**חלק יבש – רטוב 2**

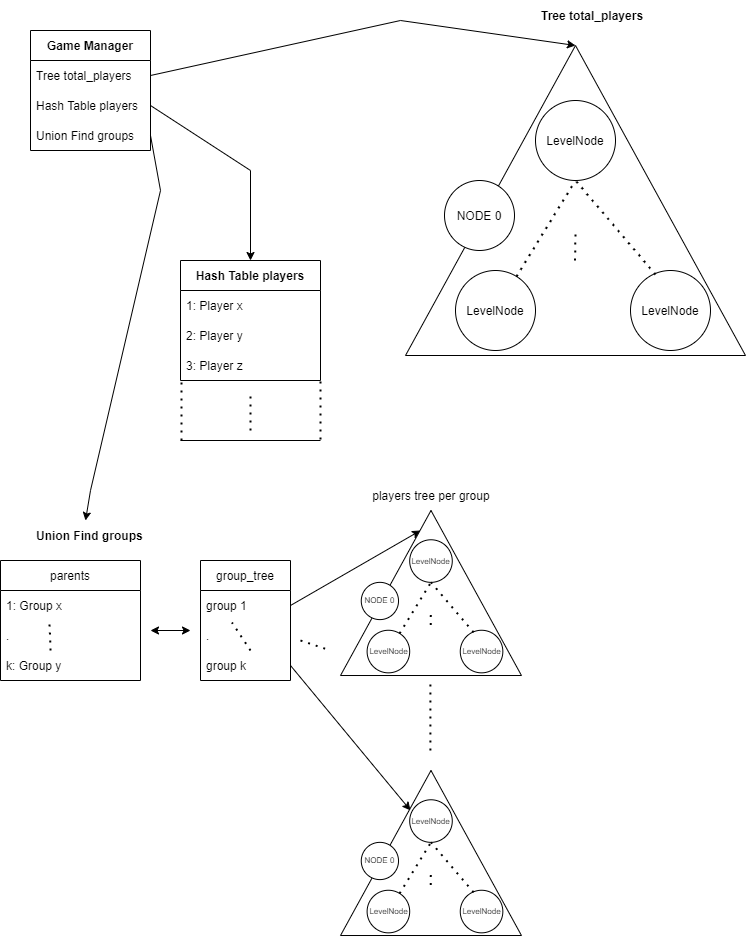
*מבנה הנתונים שלנו יכיל את המידע הבא:*

1. *K – מספר הקבוצות במשחק*
2. *Scale – הSCORE המקסימאלי שהוחלט*
3. *Groups\_UF – מבנה נתוניםUnionFind כפי שנלמד בהרצאות אשר יחזיק את K הקבוצות שבמשחק.*
   1. *גודל המבנה*
   2. *מערך "אבות" – מערך המייצג עבור כל קבוצה מי האב שלה (המספר בתא ה-i מציין את מספר הקבוצה של האב של הקבוצה ה-1+i.*
   3. *מערך גודל הקבוצה – מערך המחזיק עבור כל קבוצה מהו גודלה (לצורך מיזוג הקבוצה הקטנה לתוך הגדולה)*
   4. *מערך העצים של כל קבוצה – מערך מצביעים המצביע על העץ Levels של כל קבוצה.*
      1. *כל עץ זהה לעץ Total\_player\_levels\_tree, פשוט ברמת הקבוצה.*
4. *Players\_HT – טבלת ערבול בגודל דינאמי כפי שנלמדה בהרצאה עבור אלמנטים מסוג Players.*
   1. *מספר השחקנים בטבלה הנוכחי*
   2. *גודל הטבלה הנוכחי*
   3. *מערך שכל תא בו מחזיק רשימה מקושרת של שחקנים אשר פונקציית ה-HASH חישבה שהמפתח שלהם הוא אינדקס התא.*
5. *Player -*
   1. *מזהה השחקן*
   2. *ניקוד השחקן*
   3. *רמת השחקן*
   4. *מזהה הקבוצה*
6. *Total\_player\_levels\_tree – עץ דרגות עם מידע נוסף המייצג את מספר השחקנים בכל רמה לפי פילוח SCORE*
   1. *zero – מצביע ל-levelNode של הרמה ה-0 שאינה יושבת בתוך העץ עצמו על מנת לאפשר הכנסת שחקנים חדשים (אשר מוכנסים ברמה 0) בסיבוכיות זמן .*
   2. *Head – מצביע לראש העץ אשר מכיל את כל הרמות שאינן 0.*
   3. *גודל העץ – מחזיק כמה levelNode יש בעץ (לא כולל את רמה 0)*
   4. *LevelNode –* 
      1. *Level – איזה רמה הNODE מייצג*
      2. *Players – מערך במכיל את מספר השחקנים בתת העץ של אותו NODE לפי פילוח SCORE*
      3. *Sum\_Players – תא המכיל את סכום ה-Levels בתת העץ שלו (הסכום הוא חיבור ההכפלה בין מספר הרמה המיוצג כפול מספר השחקנים מאותה רמה בתת העץ).*

