# מבני נתונים – תרגיל רטוב 2 – יבש

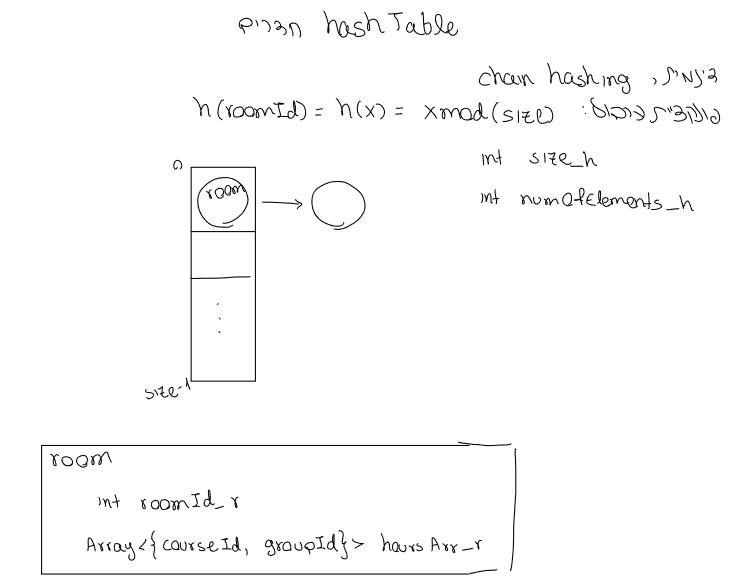
## הוכחת סיבוכיות מקום

### הסבר על מבני הנתונים שהשתמשנו בהם

#### hashtable חדרים

ניצור hashtable דינמית עם chain hashing, שבה נאכסן חדרים. פונקציית הערבול בה נשתמש היא (roomId)mod(hashtable size) (נשמור ונעדכן את גודל הhashtable במהלך הריצה).  
כל חדר הוא למעשה מבנה המכיל את הroomId, ומערך באורך m כשבכל תא i יש זוג של {מספר קורס, מספר קבוצה} אם מתקיימת הרצאה בשעה הi בחדר זה.

