דף שער

מבני נתונים 1

234218

תרגיל רטוב

2

מספר

הוגש ע"י:

שזא עבדאללה	323032474
שם	מספר זהות
מרים ארשיד	322309006
שם	מספר זהות

ציון:

לפני בונוס הדפסה:

כולל בונוס הדפסה:

נא להחזיר לתא מס':

<<מבנה נתונים- רטוב 2 :חלק יבש>>

הסבר כללי על מבני הנתונים שלנו:

-השתמשנו במבני הנתונים בעצי דרגות מיוחדים AVL , union find ו- hash table :

מחלקת **Players Manager**:

- **allplayersTable :**

השתמשנו ב **Chained hash Table** דינמי על מנת לשמור את השחקנים בו. כאשר אנו שומרים לכל שחקן :

  - Player ID
  - Group ID
  - Level
  - Score

- **Alliances :**

השתמשנו ב Union Find על מנת לאחד ולחפש את K הקבוצות.

- **AllPlayersGroup :**

זהו אובייקט מסוג Group שמכיל :

  - Int* zero
  - AVLTree* players

מערך zero הוא מערך בגודל 201 כאשר בתא $1 \le i \le 200$ נשמור מספר השחקנים בעלי LEVEL=0 ו- $score = i$ ובתא אפס שומרים מספר השחקנים בעלי LEVEL=0 הכולל.

בנוסף players הוא עץ דרגות מאוזן ממוין לפי LEVELS כאשר בכל צומת ישנן:

  - level
  - average
  - int* score[201]
  - int* sumScores[201]

- ב $average$ אנו שומרים את הסכום המשוקלל של LEVELS בתת העץ (כלומר הוא כמעט הממוצע אבל לא מחולק במספר השחקנים).
- score הוא מערך בגודל 201 כאשר בתא $1 \le i \le 200$ נשמור מספר השחקנים בעלי $score = i$ ובתא אפס שומרים מספר השחקנים בעלי אותו LEVEL הכולל.
- sumScore הוא מערך בגודל 201 כאשר בתא $1 \le i \le 200$ נשמור מספר השחקנים בעלי $score = i$ בתת העץ ובתא אפס שומרים מספר השחקנים הכולל בתת העץ.

- **AllGroups:**

מערך בגודל K כגודל הקבוצות מסוג Group כאשר בכל תא $0 \le i \le k-1$ אנו נשמור את מערך ה-zero ועץ ה – players המתאים לקבוצה $i+1$.

הערה:

- הסבר למחלקת GROUP מסופק כאשר הסברנו על $allPlayersGroup$.
- בעץ הדרגות אנו משמרים על נכונות השדות $average$ ו - $sumScores$ בעת הגלגולים כי השתמשנו בנוסחה המסופקת בתרגול עם שינוי קטן : (עבור גלגול LL כאשר B הצומת שהופר בא האיזון ו A בנה השמאלי)

