גם שווה לאינדקס של המערך
 - קוראים ריקורסיבית לפונקציה עבור החצי הימני של המערך והעץ שיוצא מחברים כבן ימני של Node שלנו
 - קוראים ריקורסיבית לפונקציה עבור החצי השמאלי של המערך והעץ שיוצא מחברים כבן שמאלי של Node שלנו
 - משוואת הסיבוכיות עבור מערך בגודל m היא $T(m)=2T(m/2)+c$ שראינו בתרגול שזה $O(m)$
- הוספת העץ מטיפוס AVLTreeNode לשדה הראשי `zero_viewed_tree` שהוא גם כן מטיפוס AVLTree כפי שראינו בהרצאה היא בסיבוכיות זמן של $O(\log(k))$ כאשר k הוא מספר האברים בעץ, כיוון שבעץ שלנו יש איבר אחד ויחיד לכל קורס נקבל שהוספה הזו היא בסיבוכיות $O(\log(n))$ כאשר n הוא מספר הקורסים במערכת בזמן הפעולה
- בסה"כ נקבל: $O(m)+O(m)+O(\log(n))+O(\log(n))=O(m+\log(n))$

:StatusType RemoveCourse(void* DS,int courseID)

- דרישות סיבוכיות זמן: $O(m*\log(M))$ כאשר M הוא מספר ההרצאות במערכת בזמן הפעולה, ו m הוא מספר ההרצאות של הקורס שהסרנו
- קודם כל אנחנו מחפשים את האיבר שמסמן את הקורס שרוצים למחוק בשדה `course_tree`, ראינו בהרצאה שחיפוש בעץ AVL הוא בסיבוכיות של $O(\log(n))$ כאשר n הוא מספר האברים בעץ שבמקרה שלנו הוא מספר הקורסים בעץ, כיוון שכל קורס מכיל לפחות שיעור אחד אז מתקיים ש $n \leq M$ ולכן סיבוכיות החיפוש היא $O(\log(M))$
 - אחר מכך אנחנו עוברים על המערך בגודל m שנמצא באיבר שמצאנו, עבור כל איבר מתקיים אחד משניים:
 - או שאיבר המערך שווה ל `nullptr`, ואז צריך לחפש את האיבר המתאים בעץ `zero_viewed_tree`, חיפוש העץ המתאים הוא בסיבוכיות $O(\log(n))$ וחיפוש של האיבר המתאים בעץ הפנימי הוא בסיבוכיות $O(\log(m))$, הראינו

שמתקיים $n \leq M$ וברור שמתקיים $m \leq M$ כיוון ש m הוא חלק מ M , ולכן
 $O(\log(n) + \log(m)) \leq O(2\log(M)) = O(\log(M))$

○ מקרה שני הוא שהמצביע מצביע לאיבר המתאים ברשימה, כל איבר ברשימה מכיל עץ AVL שמכיל את כל ההרצאות שנצפה בהם זמן T , במקרה הגרוע שכל ההרצאות נצפה בהם אותו זמן נקבל סיבוכיות להסיר של $O(\log(M))$ כפי שראינו בהרצאה

• כלומר עבור שני המקרים הסרת איבר היא בסיבוכיות של $O(\log(M))$, אנחנו מבצעים את ההסרה עבור m אברים ולכן סיבוכיות לפעולה זו היא $m \cdot \log(M)$ בנוסף לחיפוש בעץ `course_tree` שאמרנו שזה $O(\log(M))$ והסרת האיבר הזה מהעץ שגם זה $O(\log(M))$ ולכן בסה"כ נקבל $O(m \cdot \log(M) + 2\log(M)) = O(\log(M))$

:StatusType WatchClass(void* DS, int courseID, int classID, int Time)