*אבחנות:*

1. *כיוון שעץ ה-Levels מחזיק רמות שעדיין מכילות שחקנים (לא ריקות), לכן מספר האיברים בעץ זה לא יכול להיות גדול מ-N (מספר השחקנים שבקבוצה במקרה של העצים שבתוך ה-UF או מספר השחקנים הכולל במשחק במקרה של עץ ה-Levels הכולל).*
2. *גודל כל עץ קבוצה חסום ע"י מספר השחקנים באותה קבוצה לפי אבחנה 1.  
   לכן כיוון שסכום השחקנים בכל הקבוצות הוא מספר השחקנים במשחק גודל סכום ה-LevelNodes של כלל העצים בתוך ה-UF חסום ע"י N (מספר השחקנים במשחק).*

*תרשים:*

**

*מימושים לפונקציות:*

1. void\* init(int k, int scale)
   * *בדיקה שהקלט תקין, במידה ולא החזרת NULL -*
   * *אתחול המשתנה Scale לערך שהתקבל מהמשתמש -*
   * *אתחול המשתנה K לערך שהתקבל מהמשתמש -*
   * *אתחול מבנה UF של הקבוצות המכיל K קבוצות ו-K עצים ריקים -*
   * *אתחול טבלת ערבול בגודל קבוע*

***סיבוכיות הפונקציה היא במקרה הגרוע***

1. StatusType mergeGroups(void \*DS, int GroupID1, int GroupID2)
   * *בדיקה שהקלט תקין, במידה ולא החזרת INPUT\_INVALID -*
   * *ביצוע פעולת איחוד בין שתי הקבוצות אשר התקבלו בקלט, האיחוד כולל ביצוע פעולת איחוד 2 הקבוצות במבנה הUF () וביצוע איחוד בין שני עצי הLEVEL של הקבוצות כפי שנלמד בתרגול, לפי האבחנה איחוד בין שני עצי הLEVELS של הקבוצות חסום ע"י מספר השחקנים בקבוצה המאוחדת (n) -*

***סיבוכיות הפונקציה היא***  ***משוערך, בממוצע על הקלט לפי ההוכחה בהמשך***

1. StatusType addPlayer(void \*DS, int PlayerID, int GroupID, int score)
   * *בדיקה שהקלט תקין, במידה ולא החזרת INPUT\_INVALID -*
   * *בדיקה האם השחקן כבר קיים, במידה וכן יוחזר הערך FAILURE - בממוצע על הקלט*
   * *הכנסת השחקן לטבלת ערבול של השחקנים - בממוצע על הקלט*
   * *הוספת שחקן ברמה 0 לעץ השחקנים הכולל תוך עדכון כלל הצמתים הרלוונטיים במסלול הכנסתו לטובת שמירת עדכניות עץ הדרגות (כיוון שרמה 0 מופרדת משאר העץ פעולות עליה הן בסיבוכיות קבועה) -*
   * *מציאת עץ הקבוצה אליה שייך השחקן ב-UF ( משוערך) והוספת השחקן לרמה 0 של עץ הקבוצה בדומה לפעולה הקודמת - משוערך.*