$$avg(B) = avg(A_{Right}) + avg(B_{Right}) + B.score[0] * B.level$$

$$avg(A) = avg(B) + avg(A_{Left}) + A.score[0] * A.level$$

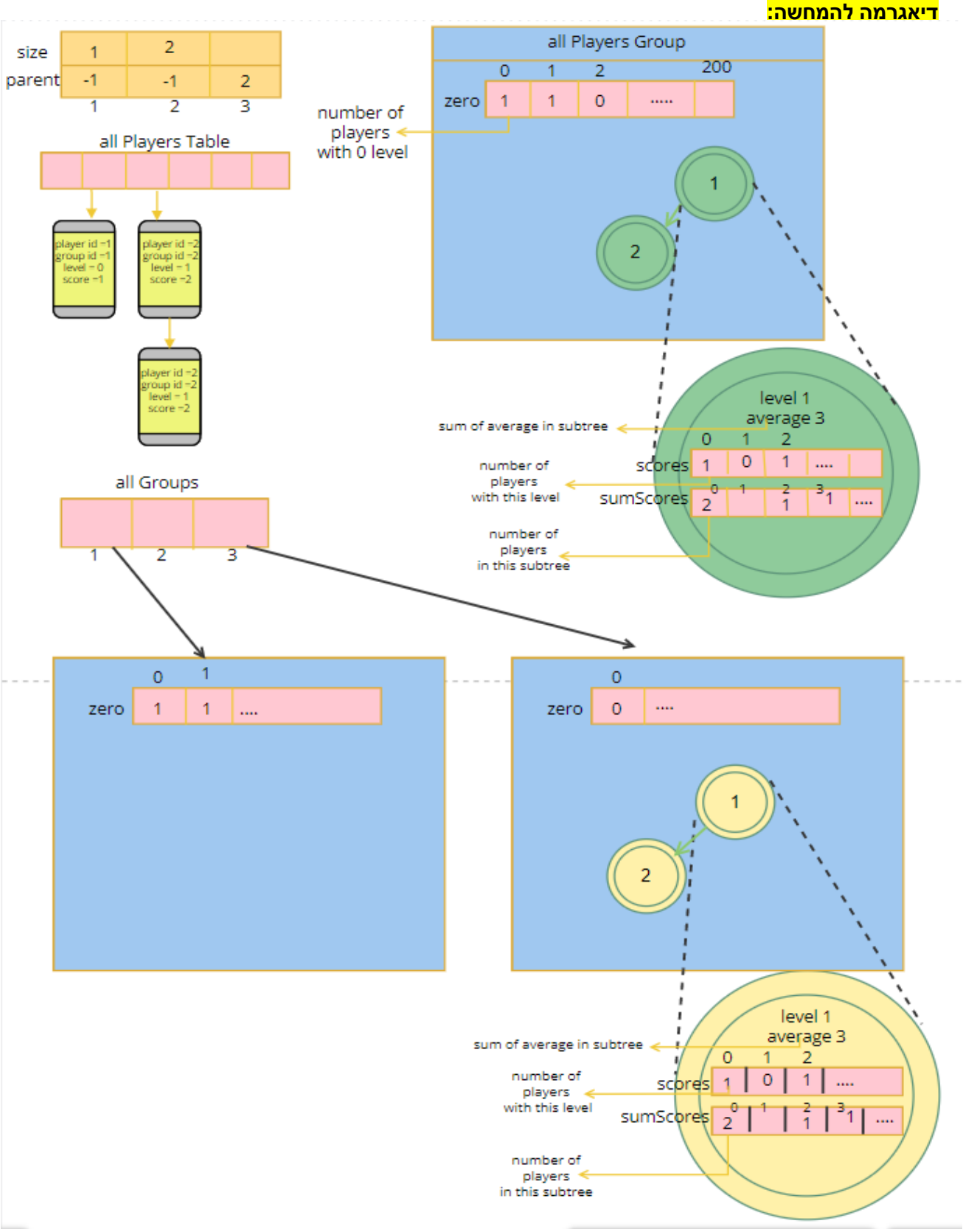
$$B.sumScore[i] = A_{Right}.sumScore[i] + B_{Right}.sumScore[i] + B.score[i]$$

