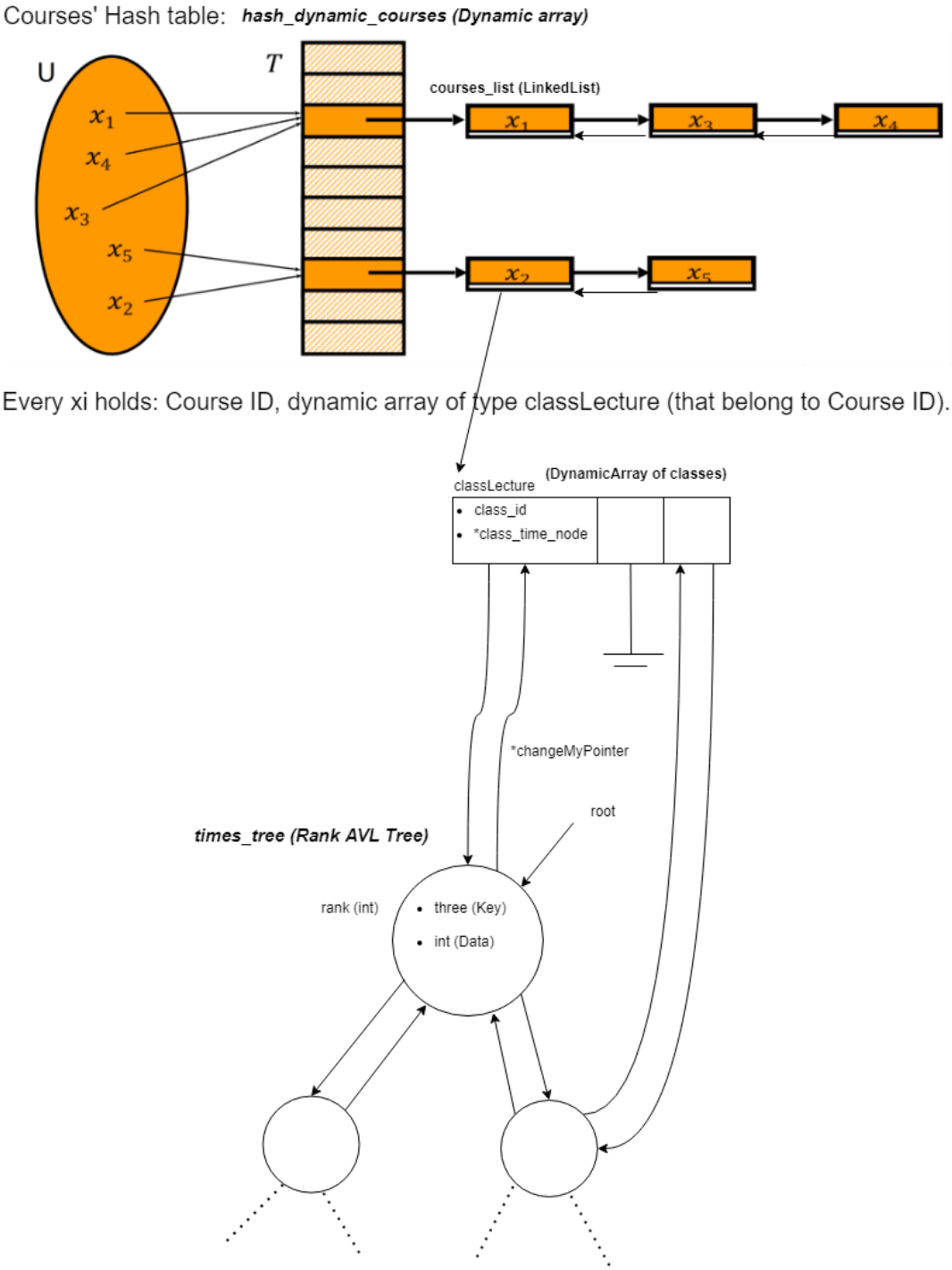
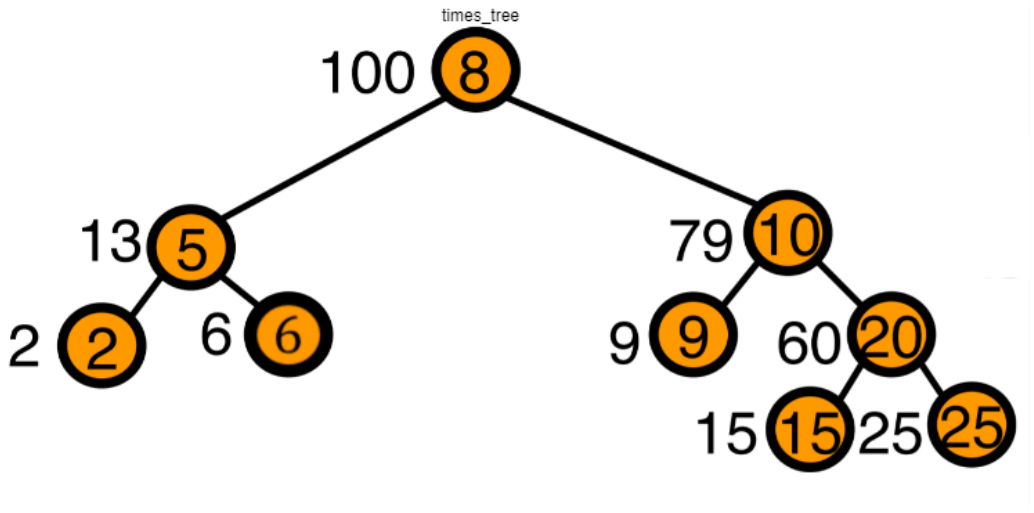
חלק יבש:

