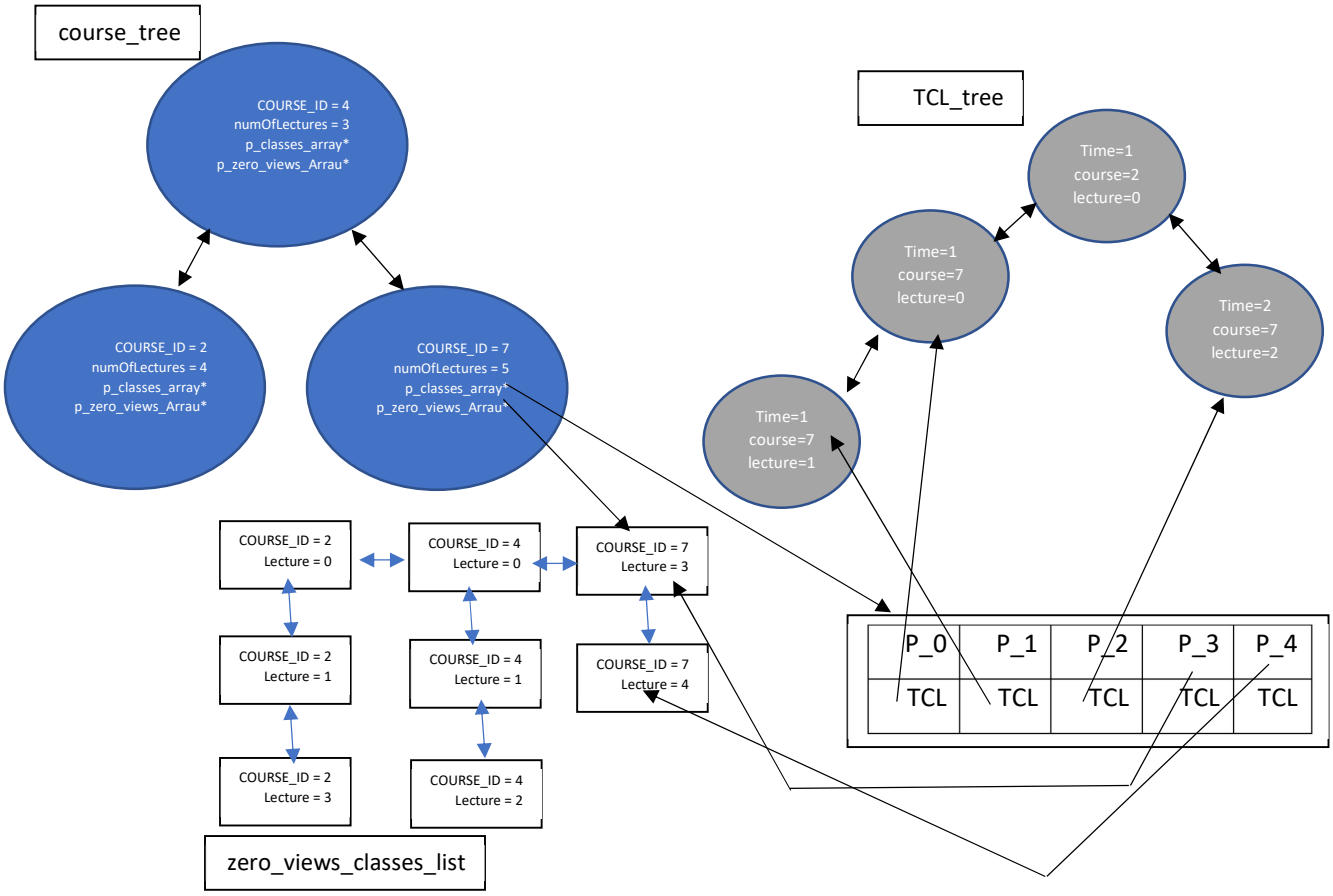
חלק יבש

מבנה נתונים מכיל:

1. course_tree – עץ חיפוש שהמפתחות בו הם course_ID.
בכל צומת course נשמרים:
lectures – פוינטר למערך של ההרצאות של הקורס.
כך שלכל הרצאה יהיה אובייקט מסוג TCL ופוינטר כך שבהתחלה הוא יצביע לnode המתאים לlecture זה ברשימה המקושרת zero_views_lectures_list. לאחר שההרצאה תיצבור את הצפייה הראשונה, נמחק את הnode המתאים ברשימה של ההרצאות ללא צפיות ונעדכן את הזמן הצפייה בTCL של ההרצאה זאת.
p_zero_views_classes – פוינטר לראש הרשימה המקושרת של ההרצאות עם 0 זמן צפייה של course.
courseID – מספר הקורס
numOfLectures – מספר ההרצאות בקורס

2. tcl_tree(time,course,lecture) – עץ חיפוש שהמפתחות בו הם time,course,lecture, כך שהחשיבות למפתח היא לפי הסדר משמאל לימין(כלומר, קודם time אחכ course ואז lecture).
בכל צומת tcl נשמר אובייקט עם 3 ערכים מטיפוס int – שהם (time,course,lecture).
בנוסף העץ יכיל MaxNode שיצביע לnode עם הערך הגדול ביותר.

3. zero_views_classes_list - רשימה מקושרת דו כיוונית שכול node בה הוא ראש של רשימה מקושרת דו כיוונית,
כך שעל כל node מצביע p_zero_views_classes, כלומר, הפוינטר של הcourse המתאים.
מכל Node יוצאת רשימה מקושרת דו כיוונית כך שעל כל node בתוכה מצביע ההרצאה המתאימה, כלומר הפוינטר של הlecture המתאים.