***סיבוכיות הפונקציה היא משוערך, בממוצע על הקלט לפי ההוכחה בהמשך***

1. StatusType removePlayer(void \*DS, int PlayerID)
   * *בדיקה שהקלט תקין, במידה ולא החזרת INPUT\_INVALID -*
   * *בדיקה האם השחקן קיים, במידה ולא יוחזר הערך FAILURE - בממוצע על הקלט*
   * *הסרת השחקן מהרמה הנוכחית שלו בעץ השחקנים הכולל ועדכון המידע הנוסף בצמתים הרלוונטיים בעץ בעת בהסרתו לטובת שמירת עדכניות עץ הדרגות, במידה והוא השחקן האחרון ברמה זו יש להסיר את NODE הרמה שלו -*
   * *מציאת עץ הקבוצה אליה שייך השחקן () הסרת השחקן מהרמה הנוכחית שלו בעץ הקבוצה בדומה לפעולה הקודמת -*
   * *הסרת השחקן מטבלת הערבול של השחקנים - בממוצע על הקלט*

***סיבוכיות הפונקציה היא משוערך, בממוצע על הקלט לפי ההוכחה בהמשך***

1. StatusType increasePlayerIDLevel(void \*DS, int PlayerID, int LevelIncrease)
   * *בדיקה שהקלט תקין, במידה ולא החזרת INPUT\_INVALID -*
   * *בדיקה האם השחקן קיים, במידה ולא יוחזר הערך FAILURE - בממוצע על הקלט*
   * *הסרת השחקן מעץ השחקנים הכולל ומעץ השחקנים של הקבוצה שלו -*
   * *עדכון ערך הרמה (level) של השחקן בטבלת הערבול - בממוצע על הקלט*
   * *הכנסת השחקן עם ערך הרמה החדש לעץ השחקנים הכולל ולעץ השחקנים של הקבוצה שלו, אם הרמה שאליו הוא הועלה כבר קיימת פשוט יש לעדכן את התאים הרלוונטיים ברמה, אחרת יש להכניס NODE רמה חדש בעץ -*

***סיבוכיות הפונקציה היא משוערך, בממוצע על הקלט לפי ההוכחה בהמשך***

*סעיף בונוס:*ניתן היה לממש את הפעולה הנ"ל בסיבוכיות של ***משוערך*** *(לא ממוצע על הקלט) בדרך הבאה:*

*במקום לממש את טבלת הערבול כך שכל תא בה מכיל רשימה מקושרת (במקרה זה סיבוכיות גישה לאחד האיברים בטבלה היא במקרה הממוצע ו**במקרה הגרוע)   
נממש את הטבלת ערבול כך שבכל תא בה יהיה* ***עץ AVL*** *אשר יחזיק את כלל האיברים המתאימים לאותו תא ספציפי.  
כעת סיבוכיות גישה לאחד האיברים בטבלה היא במקרה הממוצע ו**במקרה הגרוע. מכך נובע כי סיבוכיות הפונקציה כעת היא:*

* + *בדיקה שהקלט תקין, במידה ולא החזרת INPUT\_INVALID -*
  + *בדיקה האם השחקן קיים, במידה ולא יוחזר הערך FAILURE - במקרה הגרוע*
  + *הסרת השחקן מעץ השחקנים הכולל ומעץ השחקנים של הקבוצה שלו -*
  + *עדכון ערך הרמה (level) של השחקן בטבלת הערבול - במקרה הגרוע*
  + *הכנסת השחקן עם ערך הרמה החדש לעץ השחקנים הכולל ולעץ השחקנים של הקבוצה שלו, אם הרמה שאליו הוא הועלה כבר קיימת פשוט יש לעדכן את התאים הרלוונטיים ברמה, אחרת יש להכניס NODE רמה חדש בעץ -*

***סיבוכיות הפונקציה היא משוערך כנדרש***

1. StatusType changePlayerIDScore(void \*DS, int PlayerID, int NewScore)
   * *בדיקה שהקלט תקין, במידה ולא החזרת INPUT\_INVALID -*
   * *בדיקה האם השחקן קיים, במידה ולא יוחזר הערך FAILURE - בממוצע על הקלט*
   * *הסרת השחקן מעץ השחקנים הכולל ומעץ השחקנים של הקבוצה שלו -*
   * *עדכון ערך התוצאה (score) של השחקן בטבלת הערבול - בממוצע על הקלט*
   * *הכנסת השחקן עם ערך הרמה החדש לעץ השחקנים הכולל ולעץ השחקנים של הקבוצה שלו -*