תיאור המבנה למימוש התרגיל:



Times Tree: (Rank tree)



When every tree node z  holds a triplet of course ID, class ID, and time as 'X'

and the number of nodes in the sub-tree that X is it's root as 'z'

Note: The tree is sorted by the time watched of the classes(smallest to biggest), course ID(biggest to smallest) as a first tie breaker, and class ID(biggest to smallest) as a second tie breaker.

הסבר על מבני הנתונים:

מבנה בשם three:

המבנה מכיל מידע עבור שיעור מסויים:

1. Class ID – מזהה השיעור.
2. Course ID – מזהה הקורס אליו השיעור שייך.
3. Time – משך זמן הצפייה של השיעור.

מבנה הנתונים תומך באופרטורים (==, <, >) .

סיבוכיות המקום היא $O(1)$.

המבנה Rank AVL Tree:

המפתח בעץ הוא מסוג המבנה three והוא מתואר כשלישייה של זמן צפייה(לא אפס), מזהה הקורס מזהה השיעור. כך שהעדיפות בהשוואה במפתח היא לזמן הצפייה, אחר כך למזהה הקורס (אבל ההשוואה הפוכה) ואחר כך למזהה השיעור.

המידע הוא מטיפוס int והוא לא חשוב.

עץ ה AVL מכיל כמה משתנים חשובים:

1) nodesAmount המכיל את מספר הtreeNodes בעץ, כלומר גודל העץ.
2) Head, שהוא פוינטר לשורש העץ, כך ש Head הוא מטיפוס *treeNode שמכיל בתוכו מצביע לבן השמאלי, מצביע לבן הימני, מצביע להורה. בנוסף בכל treeNode יש מצביע ל- Key ול-

Data של הצומת, בנוסף למשתנים פנימיים לצורך איזון העץ, כמו BF(balance factor) וHeight שהוא הגובה.

3) rank, שדה נוסף השומר את מספר הצמתים בתת העץ של הצומת.
4) ChangeMyPointer, מצביע הפוך ל class_time_node שמוסבר עליו למטה.

הפעולות שהעץ תומך בהם בנוסף לפעולות המוכרות:

1. החזרת הדרגה של צומת v, ע"י אלגוריתם שראינו בהרצאה ($Rank(v)$) בסיבוכיות $O(logn)$, כאשר n הוא מספר הצמתים בעץ.
2. החזרת הצומת עם הדרגה i, ע"י אלגוריתם שראינו בהרצאה ($Select$) בסיבוכיות זמן $O(logn)$, כאשר n הוא מספר הצמתים בעץ.

העץ תומך בבניית עץ ריק בסיבוכיות זמן $O(1)$, ופעולת הכנסה והוצאה של איברים בסיבוכיות זמן $O(logn)$ כאשר n הוא מספר האיברים בעץ כפי שנלמד בתרגול.

סיבוכיות המקום של העץ היא $O(n)$. כאשר n מספר הצמתים.