דרישות סיבוכיות $O(\log(M) + t)$ כאשר M הוא מספר ההרצאות במערכת ו t הוא הזמן

- קודם כל אנחנו מחפשים את הקורס בשדה `course_tree` חיפוש בעץ AVL כפי שראינו בהרצאה הוא $O(\log(k))$ כאשר k הוא מספר האברים בעץ, בעץ שלנו כיוון שלכל קורס קיים לפחות שיעור אחד נקבל שמספר האברים הוא פחות מ M ולכן פעולת החיפוש היא בסיבוכיות של $O(\log(M))$
- במערך של `PointerNoden` של הקורס המפוקש `courseID` יש מצביעים לאברי הרשימה `viewed_list` עבור כל שיעור או הערך `nullptr`: אברי הרשימה מכילים את כל השיעורים שנצפה בהם `Time`, והערך `nullptr` אומר שהקורס לא נצפה בו ואז `Noden` שמסמן את השיעור נמצא ב `zero_viewed_tree`
 - אם השיעור המפוקש לא נצפה בו קודם אז אנחנו מסירים את האיבר שמסמן אותו מ `zero_viewed_tree` שהראינו בסעיף הקודם שזה מתבצעת בסיבוכיות של $O(\log(M))$, אחר מכך אנחנו מחפשים ב `viewed_list` איבר ברשימה בעל זמן שווה ל `Time`, ואם לא נמצא נבנה איבר ונקשר אותו
 - חיפוש איבר כזה ברשימה ממוינת בסדר עולה כאשר כל ערכיו חיוביים ושלמים הוא $O(\text{time})$ אנחנו עוברים איבר איבר עד שמוצאים איבר שגדול שווה `time`, יש בדיוק `time-1` אברים שקטנים ממנו ולכן הסיבוכיות במקרה הגרוע היא זאת)
 - אם המצביע אינו `nullptr` אנחנו עוברים על ידי המצביע לאיבר הרשימה שמכיל את `Noden` של השיעור, מסירים את האיבר מהעץ שזה כפי שהראינו בסעיף הקודם $O(\log(M))$
 - אחר מכך אנחנו מחפשים איבר ברשימה שמקיים שהזמן שלו גדול שווה מ `time+k` כאשר k הוא הזמן שנצפה בו השיעור לפני הפעלת הפונקציה, כיוון

שאנחנו כעת עומדים על איבר הרשימה פעל הערך t אז לכל היותר אנחנו נעבור על $time$ אברים כלומר סיבוכיות של $O(time)$

- בכל מקרה או שנמצא את איבר הרשימה שמקיים את דרישות הזמן, או שנבנה אחד ומקשר אותו שזה כפי שראינו בהרצאה עבור רשימה מעגלית קורה בסיבוכיות של $O(1)$
- אנחנו מוסיפים את האיבר שהסרנו לתוך העץ של איבר הרשימה שבנינו/מצאנו, כיוון שזה עץ AVL והוא מכיל שיעורים שנצפה בהם $time$ דקות, אז במקרה הגרוע -שכל ההרצאות נצפה בהם $time$ דקות- נקבל סיבוכיות זמן של $O(\log(M))$
- נשנה את המצביע ב `PointerNode` להצביע לאיבר הרשימה החדש שזה $O(1)$
- בסה"כ גם אם לא נפריד למקרים נקבל $O(4\log(M)+2Time)=O(\log(M)+Time)$

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

דרישות סיבוכיות: $O(\log(n))$ כאשר n הוא מספר הקורסים במערכת בזמן הפעולה

- אנחנו מחפשים את `Pointer node` של השיעור ב `course_tree` ומסתכלים על המצביע במערך בתא ה `classID`
- אם הוא `nullptr` נחזיר 0, אחרת נלך לאיבר הרשימה שהו מסמן ונחזיר `time` של איבר הרשימה
- `Course_tree` הוא עץ AVL שמכיל איבר לכל קורס במערכת הזמן הפעולה, ולכן לפי מה שראינו בהרצאה סיבוכיות זמן עבור פעולת חיפוש היא $O(\log(n))$ כאשר n הוא מספר הקורסים במערכת כנדרש

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

דרישות סיבוכיות: $O(m)$ כאשר m הוא מספר השיעורים המבוקשים

- אנחנו מסתכלים על השדה `viewed_list`, כיוון שזו רשימה מעגלית אנחנו יכולים ישר ללכת לאיבר האחרון ברשימה – שהו מכיל את הקורסים פעלי זמן צפיה `Time` הגדול ביותר – ואז אנחנו מבצעים חיפוש `InOrder` שמתחיל מהאיבר הקטן ביותר בעץ (הם מסודרים לפי מספר הקורס ואז מספר השיעור)
- אם מלינו את המערך נצא, אחרת נלך לאיבר הבא ברשימה (ברשימה אנחנו הולכים מהאחרון לראשון) ומבצעים אותה פעולה עד שנמלא את המערך
- אם בסוף הסיור ברשימה לא מלינו את המערך, אז אנחנו מבצעים אותה פעולה על `zero_viewed_tree` ומוסיפים את השיעורים שאף אחד לא צפה בהם מהקטן לגדול לפי מספר קורס ומספר שיעור

