

### **מבנה הנזונים שבהם השתמשנו:**

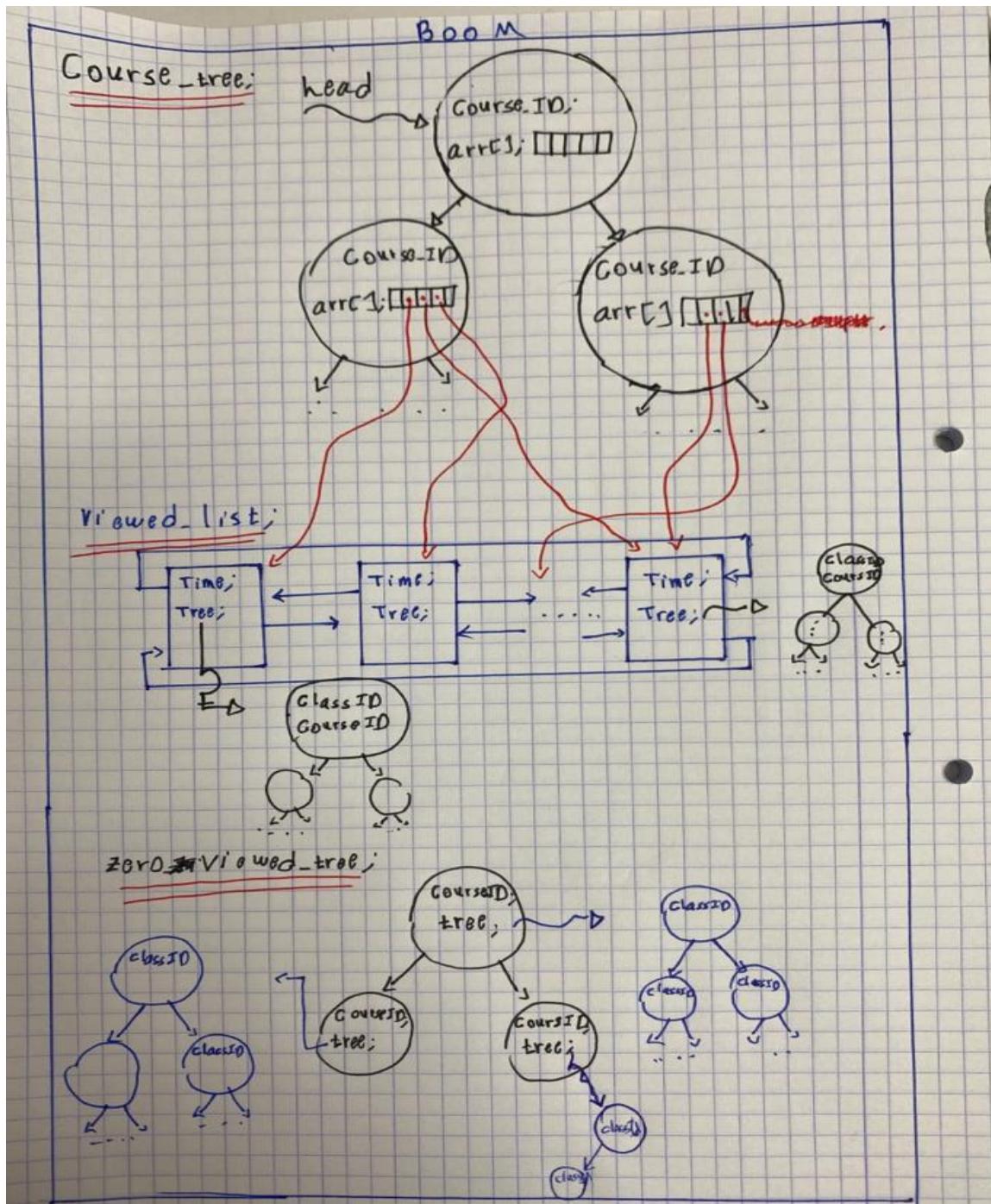
- AVLTree גנרי דומה למה שראינו בהרצאות, הוספנו לעצנו מצביע שמצוין על האיבר הקטן ביותר בעץ
  - AVLTreeNode שצתת מחלוקת גנרית שיורשת AVLTree ומושיפה פונקציות ושדות שמאפשרות למחילה זו להיות Node בתוך העץ
  - LinkedList מעגלית גנרית,
  - כל איבר ברשימה LinkedList קראנו לו ListElement והאיבר זהה גם כן גנרי ומכיל שדה שנקרא Time מטיפוס int ומצבייע גנרי (בקוד הצביענו לאיברים מטיפוס (AVLTree
  - PointerNode שזה Node גנרי שמכיל בתוכו מספר הקורס ומערך של מצביעים (המצבעים גנרים)
  - TreeNode שזה Node רגיל לא גנרי שמכיל בתוכו מספר הקורס ומספר השיעור שהוא מצין