מבנה הנתונים classLecture:

מבנה שבו שמרנו את מידע עבור הרצאה מסוימת, הוא מכיל:

1) class_id, המספר המזהה של ההרצאה.
2) class_time_node, מצביע לצומת בעץ הדרגות המאוזן times_tree.

אתחול והריסת מבנה הנתונים מתבצעות בסיבוכיות זמן $O(1)$, סיבוכיות המקום היא גם $O(1)$.

מבנה הנתונים LinkedList (רשימה מקושרת דו-כיוונית גנרית):

בנינו רשימה מקושרת דו-כיוונית שמכילה:

1) מצביע לNode שקראנו לו head שמצביע לתחילת הרשימה.
2) מצביע לNode שקראנו לו last שמצביע לסוף הרשימה.
3) size_of_list שמכיל את מספר הNodeים ברשימה.

כל איבר ברשימה הוא Class בשם linkedNode שמכיל:

1) מצביע לאיבר השמאלי.
2) מצביע לאיבר הימני.
3) data של הlinkedNode שהוא מטיפוס גנרי.

סיבוכיות הזמן של הרשימה:

- פעולת אתחול רשימה ריקה $O(1)$
- הכנסת איבר לרשימה לוקחת $O(M)$ כאשר M הוא מספר ההרצאות הכולל במערכת(גודל הרשימה במצב הגרוע ביותר).
- פעולת הוצאה לוקחת $O(1)$ כאשר הפונקציה מקבלת מצביע לNode שצריך למחוק. (אנו מניחים שהאיבר אכן קיים ברשימה.

סיבוכיות המקום של הרשימה הוא $O(M)$ כאשר M הוא מספר האיברים ברשימה(מספר ההרצאות הכולל בכל הקורסים).

מבנה הנתונים courses_list:

היא רשימה מקושרת דו-כיוונית שהשתמשנו בה לצורך שמירת קורסים אליה ע"י פונקציית ערבול בשיטת ה chain hashing.

1) Course ID, מזהה של linkedNode והוא מזהה הקורס.
2) data*, הוא מצביע למערך דינמי שמתאר את השיעורים של קורס זה.

סיבוכיות המקום במקרה הגרוע היא $O(m)$, כאשר m הוא מספר הקורסים הכולל במערכת.

המבנה Hash Table:

טבלת הערבול משתמשת בפונקציית הערבול:

$$h(k) = k(\mathrm{mod}(\mathrm{HashTable.size}))$$

שנלמדה בהרצאה.

כמו כן, כאשר מספר האיברים בטבלת הערבול גדול מדי, במימוש שלנו כאשר מספר האיברים גדול פי 2 מגודל הטבלה, אנו מבצעים פעולת הגדלה של המערך פי 2, ומכניסים את האיברים שבטבלה הישנה לטבלה החדשה.

ובמקרה שמספר האיברים קטן מ ½ מגודל הטבלה, עושים פעולת הקטנה חלקי 2, ומכניסים את האיברים שבטבלה הישנה לטבלה החדשה.

באתחול טבלת הערבול, אנו מאתחלים טבלה בגודל קבוע, כלומר סיבוכיות הזמן של אתחול המבנה היא $O(1)$.

סיבוכיות המקום היא $O(m)$, כאשר m הוא מספר הקורסים הכולל במערכת.

המבנה הראשי: (בשם CoursesManager)

השתמשנו ב 3 מבני נתונים בכדי לממש את המבנה הראשי:

1. Rank AVL Tree. כל צומת מכיל מצביע למבנה three (שמכיל מידע עבור שיעור)
בעץ זה שמרנו את השיעורים של המערכת כך שהמיון מתבצע לפי משך זמן הצפייה של כל שיעור (במקרה של שוויון מיינו לפי מזהה הקורס אבל באופן הפוך ובמקרה של שוויון מיינו לפי מזהה השיעור גם באופן הפוך).
2. Hash Table. טבלת ערבול שבה שמרנו את כל הקורסים שנמצאים במערכת (לא משנה אם עודכן להם זמן צפייה או לא), שכל Node ברשימה מכיל מצביע לצומת בעץ הדרגות שהמפתח של העץ הוא מסוג three.
עבור כל קורס שמרנו את מזהה הקורס (שלפיו מתבצע הערבול), מזהה השיעורים השייכים אליו (בכדי לגשת למידע עבור הקורס בסיבוכיות זמן $O(1)$ בממוצע) ואת זמן הצפייה של השיעור.
3. LinkedList. השתמשנו בה כחלק משיטת ה chain hashing שראינו בהרצאה. ניגשים לראש הרשימה מאיבר בטבלת הערבול, וכל איבר ברשימה מייצג קורס במערכת שהוא מכיל את מזהה הקורס ומערך דינמי לשיעורים השייכים לקורס זה. כל איבר במערך הוא מסוג המבנה classLecture ולכן הוא מכיל מזהה לשיעור ומצביע לצומת בעץ הדרגות בשם class_time_node.