***סיבוכיות הפונקציה היא משוערך, בממוצע על הקלט לפי ההוכחה בהמשך***

1. StatusType getPercentOfPlayersWithScoreInBounds (void \*DS, int GroupID, int score, int lowerLevel, int higherLevel, double \* players)
   * *בדיקה שהקלט תקין, במידה ולא החזרת INPUT\_INVALID -*
   * אם GroupID == 0 נבצע את החישוב על עץ השחקנים הכולל:
     + נמצא את הרמה המקסימאלית המקיימת שהיא קטנה שווה מ-higherLevel (בדומה לאלגוריתם SELECT אשר נלמד בתרגול) -
     + נמצא את הרמה המקסימאלית המקיימת שהיא קטנה מ- lowerLevel(בדומה לאלגוריתם SELECT אשר נלמד בתרגול) -
     + נחשב את מספר השחקנים בעלי ערך score הנמצאים בטווח higherLevel-lowerLevel כולל על ידי הפעלת פונק' הדירוג עבור כלל השחקנים בטווח המבוקש ולא בSCORE ספציפי -
     + במידה ואין שחקנים כלל בטווח המבוקש נחזיר FAILURE
     + נכניס את הערך המבוקש לתא בזיכרון של המצביע players
   * אחרת, נמצא את עץ הקבוצה הנדרשת ע"י חיפוש בUF ונבצע את החישוב על העץ של הקבוצה:
     + מציאת עץ הקבוצה -
     + נמצא את הרמה המקסימאלית המקיימת שהיא קטנה שווה מ- higherLevel(בדומה לאלגוריתם SELECT אשר נלמד בתרגול) -
     + נמצא את הרמה המקסימאלית המקיימת שהיא קטנה מ-lowerLevel (בדומה לאלגוריתם SELECT אשר נלמד בתרגול) -
     + נחשב את מספר השחקנים בעלי ערך score הנמצאים בטווח higherLevel-lowerLevel כולל על ידי הפעלת פונק' הדירוג עבור כלל השחקנים בטווח המבוקש ולא בSCORE ספציפי -
     + במידה ואין שחקנים כלל בטווח המבוקש נחזיר FAILURE
     + נכניס את הערך המבוקש לתא בזיכרון של המצביע players

***סיבוכיות הפונקציה היא משוערך לפי ההוכחה בהמשך***

1. StatusType averageHighestPlayerLevelByGroup(void \*DS, int GroupID, int m, double \* avgLevel)
   * *בדיקה שהקלט תקין, במידה ולא החזרת INPUT\_INVALID –*
   * *בדיקה כי יש לפחות m שחקנים בעץ הרלוונטי, במידה ולא החזרת FAILURE –*
   * *נבצע את הפעולות הבאות בעץ שחקנים הכולל או בעץ שחקנים של הקבוצה עצמה לפי הערך שהתקבל ב-GroupID:*
     + *נמצא את ה-node בעץ שמקיים שהחל מהרמה אותו הוא מייצג (נסמנה L) יש לפחות m שחקנים ברמה הזו וברמות גבוהות יותר, ואין אף רמה גבוהה יותר מ-L שמקיימת את התנאי הזה -*
     + *נחשב את הרמה הממוצעת של m השחקנים ברמות הגבוהות ביותר, באמצעות לקיחת כל השחקנים מעל רמה L וחלק מהשחקנים מרמה L (כדי שיהיו m שחקנים בסך הכל) באמצעות חישוב הדרגה של הצומת L לפי השדה total\_sum וחלוקה במספר השחקנים (m) -*