$$A.sumScore[i] = A_{Left}.sumScore[i] + B.sumScore[i] + A.score[i]$$

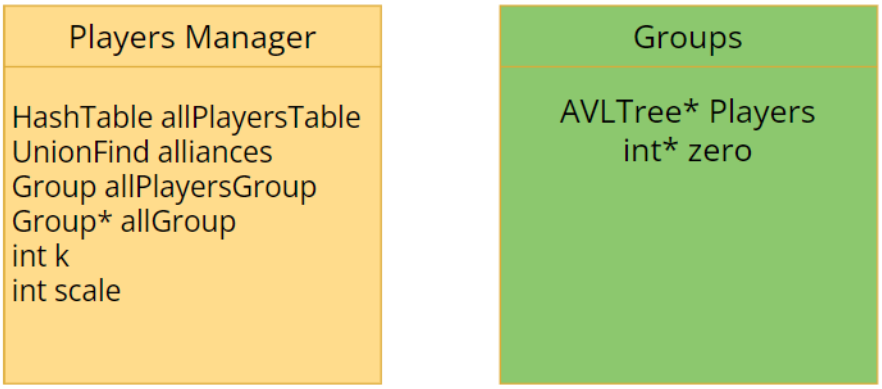
פעולות הגלגול כולל תיקון נותרת $O(1)$ ולכן זמן ההכנסה וההוצאה מעץ הדרגות מבוסס על עץ AVL הוא כמו זמן ההכנסה והוצאה של AVL. (באופן דומה עבור שאר הגלגולים)

- כאשר אנו מוסיפים\מוחקים שחקן לרמה שכבר קיימת בעץ אנו מסיירים על המסלול הרמה עד השורש ומעדכנים את השדות בהתאם פעולה זו היא בסיבוכיות $O(h)$ כאשר h הוא גובה העץ ובעץ AVL מתקיים $h = O(logn)$.

דיאגרמה להמחשה:



והמחלקות:



סיבוכיות מקום:

נסמן:

- n מספר השחקנים במערכת.
- k מספר הקבוצות במערכת.

- מערך ה- alliances בגודל $2k$ ולכן $O(k)$.
- מערך ה- $allGroups$ בגודל k בכל תא ישנם מערך zero בגודל 201 קבוע בנוסף הסכום הכולל של מספר הצמתים בכל העצים בכל הקבצות חסום ע"י מספר השחקנים שהוכנסו כלומר הוא $O(n)$ ולכן סה"כ נקבל $O(n+k)$.
- ו- $allPlayersTable$ הוא $Hash\ Table$ לפי ההרצאה סיבוכיות המקום שלו היא $O(n)$.
- והאובייקט $allPlayersGroup$ מכיל מערך zero בגודל 201 קבוע בנוסף הסכום הכולל של מספר הצמתים בעץ הרמות חסום ע"י מספר השחקנים שהוכנסו כלומר הוא $O(n)$ ולכן סה"כ נקבל
  - $O(n+k)$.
- כלומר סיבוכיות מקום הכוללת של המבנה $O(n+\mathrm{k})$.

הערה:

בתוך הפונקציות שלנו אנו גם כן מקצים מקום זמני אבל גם המקום הזמני חסום ע"י $O(n+\mathrm{k})$.

הסבר למימוש הפונקציות:

---

void init(int k, int scale)*

- הפונקציה בודקת נכונות הקלט , מתקיים ב O(1).
- הפונקציה יוצרת מצביע למבנה חדש עם מערך בגודל k , מאתחלים UnionFind שהוא שני מערכים בגודל k כל אחת,מתקיים ב O(k).
  - סה"כ: O(k).

---

*StatusType mergeGroups(void *DS, int GroupID1, int GroupID2)*

- בודקים שה DS תקין, ומספר הקבוצה תקין (בטווח הנתון)- O(1) .

- מחפשים את הקבוצות בעזרת מבנה UnionFind מתקיים ב O(log*k) משוערך עם Union לפי השיטה שראינו בהרצאה, עצים הפוכים עם איחוד לפי גודל וכיווץ מסלולים.
- ממזגים את העצים באמצעות אותו אלגוריתם שהוצג בתרגול וממשנו גם ברטוב 1 עם הכבדה על עדכון השדות המתאימות- O(n) .
- מאחדים את הקבוצות בעזרת מבנה UnionFind מתקיים ב O(log*k) משוערך עם find .
  - סה"כ: O(log*k +n) משוערך, בממוצע על הקלט.