הסבר על הפעולות הנדרשות:

void* Init()

מאתחל מבנה נתונים ריק מסוג CoursesManager.

אתחלנו דרך הקצאה דינמית למבנה ריק, לכן נקבל שסיבוכיות הזמן הנדרשת של הפונקציה זהה לסיבוכיות הזמן לאתחול CoursesManager ולכן זה שווה ל O(1) (סה"כ מקצים זיכרון לעץ דרגות וטבלת ערבול בגודל 2 ולכן אתחולם מתבצע ב O(1)).

סיבוכיות מקום: O(1), כי מקצים בסה"כ O(1) מצביעים, כל מצביע דורש O(1) מקום.

StatusType AddCourse(void *DS, int CourseID)

קודם כל, בדקנו את תקינות הקלט, והחזרנו ערכי שגיאות בסיבוכיות $O(1)$.

בפונקציה הזו אנו מוסיפים הקורס החדש לטבלת הערבול ומוסיפים אותו לרשימה מקושרת במידה וקיימת אחת כזו ואם לא אז יוצרים אחת חדשה שהוא הראש שלה. בכל מקרה מאתחלים מערך בגודל קבוע של 2 אבל עדיין קורס זה מאותחל עם 0 הרצאות. לפי מה שראינו בהרצאה זה לוקח O(α) משוערך, בממוצע על הקלט. ובמימוש שלנו אנו שומרים על α שהוא שווה מספר האלמנטים חלקי גודל המערך(בסדר גודל קטן שווה 0.5 בכך שכל פעם מספר האלמנטים שווה מחצית או רבע מגודל המערך. עדכון הגודל מתבצע לפי מה שראינו בתרגול . לכן הסיבוכיות O(1) משוערך, בממוצע על הקלט. וסיבוכיות מקום נוסף O(1).

StatusType RemoveCourse(void *DS, int CourseID)

קודם כל, בדקנו את תקינות הקלט, והחזרנו ערכי שגיאות בסיבוכיות $O(1)$.

בודקים האם הקורס כבר קיים במערכת ע"י חיפוש בטבלת הערבול של הקורסים בסיבוכיות $O(1)$ משוערך, בממוצע על הקלט. אם לא אז מחזירים שגיאה.

אחרת ניגשים למזהה השיעור לכל שיעור ששייך לקורס זה ו:

- מבצעים הסרה לשיעור מעץ הדרגות המאוזן, בסיבוכיות $O(\log M)$ כי הוא מאוזן, כאשר M הוא מספר השיעורים הכולל בכל הקורסים שיש להם זמן צפייה לא אפס ובסה"כ $O(m\log M)$ כאשר m הוא מספר השיעורים המועברים בקורס זה.
- מבצעים הסרה מטבלת הערבול בסיבוכיות זמן $O(1)$ משוערך לפי אותו טעון מקודם.

StatusType AddClass(void* DS, int courseID, int* classID)

קודם כל, בדקנו את תקינות הקלט, והחזרנו ערכי שגיאות בסיבוכיות $O(1)$.

בודקים האם הקורס כבר קיים במערכת ע"י חיפוש בטבלת הערבול של הקורסים בסיבוכיות $O(1)$ משוערך, בממוצע על הקלט. אם לא אז מחזירים שגיאה.