***סיבוכיות הפונקציה היא משוערך לפי ההוכחה בהמשך***

1. StatusType getPlayersBound(void \*DS, int GroupID, int score, int m, int \* LowerBoundPlayers, int \* HigherBoundPlayers)
   * *בדיקה שהקלט תקין, במידה ולא החזרת INPUT\_INVALID –*
   * *בדיקה כי יש לפחות m שחקנים בעץ הרלוונטי, במידה ולא החזרת FAILURE –*
   * *נבצע את הפעולות הבאות בעץ שחקנים הכולל או בעץ שחקנים של הקבוצה עצמה לפי הערך שהתקבל ב-GroupID:*
     + *נמצא את ה-node בעץ שמקיים שהחל מהרמה אותו הוא מייצג (נסמנה L) יש לפחות m שחקנים ברמה הזו וברמות גבוהות יותר, ואין אף רמה גבוהה יותר מ-L שמקיימת את התנאי הזה* (בדומה לאלגוריתם SELECT אשר נלמד בתרגול)  *-*
     + *כל השחקנים בעץ פחות השחקנים ברמה L הם שחקנים שחייבים לקחת לפי החישוב המוצג בשאלה ולכן נעדכן את LowerBoundPlayers בהתאם –*
     + *נחשב את כמות השחקנים עם ניקוד score ואת כמות השחקנים הכוללת ברמה L כדי לחשב את LowerBoundPlayers ו-HighrBoundLevel –*
     + *לפי החישוב בסעיף הקודם נחשב את מספר השחקנים שאפשר לקחת עם ניקוד score מתוך רמה L ונוסיף אותם לכמות השחקנים שחייבים לקחת מתוך הרמות הגבוהות מ-L ונשים את טווח התוצאות במשתנים LowerBoundPlayers ו-HighrBoundLevel –*

***סיבוכיות הפונקציה היא משוערך לפי ההוכחה*** ***בהמשך***

1. void Quit(void \*\*DS):
   * *הגדרת כלל המצביעים של מבנה הנתונים לNULLPTR וכך לגרום לקריאה לפונק' הריסה של המבנים העצמאים.*
   * *מחיקת מבנה UF וכלל העצים שתחתיו, כפי שנובע מהוכחת סיבוכיות המקום במבנה לכל היותר N+K אובייקטים הנדרשים למחיקה -* ***O(n+k)***
   * *מחיקת טבלת הערבול, מכילה N אובייקטים של שחקנים -* ***O(n)***
   * *מחיקת עץ הרמות של כלל השחקנים -* ***O(n)***

***סיבוכיות הזמן של הפונקציה היא במקרה הגרוע ביותר O(n+k)***

***הוכחת סיבוכיות זמן:***

*נוכיח את שני החסמים עבור סיבוכיות הזמן המשוערכת ביחד, כלומר נוכיח כי עבור רצף פעולות של:*

1. *פעולות addPlayer*
2. *פעולות removePlayer*
3. *פעולות increasePlayerIDLevel*
4. *פעולות changePlayerIDScore*
5. *פעולות mergeGroups*
6. *פעולות getPercentOfPlayersWithScoreInBounds*
7. *פעולות averageHighestPlayerLevelByGroup*
8. *פעולות getPlayersBound*