המבנה הראשי שבו השתמשו במבנים אלו קראנו לו Boom, והוא 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 במבנה של PointerNodes מצביע לאיבר מסווג 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 -שקראננו לו->ListElement- מכיל בתוכו עצם מסוג AVLTree והעץ מכיל אברים מסוג TreeNode
    - כל איבר בראשימה שלנו -ListElement- מכיל בתוכו כל השיעורים מהקורסים השונים שצפו בהם הסטודנטים בזמן Time
  - Zero\_viewed\_tree
    - השתמשנו בשדה הזה כדי לאחסן את כל השיעורים בכל הקורסים שאף סטודנט לא צפה בהם (TimeWatched=0)
    - המבנה הוא מטיפוס Tree AVLTreeNode והוא מטיפוס AVLTreeNode וכל איבר בעץ הירוש הוא מטיפוס Node

- כל איבר בעץ שלנו מכיל עצ מטיפוס AVLTreeNode, העז זהה מכיל שדה שנקרא course\_id כל עז מסמן קורס שונה והאיברים בעז זהה הם השיעורים מהקורס שאף אחד לא צפה בהם

ציור למבנה הנתונים Boom:



## מימוש הפעולות הנדרשות:

**:Void\* init()**

דרישות סיבוכיות זמן:  $O(1)$

אנחנו מתחילה את שלושת השדות במבנה שלנו Boom

I course\_tree שערך ריק הוא בסיבוכיות זמן  $O(1)$  וכן course\_tree.zero\_viewed שערך ריק הוא בסיבוכיות זמן  $O(1)$  ובסה"כ קיבל סיבוכיות של  $O(1)$

בנוסף ראיינו בהרצאה שתחול של רשימה ריקה הוא בסיבוכיות  $O(1)$  וכן תחול השדה viewed\_list שערך ריק הוא בסיבוכיות  $O(1)$

בסה"כ קיבל סיבוכיות קבועה  $O(1)$

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

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

ראשית אנחנו מזמנים מקרים PointerNode course\_id=course\_id עם גודל  $m$  שכל אבריו  
מאוחלים לptr=null

אחר כך מוסיפים את האיבר הזה לשדה course\_tree שהוא מטיפוס AVLTree

בנוסף אנחנו מתחילה עץ מטיפוס AVLTreeNode course\_id=courses עם גודל  $m$  אברים לעץ  
מטיפוס Node שממוספרים מ 0 עד  $m-1$  על ידי אלגוריתם שנפיר כעת ומוסיפים את העץ  
זה לשדה הראשי zero\_viewed\_tree

הסבר לסיבוכיות:

- ראשית תחול המבנה PointerNode course\_id עם ההשמה בשדה id היא בסיבוכיות זמן קבוע (כיוון שלאו פעולות קבועות שלא תלויות בקלט כלשהו) ואתחול המערך בגודל  $m$  כרך שכל איבר מאוחל לערך מסוים כפי שראינו בהרצאה הוא בסיבוכיות של  $O(m)$