אחרת ניגשים לקורס שבתוך הרשימה המקושרת ב $O(\alpha)$ משוערך, שזה $O(1)$ משוערך, בממוצע על הקלט, ומוסיפים למערך הדינמי של השיעורים את מזהה השיעור החדש במידה והמערך לא התמלא. אם כן, מגדילים את המערך כפי שראינו בתרגול ואז מוסיפים. חשוב לציין כי אנו שומרים שדה נוסף בכל חולייה של קורס ברשימה המקושרת בשם last_class שמייצג את מזהה השיעור האחרון שהוקצה עבורו מידע(קיים במערכת של קורס זה) ובודקים אם מזהה השיעור שאנו מוסיפים קטן או שווה לשדה זה ומוסיפים במידה וכן. בסוף התהליך מקדמים את last_class ואז שמים אותו בתוך הפרמטר classID.

לפי התרגול הכל מתבצע ב $O(\alpha)$ משוערך, שזה $O(1)$ משוערך, בממוצע על הקלט לפי אותו טיעון מקודם.

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

קודם כל, בדקנו את תקינות הקלט, והחזרנו ערכי שגיאות בסיבוכיות $O(1)$.

בודקים האם הקורס כבר קיים במערכת ע"י חיפוש בטבלת הערבול של הקורסים בסיבוכיות $O(1)$ משוערך, בממוצע על הקלט. אם לא אז מחזירים שגיאה.

אחרת ניגשים לקורס שבתוך הרשימה המקושרת ב $O(\alpha)$ משוערך, שזה $O(1)$ משוערך, בממוצע על הקלט, וניגשים לשיעור במערך הדינמי של השיעורים לפי המזהה שלו ב $O(1)$ במידה והוא מועבר בקורס זה, אחרת מחזירים שגיאה. ואז ע"י המצביע class_time_node ניגשים לעץ הדרגות המאוזן ובמידה ושיעור זה עוד לא נצפה, מוסיפים לשם צומת חדש עם מפתח מסוג three שמכיל את השלישייה (time, courseID, classID), וזה מתבצע ב $O(\log M)$ לפי ההרצאה כאשר M הוא גודל העץ(מספר השיעורים הכולל בכל הקורסים שיש להם זמני צפייה לא אפס). במקרה והוא כן נצפה אז מחפשים אותו תחילה בעץ ומוחקים אותו ב $O(\log M)$ לפי ההרצאה כאשר M הוא גודל העץ, ואז לפי אותו תהליך מקודם מוסיפים צומת שהמפתח שלה היא שלישייה חדשה כאשר הtime ה חדש הוא הקודם פלוס הפרמטר time. תהליך זה מתבצע ב $O(\log M)$. סה"כ ייקח לנו התהליך הכולל של מחיקה והוספה $O(2*\log M)$ וזה שווה ל $O(\log M)$.

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

קודם כל, בדקנו את תקינות הקלט, והחזרנו ערכי שגיאות בסיבוכיות $O(1)$.

בודקים האם הקורס כבר קיים במערכת ע"י חיפוש בטבלת הערבול של הקורסים בסיבוכיות $O(1)$ משוערך, בממוצע על הקלט. אם לא אז מחזירים שגיאה.

אחרת ניגשים לקורס שבתוך הרשימה המקושרת ב $O(\alpha)$ משוערך, שזה $O(1)$ משוערך, בממוצע על הקלט, וניגשים לשיעור במערך הדינמי של השיעורים לפי המזהה שלו ב $O(1)$ במידה והוא מועבר בקורס זה, אחרת מחזירים שגיאה. ואז ע"י המצביע class_time_node ניגשים לעץ הדרגות המאוזן לצומת המבוקש וע"י מפתח צומת זה נקבל את משך זמן הצפייה של שיעור זה מהשדה time ואז נשים אותו לתוך הפרמטר timeViewed.

לפי התרגול הכל מתבצע ב $O(\alpha)$ משוערך, שזה $O(1)$ משוערך, בממוצע על הקלט לפי אותו טיעון מקודם.

StatusType GetIthWatchedClass(void* DS, int i, int* courseID, int* classID)

קודם כל, בדקנו את תקינות הקלט, והחזרנו ערכי שגיאות בסיבוכיות $O(1)$.

בודקים אם יש i או יותר שיעורים עם צפיות(לא אפס) במערכת ע"י בדיקת דרגת שורש העץ שמייצגת את מספר כל השיעורים עם צפיות(לא אפס). אחרת, מחזירים שגיאה.

מבצעים אותו אלגוריתם select(k) שהוצג בתרגול לעץ דרגות אשר עובד ב $O(h)$ שזה $O(\log(M))$ במקרה גרוע, כאשר h הוא גובה העץ ו k שווה ל i ו M הוא מספר השיעורים במערכת בזמן ביצוע הפעולה.