*כאשר*

*נתחיל במספר אבחנות:*

* + *כמות הפעולות removePlayer המבוצעות במלואן קטנה שווה לכמות פעולות addPlayer כיוון שלא ניתן להסיר שחקן שלא נוסיף למשחק ולכן .*
  + *אנו משתמשים ב-Hash Table עבור אחסון השחקנים במשחק. לטובת כך, אנו משתמשים בשיטת Rehashing שהוצגה בהרצאה ודואגים כי פקטור העומס מקיים . בנוסף נשים לב כי הפעולות שעובדות עם ה-HashTable הן:  
    addPlayer, removePlayer, increasePlayerIDLevel, changePlayerIDScore, ובכולן מתבצעות פעולות insert, remove על ה-Hash Table ולכן סיבוכיות רצף הפעולות הזו היא O(1) משוערך בממוצע על הקלט (כולל הרחבה וצמצום המערך) כפי שהוכחנו בהרצאה.*
  + *במהלך השימוש ב-Union Find אנו מבצעים כיווץ מסלולים, כלומר בכל פעולת Find אנו מחברים את כל הקבוצות אל השורש של העץ במייצג קבוצות שאוחדו ולכן סיבוכיות הזמן של ביצוע הפעולות שתלויות ב-k כמות הקבוצות מתבצעות רק ב-Union Find ולכן הזמן המשוערך עבור ביצוע הפעולות הללו (כמות הקבוצות ב-Union Find היא ככמות הקבוצות – k) הוא שכן כל הפעולות שלנו שמתמשות ב-Union Find הן פעולות Find, Merge ולכן הסיבוכיות המשוערכת של רצף פעולות כזה הוא כפי שהוכחנו בהרצאה.*
  + *כפי שפירטנו בפרק הקודם, במהלך ביצוע הפעולות removePlayer, increasePlayerIDLevel, changePlayerIDScore אנו מבצעים פעולות הסרה והכנסה לעצי AVL מאוזנים – לעץ AVL שמכיל את כל השחקנים במשחק ולעץ השחקנים של הקבוצה הרלוונטית שמקיים כי הגודל שלו חסום ע"י n. אנו מבצעים מספר סופי של פעולות כאלו ולכן הסיבוכיות של סדרת פעולות כמתואר היא במקרה הגרוע ולכן גם במקרה המשוערך לרצף פעולות כזה. נדגיש כי במהלך ביצוע פעולת addPlayerאנחנו מכניסים את השחקן עם רמה 0 לצומת 0 בעץ שהיא מנותקת משאר העץ ולכן במקרה הזה, ביצוע פעולה זו הוא O(1) עבור שני העץ (עץ השחקן הכללי ועץ השחקנים של הקבוצה).*

*ראשית, נוכיח את סיבוכיות החלק שנוגע למספר הקבוצות:*

*ולכן הוכחנו כנדרש כי לכל רצף של t פעולות (1)-(8) החלק שתלוי במספר הקבוצות הוא בסיבוכיות משוערכת של .*

*שנית, נוכיח את סיבוכיות החלק שנוגע למספר השחקנים:*

*כאשר נזכיר כי הוא כמות פעולות addPlayer ונסמן כמות פעולות removePlayer, increasePlayerIDLevel, changePlayerIDScore*

*ולכן הוכחנו כי כל רצף כזה של פעולות הוא בסיבוכיות משוערכת של עבור הפעולות removePlayer, increasePlayerIDLevel, changePlayerIDScore ו-1 עבור addPlayer (החלק שתלוי במספר השחקנים בלבד)*

*ולכן הוכחנו כי הסיבוכיות המשוערכת של הפעולות שלנו היא כנדרש.*

***סיבוכיות מקום:***

* + בכל זמן נתון מבנה הנתונים מחזיק מקום בזיכרון באופן הבא (K מספר הקבוצות, N *מספר השחקנים במשחק)*:
    - *K – משתנה מספרי* ***O(1)***
    - *Scale – משתנה מספרי* ***O(1)***
    - מבנה UF של קבוצות *–* 
      * *מערך "אבות" – מערך בגודל K*
      * *מערך גודל הקבוצה – מערך בגודל K*
      * *מערך העצים של כל קבוצה – מערך בגודל K שבו כל תא מצביע על עץ.*
        1. *גודל כל עץ קבוצה חסום ע"י מספר השחקנים באותה קבוצה לפי האבחנה.  
           כיוון שסכום השחקנים בכל הקבוצות הוא מספר השחקנים במשחק גודל הזיכרון שהעצים תופסים חסום ע"י N.*

*לכן מבנה זה הוא בסיבוכיות מקום של* ***O(n+k)***

* + - טבלת ערבול שחקנים – הטבלה דינאמית ולכן מחזיקה במקרה הגרוע ביותר כמעט n4 (רגע לפני הקטנה)

*לכן מבנה זה הוא בסיבוכיות מקום של* ***O(n)***

* + - עץ Levels של כל המשחק – *גודל העץ חסום ע"י מספר השחקנים במשחק לפי האבחנה.*

*לכן מבנה זה הוא בסיבוכיות מקום של* ***O(n)***

***סיבוכיות המקום של כלל המבנה היא במקרה הגרוע ביותר O(n+k)***