מאחר והhashtable דינמי סיבוכיות המקום היא לכל היותר

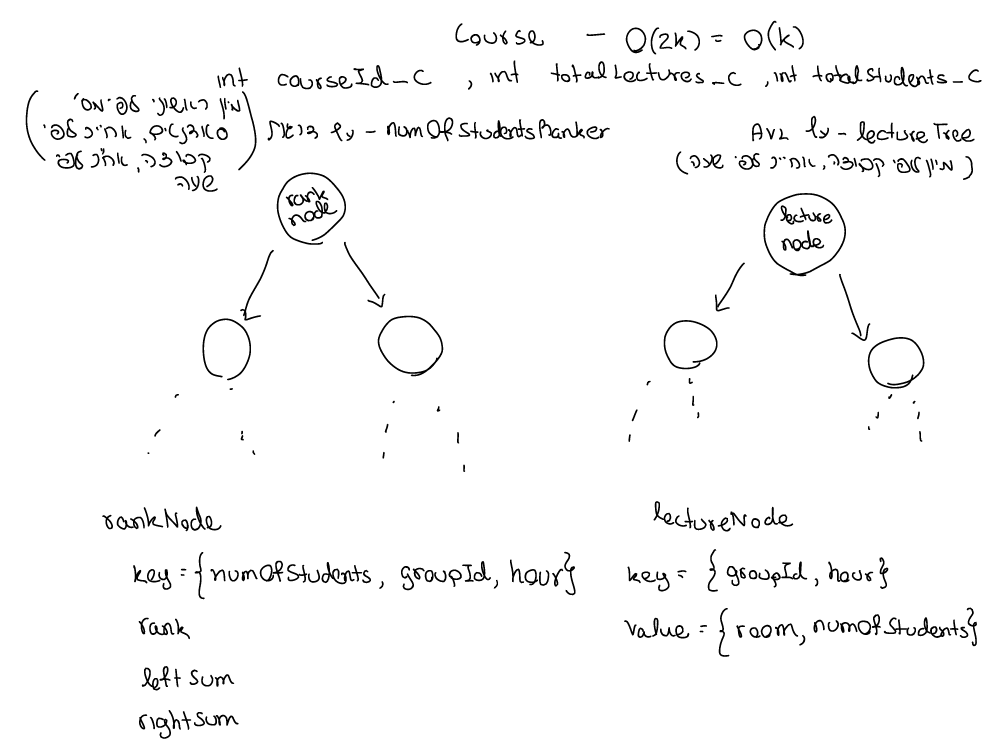


#### קורס – מילון הרצאות + עץ דרגות

ניצור מבנה שמייצג קורס (ואת כל הקורסים המאוחדים תחתיו), המכיל את מספר הקורס אותו הוא מייצג (השורש), מספר ההרצאות הכולל, מספר הסטודנטים הרשומים לכל ההרצאות, ושני עצים:

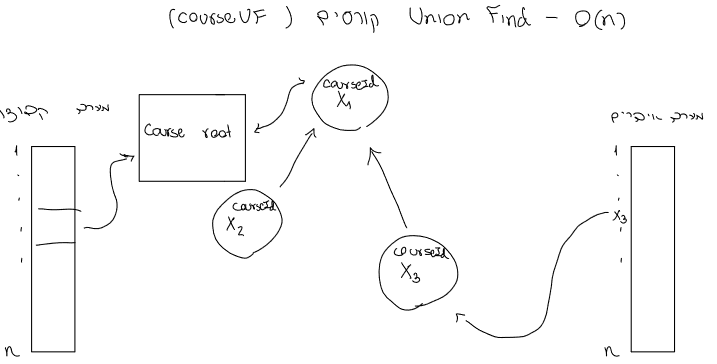
* lectureTree – עץ AVL המכיל את כל ההרצאות המשוייכות לאותו הקורס. העץ ממויין לפי groupId, ואחר כך לפי hour. כל node מחזיק בנוסף לgroupId ולhour את החדר בו מתקיימת ההרצאה ואת מספר הסטודנטים בהרצאה הזו. סה"כ כל העצים הם .
* numOfStudentsRanker – עץ דרגות. ממויין לפי מספר הסטודנטים (מהגדול לקטן), לאחר מכן לפי מספר הקבוצה, ולאחר מכן לפי שעה. כל node מחזיק (בנוסף לkey ולדרגה) את סכום הסטודנטים בכל אחד מתתי העצים שלו. את סכום הסטודנטים אפשר לעדכן תוך כדי ריצה בלי תוספת סיבוכיות, באופן דומה למה שראינו בתרגול.  
  הסיבוכיות גם כאן של כל עצים אלו בכל הקורסים היא .

סה"כ הסיבוכיות היא



#### Union Find קורסים

נאתחל union find בגודל n במימוש הכי מתקדם שלמדנו בתרגול, כאשר האיברים הם מספרי הקורסים, ולכל קבוצה יש מבנה קורס משלה.

סיבוכיות המקום של מבנה זה היא   


סך הכל סיבוכיות המקום של כל המבנה היא:

## הוכחת סיבוכיות זמן

### init

בשלב זה הhashtable והעצים אז אתחולם לוקח O(1). נותר רק לאתחל את הUnion Find מה שייקח . סה"כ - .

### ADd Room

בהנחה והפרמטרים תקינים, ניגש לhashtable לתא המתאים לroomId שקיבלנו. אם לא קיים roomId באותו מספר, נוסיף את החדר החדש. גישה והוספה לhashtable דינמית עם chain hashing וערבול "רגיל" (לא אוניברסלי) היא בממוצע על הקלט באופן משוערך כנדרש.

### delete room

זהה לAddRoom, רק שלפני המחיקה נעבור על כל התאים במערך ( כי m הוא קבוע ) ונוודא שאין הרצאות באותו החדר. מחיקה מhashtable דינמית עם chain hashing וערבול "רגיל" (לא אוניברסלי) היא בממוצע על הקלט באופן משוערך כנדרש.

### add lecture

נמצא את עץ ההרצאות המתאים לקורס המבוקש דרך הunion find - משוערך.