- כפי שראינו בהרצאה סיור InOrder נותן לנו מערך ממוין, בקריאות הרקורסוביות של העץ ובפעולה ראשית סמנו תנאי שאם נמלא את המערך אז נצא מיד, כלומר בקריאה ריקורסיבית במקרה הגרוע -המקרה שקראנו רק לבן שמאלי והעומק הוא m - אז סיבוכיות הזמן היא $O(2m)=O(m)$
- במקרה הגרוע הראשון שכל האברים הם בעץ אחד כפי שאמרנו נקבל סיבוכיות של $O(2m)=O(m)$
- במקרה שהם מפוזרים על עצים שונים, כלומר בעץ של האיבר הראשון קיים k_1 אחרים וכו כך ש $k_1+k_2+k_3+...+k_n=m$ אז נקבל סיבוכיות של $O(2k_1)+O(2k_2)+...+O(2k_n)=O(2(k_1+k_2+k_3+...+k_n))=O(2m)=O(m)$

:Quit(void DS)**

דרישות סיבוכיות $O(n+m)$ כאשר n הוא מספר הקורסים בזמן הפעולה ו m הוא מספר השיעורים הכולל בכל הקורסים

- בפונקציה הזו אנחנו מפעילים את ה Destructor של כל שדה ושדה
- עבור `course_tree`, העץ מכיל איבר לכל קורס ברשימה, מחיקת איבר -משחררים את המערך שהקצאנו,המצבעים לרשימות אנחנו משחררים כאשר מוחקים את הרשימה- היא בסיבוכיות קבועה,אנחנו מבצעים סיור PostOrder על העץ ומוחקים כל איבר שנתקלים בו
- הסיור כפי שראינו בהרצאה הוא בסיבוכיות של $O(n)$ כאשר n הוא מספר הקורסים בזמן הפעולה וכיוון שמחיקה היא בסיבוכיות של $O(1)$ אז נקבל סה"כ שמחיקה היא $O(n)$
- עבור `viewed_list` אנחנו מוחקים את כל אברי הרשימה איבר איבר, סיבוכיות מחיקה איבר הרשימה היא בסיבוכיות מחיקת העץ שהיא מכילה
- במקרה הגרוע כל הזמן של האברים שונה, כלומר קיים m אברי רשימה, מחיקה כל אחד כיוון שהיא מכילה רק איבר אחד היא $O(1)$ ואנחנו עוברים על m אברי רשימה ולכן נקבל סה"כ $O(m)$
- עבור `zero_viewed_list`, במקרה הגרוע שאף אחד לא צפה באף קורס נקבל שבעץ יש n עצים(כל עץ מסמן קורס) ובסה"כ יש להם m אברים
- במקרה הגרוע שהאברים מפוזרים על עצים שונים כאשר בעץ הראשון יש k_1 אברים וכו... נקבל שמחיקת האברים של העצים כולם היא בסיבוכיות של $O(k_1)+O(k_2)+...+O(k_n)=O(k_1+k_2+...+k_n)=O(m)$
- בנוסף אנחנו מבצעים סיור על n עצים שונים במקרה הגרוע שזה $O(n)$ בסה"כ נקבל $O(n+m)$
- סה"כ קבלנו סיבוכיות של $O(n)+O(m)+O(n+m)=O(n+m)$

סיבוכיות מקום:

- עבור `course_tree` אנחנו מקצים אברים עבור כל קורס כל שבכל איבר קיים מערך בגודל מספר ההרצאות של הקורס, ולכן בסה"כ $O(n+m)$
- עבור `viewed_list` אנחנו מקצים אברי רשימה עבור הרצאות שונות לפי זמן צפיה, בסה"כ בהנחה שלכל הרצאה הקצאנו איבר רשימה שונה אנחנו מקצים $O(2m)=O(m)$
- עבור `zero_viewed_tree` אנחנו גם כן מקצים עץ לכל קורס ולכל עץ מקצים איבר לכל שיעור, לכן במקרה הגרוע שאף אחד לא צפה באף שיעור נקבל סיבוכיות של $O(n+m)$
- בסה"כ נקבל סיבוכיות של $O(n+m)$