StatusType AddPlayer(void DS, int PlayerID, int GroupID, int Level)*

- בודקים תקינות הנתונים- O(1).
- בודקים אם השחקן קיים בעזרת חיפוש ב HashTable , מתקיים ב O(1) משוערך, בממוצע על הקלט .
- מוסיפים את השחקן לטבלת השחקנים - מתקיים ב O(1) משוערך, בממוצע על הקלט .
- מוסיפים את השחקן ל AllPlayersGroup מתקיים ב - O(1) כי level=0 סה"כ מוסיפים אחד לשני מקומות במערך.
- מחפשים את הקבוצה בעזרת מבנה UnionFind מתקיים ב O(log*k) משוערך.
- נוסיף את השחקן לקבוצה המתאימה מתקיים ב - O(1) כי level=0 סה"כ מוסיפים אחד לשני מקומות במערך.
  - סה"כ: (O(log*k משוערך, בממוצע על הקלט.

---

*StatusType removePlayer(void *DS, int PlayerID)*

- בודקים תקינות הנתונים- O(1).
- בודקים אם השחקן קיים בעזרת חיפוש ב HashTable , מתקיים ב O(1) משוערך, בממוצע על הקלט .
- שומרים את התכונות שלו במשתנים זמניים למטרת החיפוש בשאר המבנים מתקיים ב O(1).
- מוחקים את השחקן מ AllPlayersTable, מתקיים ב O(1) משוערך, בממוצע על הקלט .
- מוחקים את השחקן מ AllPlayersGroup עם עדכון השדות רק במסלול האיבר שנמחק- O(logn) .
- אם החולייה אחרי מחיקת השחקן תישאר ריקה נמחק אותה מהעץ של AllPlayersGroup עם עדכון השדות רק במסלול האיבר שנמחק – O(logn) .
- מחפשים את הקבוצה של השחקן שנרצה למחוק בעזרת מבנה UnionFind מתקיים ב O(log*k) משוערך.
- מוחקים את השחקן מהקבוצה המתאימה עם עדכון השדות רק במסלול האיבר שנמחק- O(logn) .
- אם החולייה אחרי מחיקת השחקן תישאר ריקה נמחק אותה מהעץ של קבוצתו עם עדכון השדות רק במסלול האיבר שנמחק – O(logn) .
  - סה"כ: O(logn +log*k) משוערך, בממוצע על הקלט .

*StatusType increasePlayerIDLevel(void *DS, int PlayerID, int LevelIncrease)*

- בודקים תקינות הנתונים- O(1).
- בודקים אם השחקן קיים בעזרת חיפוש ב HashTable , מתקיים ב O(1) משוערך, בממוצע על הקלט .
- שומרים את התכונות שלו במשתנים זמניים למטרת החיפוש בשאר המבנים מתקיים ב O(1).
- מחפשים את הקבוצה של השחקן בעזרת מבנה UnionFind מתקיים ב O(log*k) משוערך עם Union לפי השיטה שראינו בהרצאה, עצים הפוכים עם איחוד לפי גודל וכיווץ מסלולים.
- מוחקים את השחקן-O(logn +log*k) משוערך, בממוצע על הקלט .
- מוסיפים השחקן בחזרה ל HashTable עם רמה חדשה ששווה ל old_level+ LevelIncrease, מתקיים ב O(1) משוערך, בממוצע על הקלט .
- נוסיף את השחקן עם רמה חדשה ששווה ל old_level+ LevelIncrease לקבוצה המתאימה ול- AllPlayersGroup עם עדכון השדות רק במסלול האיבר מתקיים ב – O(logn) .
  - סה"כ:O(log*k + logn) משוערך, בממוצע על הקלט .

בנוגע לבונוס : נשתמש בהאש טאבל עם עצים מאוזנים במקום האש טאבל עם שרשרות כי אז נקבל שבמקרה הגרוע חיפוש בעץ מאוזן היא $O(logn)$ ואז נקבל כי הסיבוכיות לוקחת $O(logn)$ לחיפוש השחקן + חיפוש הקבוצה משוערך O(log*k) ומחיקתו והוספתו לעצים מאוזנים היא גם כן $O(logn)$ ולכן סה"כ:O(log*k + logn) משוערך.