נחפש בlectureTree אם קיימת כבר הרצאה של הקבוצה הזו בשעה המבוקשת. אם לא, נוסיף node לlectureTree לייצג את ההרצאה החדשה, ונעדכן את מספר ההרצאות. בנוסף, נוסיף את מספר הסטודנטים הרשומים להרצאה למספר הסטודנטים הכולל, וניגש לnumOfStudentsRanker ונוסיף node חדש. סה"כ שתי הגישות לוקחות

*בנוסף, ניגש לתא המתאים לחדר בhashtable ונוסיף את ההרצאה החדשה למערך שלו.* גישה זו תיקח בממוצע על הקלט במשוערך.

לכן סך הכל סיבוכיות הזמן של הפונקציה היא  *בממוצע על הקלט באופן משוערך.*

### delete lecture

ניגש לתא המתאים לחדר בhashtable, נחפש במערך האם קיימת הרצאה בשעה המבוקשת - בממוצע על הקלט.  
  
אם כן, ניגש למספר הקורס ששמור במערך של החדר בunion find - משוערך.  
  
יש לנו את מספר הקבוצה (מהמערך של החדר) ואת השעה, אז ניגש לlectureTree ונמצא את מספר הסטודנטים הרשומים.   
כעת יש לנו את כל מה שצריך כדי לגשת לnumOfStudentsRanker ולמחוק את הnode המתאים. נחסיר ממספר הסטודנטים הכולל את מספר הסטודנטים שהיו רשומים להרצאה.  
בנוסף נמחק מlectureTree את הnode התואם להרצאה ונחסיר 1 ממספר ההרצאות הכולל. – סה"כ גישות לעצים

ננקה את התא המתאים במערך של החדר בhashtable- בממוצע על הקלט במשוערך.

סך הכל סיבוכיות הזמן של הפונקציה היא  *בממוצע על הקלט באופן משוערך.*

### merge courses

בהנחה והפרמטרים תקינים, ניגש לשני הקורסים שרוצים לאחד. נאחד אותם בפעולת union – משוערך.

כעת, נאחד בין העצים שלהם (נשמור את העצים הישנים כtemp)  
ראשית נאחד את lectureTree, כפי שלמדנו לאחד עצי avl. במהלך פעולת האיחוד אם מצאנו שתי הרצאות של אותה קבוצה באותה שעה (key זהה), נעצור את פעולת האיחוד ונחזיר כישלון. איחוד עצים מאוזנים –   
נאחד את numOfStudentsRanker – לא אמור להיות מצב של keys זהים כי בדקנו כבר בlecture tree. איחוד עצים מאוזנים -

את העצים המאוחדים נשים בקורס שמהווה את השורש של הקבוצה בunion find ונמחק את העצים הישנים –   
  
נאחד את הhashtables ע"י הוספת כל הערכים בhashtable מהקורס שאינו שורש לקורס שהוא שורש. ומחיקת הhashtable הישן. פעולה זו תיקח בממוצע על הקלט באופן משוערך.

בנוסף נעדכן את מספר ההרצאות ומספר הסטודנטים הכולל בשורש להיות הסכום של שני הקורסים.

סך הכל סיבוכיות הזמן של הפונקציה היא  *בממוצע על הקלט באופן משוערך.*

### competition

ניגש לשני הקורסים בunion find - משוערך.

לכל קורס, נמצא בnumOfStudentsRanker את הnode בדרגה הnumGroups-1, ונסכום את מספר הסטודנטים שלו עם מספר הסטודנטים בתת העץ השמאלי שלו (מספר הסטודנטים ממויין מהגדול לקטן, אז תת העץ השמאלי יכיל את כל הnodes בעלי דרגה נמוכה יותר( -

כעת נותר רק להשוות בין הסכומים ולהחזיר את הקורס המנצח.

סך הכל סיבוכיות הזמן של הפונקציה היא  *משוערך.*

### Get Average students in course

ניגש לקורס בunion find - משוערך.

כעת כל שנותר הוא לחלק את סך כל הסטודנטים בסך כל ההרצאות כדי לקבל את הממוצע.

סך הכל סיבוכיות הזמן של הפונקציה היא  *משוערך.*

### quit

שחרור המבנה גורר מעבר אחד אחד על כל הnodes בעצים - , כל הקורסים בunion find - , וכל התאים בhashtable החדרים - . סה"כ