מימושים לפונקציות:

Void *init()

אתחל עץ course_tree, עץ tcl_tree, פוינטר לרשימה מקושרת zero_views_class_list.
כולם ריקים בהתחלה לכן סיבוכיות זמן היא O(1).

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

ניצור מצביע לצומת חדשה בעץ course_tree.
נכניס אותה לעץ course_tree כך שהעץ יישמר כעץ חיפוש.
נאתחל את הפרמטרים שלה באופן הבא:
- פוינטר לאבא שלה ופוינטרים לילדים שלה(אם יש).
- course_ID – לפי הנתון.
- p_zero_views_classes – ניצור Node חדש שיהיה ראש הרשימה המקושרת של course זה.
נבצע חיפוש בעץ course_tree ונמצא את הצומת J שהכי קרובה מבחינת courseID לצומת החדשה,
נוסיף את הnode מימין או משמאל לצומת J שנמצאה הכי קרובה לצומת החדשה, בהתאם ליחסים
בניהם – אם גדול יותר אז מימין, אם קטן יותר אז משמאל, כלומר סדרה עולה.
-lectures – ניצור מערך בגודל numOfClasses.
נעבור על המערך, ועבור כל תא נשרשר Node של הlecture המתאים ב- p_zero_views_classes .
נאחסן בכל תא את הפוינטר המתאים.

סיבוכיות זמן:
- הכנסה של הצומת לעץ ((O(log(n.
- מציאת הצומת שקרובה ביותר לצומת הנתונה ((O(log(n.
- אתחול מערך בגודל והשמה של פוינטר מתאים O(m) כאשר m=numOfClasses.
סכ"ה :O(log(n) + m) .

StatusType RemoveCourse(void *DS, int courseID)

ראשית נבצע חיפוש בעץ course_tree ונמצא את הצומת המתאים עם המפתח courseID.
נעבור על המערך lectures ועבור כל הרצאה:
-אם קיים TCL נמצא אותו בעץ TCLTree ונמחק אותו
לאחר מכן נמחק את כל הרשימה של הקורס המתאים ב p_zero_views_classes
לאחר שסיימנו, נמחק את הצומת מהעץ course_tree וגם שם לאחר המחיקה נבצע גלגולים
בהתאמה.

סיבוכיות זמן:
- חיפוש בעץ course_tree ((O(log(n
- מעבר על המערך p_class_array O(m) כאשר m=numOfClasses ובנוסף פעולות הגלגול הם
O(log(M) כאשר M הוא מספר ההרצאות במערכת בזמן הפעלה.
- פעולות הגלגול על course_tree הם ((O(log(n.
סכ"ה: ((O(mlog(M

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

ראשית נבצע חיפוש בעץ course_tree ונמצא את הצומת המתאים עם המפתח courseID.
ניגש למערך lectures של הcourse למקום הclassID. כעת יש 2 מקרים:
- אם עוד אין להרצאה צפיות, אז ניצור לה צומת חדשה ונכניס אותה באופן ממוין לtcl_tree.נמחק את הnode המתאים בp_zero_views_classes להרצאה זאת.
- אם יש לה צפיות, נגדיל בtime את מספר הצפיות שלה ונכניס את הצומת מחדש לעץ תוך שמירה על העץ כעץ חיפוש.
לבסוף נעדכן את p_max_tcl להצביע על הצומת הגבוהה ביותר מבחינת שלושת הערכים:
time, course, lecture כאשר המשקל ניתן משמאל לימין.

סיבוכיות זמן:
- חיפוש בעץ course_tree ((O(log(n.
- גישה לNode ומחיקתו ב p_zero_views_classes (O(1.
- הוצאה והכנסה חדשה של הצומת לעץ חיפוש tcl_tree ((O(log(M.
- עדכון p_max_tcl ((O(log(M.
סכ"ה: ((O(log(M.

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

ראשית נבצע חיפוש בעץ course_tree ונמצא את הצומת המתאים עם המפתח courseID.
ניגש למערך של הcourse למקום הclassID. כעת יש 2 מקרים:
- אם קיים עבורו node בp_zero_views_classes אז נחזיר 0.
-אחרת מספר הצפיות גדול שווה ל1, נחזיר את מספר הצפיות ששמור בTCL.

סיבוכיות זמן:
- חיפוש בעץ course_tree ((O(log(n.
- גישה למספר צפיות (O(1.
סכ"ה: ((O(log(n.

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

נעזר בp_max_tcl ובעזרת סיור Inorder **הפוך** נכניס את הרצאות הראשונים לתוך מערך classes ובו זמנית נכניס את הcourse המתאים לכל class במערך courses.
במידה וסיימנו לעבור על העץ tcl_tree ונותרו עוד הרצאות להוסיף אז נעבור לרשימה המקושרת p_zero_views_classes מההתחלה.

סיבוכיות זמן:
- גישה לp_max_tcl (O(1.
- מעבר על m ההרצאות הראשונים (O(m כאשר m = numOfOclasses.
סכ"ה: (o(m.

void Quit(void **DS)

נעבור על שני העצים ועל הרשימה המקושרת ונשחרר כל אחד מהאיברים.
סיבוכיות הזמן היא (O(m+n. (נובע ישירות מסיבוכיות המקום).

סיבוכיות מקום:

ב-course_tree יהיה n צמתים, ובכל צומת יהיוNumOfClasses לכן לכל היותר יהיו n+ M, ומכאן-O(M).
ב-tcl_tree יהיה לכל היותר M צמתים O(M).
ב-רשימה מקושרת יהיה לכל היותר n + M ומכאן O(M).

סכ"ה O(M).