○ בסה"כ קיבל עבו אתחול זהה  $O(m) = O(1) + O(m)$

- הוספה איבר לעץ AVL כפי שראינו בהרצאה הוא בסיבוכיות זמן של  $O(\log k)$  אשר  
א הוא מספר האברים הנמצאים כעת בעץ  
○ כיוון שכל איבר בעץ שלנו AVLTree מסמן קורס שונה שהוכנס אז העץ שלנו  
מכיל  $k$  אברים - כאשר  $k$  הוא מספר הקורסים במערכת בזמן הפעולה. וכך  
הוספה האיבר תהיה בסיבוכיות זמן של  $O(\log k)$

- אתחול עץ מטיפוס AVLTreeNode עם כו אברים
  - האלגוריתם: קודם כל אנחנו מאתחלים מערך מטיפוס int[] arr שכל איבר שווה לאינדקס שלו במערך גלומר arr[i] ושוב ראיינו בהרצאה שזה  $O(n)$  כאשר כו הוא אורך המערך
  - אנחנו קוראים לפונקציה לבנות את העץ שנ Kraה לה orderedArrayToAVLTree
  - 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
  - משווות הסיבוכיות עבור מערך בגודל  $n$  היא  $c + \frac{2T(m)}{m} = O(2T(m))$  שראינו בתרגול שזה  $O(n)$
- 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, ציוון שבעץ שלנו יש איבר אחד ויחיד לכל קורס נקלט שהוספה זו היא בסיבוכיות  $O(\log n)$  כאשר ח הוא מספר הקורסים במערכת בזמן הפעולה
- בסה"כ נקבל:  $(O(n) + O(m) + O(\log(n))) = O(m + \log(n)) = O(m * \log(M))$

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

דרישות סיבוכיות זמן:  $O(M * \log(M))$  כאשר  $M$  הוא מספר הרצאות במערכת בזמן הפעולה, כו הוא מספר הרצאות של הקורס שהסרנו

- קודם כל אנחנו מחפשים את האיבר שמסמן את הקורס שרוצים למחוק בשדה course\_tree, ראיינו בהרצאה שחייב שבעץ AVL הוא בסיבוכיות של  $O(\log(M))$  כאשר זה הוא מספר האברים בעץ שבמקרה שלנו הוא מספר הקורסים בעץ, ציוון שכל קורס מכיל לפחות שיעור אחד אז מתקיים  $M \geq n$  ולכן סיבוכיות החיפוש היא  $O(\log(M))$
- אחר מכן אנחנו עוברים על המערך בגודל  $n$  שנמצא באיבר שמצאנו, עבור כל איבר מתקיים אחד משני:
- או שאיבר המערך שווה ל`NULL`, אז צריך לחפש את האיבר המתאים בעץ של האיבר המתאים בעץ הפנימי הוא בסיבוכיות  $O(\log(m))$ , הראיינו

- ש망קיים ש  $M \geq k$  וברור ש망קיים ש  $M < k$  כיוון ש הוא חלק מ  $M$ , ולכן  $(M)O = O(2\log(m) + \log(k))$
- מקרה שני הוא שהמצבי מצביע לאיבר המתאים בראשימה, כל איבר בראשימה מכיל עץ AVL שמכיל את כל הרצאות שנכפה בהם זמן  $T$ , במקרה הגרוע של הרצאות נכפה בהם אותו זמן נקבל סיבוכיות להסיר של  $(M)O$  כפי שראינו בהרצאה
  - כלומר עבור שני המקרים הסרת איבר היא בסיבוכיות של  $(M)O$ , אנחנו מבצעים את הרסירה עבור  $k$  אברים ולכן סיבוכיות לפעולה זו היא  $(M)O^k$  ועוד במקרה של  $course\_tree$  שאמרנו זהה  $(M)O$  והסרת האיבר הזה מהעץ שגם זה  $(M)O$  ולכן בסה"כ נקבל  $((M)O + 2\log(M)O) = O(M^2\log(M))$