Select(k) – מציאת האיבר בעץ בעל אינדקס k

נתחיל משורש העץ v ונבצע את האלגוריתם הבא:

- אם $w(v \to left) = k - 1$, נחזיר את השורש.
- אם $w(v \to left) > k - 1$, נחפש רקורסיבית בתת העץ השמאלי את האיבר בעל אינדקס k.
- אם $w(v \to left) < k - 1$, נחפש רקורסיבית בתת העץ הימני את האיבר בעל אינדקס $k - w(v \to left) - 1$.

w מייצג את הדרגה (rank בקוד שלנו) והוא מספר הצמתים בתת העץ של הצומת.
סיבוכיות הזמן היא $O(\log M)$ והמקום היא עומק הרקורסיה שהיא $O(\log(M)$.

void Quit(void **DS)

בדקנו את תקינות הקלט בסיבוכיות O(1).

נמיר את DS מ void* -ל courseManager* ע"י static_cast וזה גם קורה בזמן O(1), לאחר מכן מבצעים delete ואז:

נקראים הDestructors של האובייקטים בתוך courseManager שהם:

1) העץ times_tree שמכיל m איברים, כדי לשחרר אותו, צריכים סיבוכיות זמן O(m) כאשר m הוא מספר השיעורים הכולל בכל הקורסים וסיבוכיות מקום O(logm) כאשר m הוא סך כל השיעורים במערכת כלומר עומק הרקורסיה הוא בגובה העץ.
2) הריסת הHashTable בשם hash_dynamic_courses. הפעולה מתבצעת בסיבוכיות זמן O(n) מכיוון שמספר האיברים בטבלה במקרה הגרוע ביותר הוא n והוא מספר הקורסים הכולל, כלומר כל קורס נמצא בתא בודד וברשימה מקושרת בודדת שיש בה מספר מסויים של שיעורים המייוצגים במערך דינמי, כך שבסה"כ גדלי המערכים הדינמיים הוא m שהוא מספר השיעורים הכולל. והריסת כל איבר מתבצעת בסיבוכיות זמן O(1). בנוסף גודל הטבלה בכל רגע הוא O(n) (לפי המימוש שלנו), ולכן הריסת הטבלה מתבצע בסיבוכיות O(n+m).

לאחר מכן, נעדכן את הערך של DS לפי הדרישה לnullptr.

בסה"כ קיבלנו סיבוכיות זמן של O(m)+O(n+m), כלומר O(n+m).

סיבוכיות מקום: O(logM) כפי שהוסבר בסעיף 1 למעלה.

סיבוכיות זמן ומקום:

עמדנו בסיבוכיות זמן של כל הפונקציות כפי שנדרש.

לגבי סיבוכיות מקום: כפי שפורט קודם (כמו בQuit), במבנה הנתונים שבנינו קיימים HashTable, LinkedList, AVL Tree. ננתח את סיבוכיות המקום של כל אחת מהם:

1) עבור כל השיעורים במערכת שמרנו עץ דרגות AVL שמכיל את זמני הצפייה(לא אפס) של השיעורים. סיבוכיות המקום היא $O(\sum_{i=1}^{n} m_i = m)$ כאשר m_i הוא מספר השיעורים בקורס ה i (לכל היותר יש n קורסים). ומכיוון שכל איבר בעץ דורש מקום O(1), נקבל שסיבוכיות המקום היא O(1)*O(m)=O(m).
ולכן סיבוכיות המקום של מבנה מסוג AVL Tree היא O(m).
2) עבור טבלת הערבול, שמספר האיברים בטבלה הוא n, בנוסף גודל הטבלה בכל רגע הוא O(n) (לפי המימוש שלנו), ולכן סיבוכיות המקום של הטבלה הוא O(n).
3) כל תא בטבלת הערבול הוא בעצם רשימה מקושרת שמכילה קורסים ובכל חולייה שמייצגת קורס יש מערך דינמי של השיעורים המועברים בקורס זה בגודל m_i כאשר m_i הוא מספר השיעורים בקורס ה i(הזה). לכן בסה"כ סיבוכיות המקום היא $O(\sum_{i=1}^{n} m_i = m)$.

ולכן בסה"כ קיבלנו שסיבוכיות המקום היא O(n+m) כפי שנדרש.