*StatusType changePlayerIDScore(void *DS, int PlayerID, int NewScore)*

- בודקים תקינות הנתונים- O(1).
- בודקים אם השחקן קיים בעזרת חיפוש ב HashTable , מתקיים ב O(1) משוערך, בממוצע על הקלט .
- שומרים את התכונות שלו במשתנים זמניים למטרת החיפוש והוספה מחדש בשאר המבנים מתקיים ב O(1).
- מחפשים את קבוצת השחקן בעזרת מבנה UnionFind מתקיים ב O(log*k) משוערך.
- מוחקים את השחקן-O(logn +log*k) משוערך, בממוצע על הקלט .
- מוסיפים השחקן בחזרה ל HashTable עם Score חדש, מתקיים ב O(1) משוערך, בממוצע על הקלט .
- נוסיף את השחקן עם Score חדש לקבוצה המתאימה וגם ל AllPlayersGroup עם עדכון השדות רק במסלול האיבר מתקיים ב – O(logn) .
  - סה"כ:O(log*k + logn) משוערך, בממוצע על הקלט .

*StatusType getPercentOfPlayersWithScoreInBounds (void *DS, int GroupID, int score, int lowerLevel, int higherLevel, double * players)*

- בודקים תקינות הנתונים- O(1).

- אם groupID==0 מבצעים את האלגוריתם הבא על AllPlayersGroup אחרת נחפש את הקבוצה בעזרת UnionFind ב O(log*k) משוערך, ואחר כך נבצע אותו אלגוריתם אך על הקבוצה המתאימה.
- נספור את השחקנים שהרמה שלהם בטווח הנתון על ידי חיפוש החוליות המתאימות בעץ ונבצע פעולת rank פעמיים (פעם לכל Level)בדומה למה שראינו בתרגול אך על השדה של מספר השחקנים– O(logn) .
- נחסיר את התוצאות שקיבלנו ונוסיף את מספר השחקנים במערך zero (שמתאים לשחקנים בעלי רמה 0) אם 0 בתחום ובכך נקבל מספר השחקנים הכולל- O(1).
- באופן דומה נבצע פעולת rank אבל על השדה שמתאים ל score הנתון (כלומר [scores[score) ונקבל הסכום הרצוי – O(logn) .
- נחלק את הסכום במספר השחקנים ונשמור את התוצאה ב $players$ - O(1).
- ס"ה: O(logn+log*k) משוערך, במממוצע על הקלט.

---

*StatusType averageHighestPlayerLevelByGroup(void *DS, int GroupID, int m, double * avgLevel)*

- בודקים תקינות הנתונים- O(1).
- אם groupID==0 מבצעים את האלגוריתם הבא על AllPlayersGroup אחרת נחפש את הקבוצה בעזרת UnionFind ב O(log*k) משוערך, ואחר כך נבצע אותו אלגוריתם אך על הקבוצה המתאימה.
- נבצע פעולות $select(head.sumScores[0]-m+1)$ ו $select(head.sumScores[0])$ כדי לקבל את טווח הרמות המתאים ל m השחקנים בעלי הרמות הגבוהים ביותר.
- אחר כך נבצע שתי פעולות $rankAvg$ על הצמתים שקבלנו מ$select$ כדי לחשב את הסכום המשוקלל ונחלק ב m.
  - סה"כ: O(logn+log*k) משוערך, במממוצע על הקלט.

---

*void Quit(void **DS)*

- בודקים אם NULL!=DS אם לא סיים O(1).
  - אם כן בודקים אם NULL!=DS* אם לא נסיים O(1).
    - אם כן מבצעים delete ל DS* שמשחרר את כל העצים והשדות בגלל שסיבכיות המקום שלנו היא$O(n+\mathrm{k})$ אז סיבוכיות הזמן היא גם $O(n+\mathrm{k})$.
  - סה"כ: $O(n+\mathrm{k})$..