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

דרישות סיבוכיות  $O(t+M)$  כאשר  $M$  הוא מספר הרצאות במערכת ו  $t$  הוא הזמן

- קודם כל אנחנו מחפשים את הקורס בשדה  $course\_tree$  חיפוש בעץ AVL כפי שראינו בהרצאה הוא  $(k)O$  כאשר  $k$  הוא מספר האברים בעץ, אך לנו כיוון שלכל קורס קיימן לפחות אחד נקלט במספר האברים הוא פחות מ  $M$  ולכן פועלות החיפושים היא בסיבוכיות של  $(M)O$
- במערך של `PointerNode` של הקורס המפקש `courseID` יש מצביעים לאברי הרשימה `viewed_list` עבור כל שיעור או הערך `nullptr`: אברי הרשימה מכילים את כל השיעורים שנכפה בהם `Time`, והערך `nullptr` אומר שהקורס לא נכפה בו ואז `Node` שמסמן את השיעור נמצא ב `zero_viewed_tree` נמצאת ב `zero_viewed_list`
- אם השיעור המפקש לא נכפה בו קודם אז אנחנו מסירים את האיבר שמסמן אותו `meshed_zero_viewed_tree` שהראינו בסעיף הקודם זהה מתבצעת בסיבוכיות של  $(M)O$ , אחר מכן אנחנו מחפשים `list_viewed` איבר ברשימה בעל זמן שווה ל `Time`, ואם לא נמצא נבנה איבר ונקשר אותו
- חיפוש איבר זהה ברשימה ממונית בסדר עולה כאשר כל ערכי חיבויים ושלים הוא  $(time)O$  אנחנו עוברים איבר אחר עד שמדובר איבר שגדל שווה `meshed_time`, יש לבדוק  $1-time$  אברים קטנים ממנו ולכן הסיבוכיות במקרה הגרוע היא זאת
- אם המצביע אינו `nullptr` אנחנו עוברים על ידי המצביע לאיבר הרשימה שמכיל את `Node` של השיעור, מסירים את האיבר מהעץ זהה כפי שהראינו בסעיף הקודם  $(M)O$
- אחר מכן אנחנו מחפשים איבר ברשימה שמיים שהזמן שלו גדול שווה מ  $time+k$  כאשר  $k$  הוא הזמן שנכפה בו השיעור לפני הפעלת הפונקציה, כיון

שאנו חזו כעת עומדים על איבר הרשימה פועל הערך  $\neq$  אז לכל היותר אנחנו נעבור על time אברים כלומר סיבוכיות של  $O(\text{time})$

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

#### **:StatusType TimeViewed(void\* DS, int courseId, int classId, int\* timeViewed)**

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

- אנחנו מוחשים את הpointer node של השיעור בcourse\_tree ומסתכלים על המצביע במערך בתא ה classID
- אם הוא **NULL** נחזיר 0, אחרת נלך לאיבר הרשימה שבו 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 פועלות חיפוש היא  $(\log(h))$  כאשר  $h$  הוא מספר הקורסים במערכת כנדרש

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

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

- אנחנו מסתכלים על השדה **list\_viewed**, כיוון שהוא רשימה מעגלית אנחנו יכולים ישירות ללקת לאיבר האחרון ברשימה – שבו מכיל את הקורסים פעלי זמן צפיה Time גדול ביותר – וזה אנחנו מבצעים חיפוש **InOrder** שמתחיל מהאיבר הקטן ביותר בעץ (הם מסודרים לפי מספר הקורס ואז מספר השיעור)
- אם מלאנו את המערך נצא, אחרת נלך לאיבר הבא ברשימה (ברשימה אנחנו הולכים מהאחרון הראשון) ובמצאים אותה פועלה עד שנמלא את המערך
- אם בסוף הסיוור ברשימה לא מלאנו את המערך, אז אנחנו מבצעים אותה פועלה על **tree\_viewed\_zero** ומוציאים את השיעורים שאף אחד לא צפה בהם מהקטן לגודל לפי מספר קורס ומספר שיעור

- כפי שראינו בהרצאה סיר Order גותן לנו מערך ממון, בקריאהות הרק'ירסוביוט של העץ ובפועלה ראשית סמן תנאי שאם נמלא את המערך אז נצא מיד, ככלומר בקריאה ריק'ורסיבית במקרה הגרוע - במקרה שקראנו רק לבן שמאלית והעומק הוא 0- אז סיבוכיות הזמן היא  $O(m^2)$
- במקרה הגרוע הראשון שכל האברים הם בעץ אחד כפי שאמרנו קיבל סיבוכיות של  $O(m^2)$
- במקרה שהם מפוזרים על עצים שונים, ככלומר בעץ של האיבר הראשון קיימ  $k_1 + k_2 + k_3 + \dots$  אז קיבל סיבוכיות של  $O(2k_1 + O(2k_2) + \dots + O(2kn)) = O(2(k_1 + k_2 + k_3 + \dots + kn)) = O(2m) = O(m)$

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

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

- בפונקציה זו אנחנו מפעילים את ה Destructor של כל שדה ושדה
- עבור course\_tree, העץ מכיל איבר לכל קורס בראשימה, מחייבת איבר - משחררים את המערך שהקצנו, המצביעים לרשימות אנחנו משחררים כאשר מוחקים את הרשימה- היא בסיבוכיות קבועה, אנחנו מבצעים סיר PostOrder על העץ ומוחקים כל איבר שנתקלים בו
- הסיר כפי שראינו בהרצאה הוא בסיבוכיות של  $(h)O$  כאשר  $h$  הוא מספר הקורסים בזמן הפעולה וכיון שמחיקה היא בסיבוכיות של  $(1)O$  אז קיבל סה"כ שמחיקה היא  $(h)O$
- עבור viewed\_list אנחנו מוחקים את כל אבר הרשימה איבר איבר, סיבוכיות מחיקה איבר הרשימה היא כסיבוכיות מחיקת העץ שהוא מכילה
- במקרה הגרוע כל הזמן של האברים שונה, ככלומר קיימ  $m$  אבר רשימה, מחיקה כל אחד כיון שהוא מכילה רק איבר אחד היא  $(1)O$  ונחנו עוברים על  $m$  אבר רשימה ולכן קיבל סה"כ  $(m)O$
- עבור list\_zero\_viewed, במקרה הגרוע שאף אחד לא צפה באף קורס מקבל שבעץ יש  $h$  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... קיבל שמחיקת האברים של העצים כולם היא בסיבוכיות של  $(m)O = (kn)O = (k_1 + k_2 + \dots + k_n)O = (k_1 + O(k_2) + \dots + O(k_n))O$
- בנוסף אנחנו מבצעים סיר על  $h$  עצים שונים במקרה הגרוע שזה  $(n)O$  בסה"כ קיבל  $(m+h)O$
- סה"כ קיבלנו סיבוכיות של  $(m+n)O = O(n+m) = O(n+m) + O(m) + O(h) = O(m+n+h)$

### סיבוכיות מקומ:

- עבור `course_tree` אנחנו מזמנים אברים עבור כל קורס כל שבכל איבר קיימ מעריך גודל מספר הרצאות של הקורס,ולכן בסה"כ  $O(m+n)$
- עבור `viewed_list` אנחנו מזמנים אבר רשימה עבור הרצאות שונות לפי זמן צפיה,בסה"כ בהנחה שלכל הרצאה הקצינו איבר רשימה שונה אנחנו מזמנים  $O(2m)=O(m)$
- עבור `zero_viewed_tree` אנחנו גם כן מזמנים עץ לכל קורס ולכל עץ מזמנים איבר לכל שיעור,לכן במקרה הנוכחי שאף אחד לא צפה באף שיעור נקבל סיבוכיות של  $O(n+m)$
- בסה"כ נקבל סיבוכיות של  $(m+n)O$