רטוב 1:

הסבר על מבני הנתונים הבסיסיים שהשתמשנו בהם :

המבנה הבסיסי:

עץ AVL :

, בסיבוכיות זמן log(n)השימוש העיקרי במבנה הזה הוא על מנת לנצל את התכונה של חיפוש . הזנה והסרת איברים בעץ AVLשל

מימשנו מילון גנרי בעל מפתח וערך גנריים באמצעות עץ ה-AVL , נשתמש בהם בטיפוסים שונים לטובת המימוש שלנו .

המבנה העיקרי :

כולל שני עצים עיקרים :

1.עץ של קבוצות:

שכל מפתח בעץ הוא המספר המזהה של הקבוצה (ID) וה data של כל קבוצה הוא טיפוס בשם GROUP שכולל :את הid של הקבוצה , מספר השחקנים שיש בה ומצביע על השחקן עם ה LEVEL הכי גבוה , בנוסף כולל עץ של רק השחקנים השייכים לקבוצה לכל שחקן בעץ זה המפתח -הוא טיפוס שכולל את ה ID של השחקן ואת הLEVEL שלו . כלומר העץ מסודר לפי סידור יורד ל level של כל שחקן (אם ה LEVEL של שני שחקנים שונה אז סידור לפי LEVEL אם יש להם את אותו LEVEL אז מסודרים לפי ה IDS שלהם בסדר עולה) בנוסף לסידור עולה של ה ID שלו.

והDATA-הוא מצביע על טיפוס של PLAYER שכולל מצביע לקבוצה ששייך אליה השחקן את ה ID שלו ואת ה LEVEL שלו .

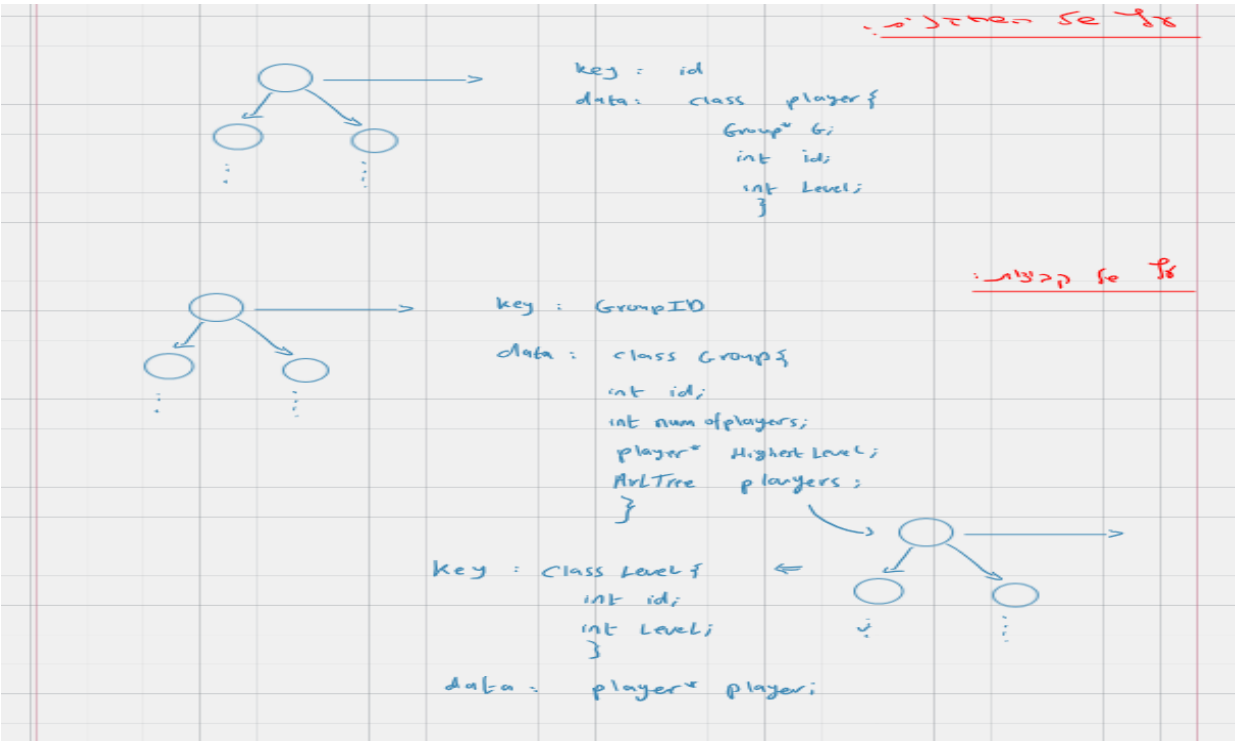
2. עץ של שחקנים :

העץ הזה כולל כל השחקנים שיש בכל המשחק המפתח לכל איבר בעץ הוא ה ID של כל שחקן וה-DATA הוא מצביע על טיפוס של PLAYER שכולל מצביע לקבוצה ששיך אליה השחקן את ה ID שלו ואת ה LEVEL שלו.

3. עץ של הקבוצות הלא ריקות :

מכיל את הקבוצות הלא ריקות , הטיפוס GROUP מכיל שדה בוליאני שמסמן

אם התווספו שחקנים לקבוצה זו , אם כן אנחנו מוספים את הקבוצה לעץ הקבוצות הלא ריקות . המפתח וה DATA של העץ הזה בדיוק כמו עץ הקבוצות.



נפרט עבור כל פונקציה את הסיבוכיות של המימוש שלה :

Init :

אתחול של מצביע למבנה ריק שכולל עצים ריקים לכן זה o(1).

AddGroup(void *DS, int GroupID) :

תחילה נצור אובייקט של GROUP ריק עם GROUPID הנתון – O(1)

נבדוק אם האובייקט שיצרנו נמצא בעץ הקבוצות - O(log K) כאשר K מספר האיברים בעץ הקבוצות (חיפוש בעץ AVL).

אם כן נמצא נמחק את האובייקט ונחזיר שגיאה -O(1)

אם לא נמצא אז נוסיף אותו -O(log K) כאשר K מספר האיברים בעץ הקבוצות (הוספת איבר בעץ AVL).

בסה"כ הוספת איבר לעץ הקבוצות, כפי שלמדנו זה O(log k) במקרה הגרוע, כאשר k הוא מספר הקבוצות(האיברים בעץ).

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

תחילה נצור אובייקט של PLAYER ריק עם PlayerID ו Level- אתחול מצביעם וערכים O(1)

נבדוק אם קיימת קבוצה עם **GroupID** הנתון - O(1)

נבדוק אם האובייקט של PLAYER שיצרנו נמצא בעץ השחקנים - O(log N) כאשר N מספר האיברים בעץ השחקנים (חיפוש בעץ AVL).

אם כן נמצא נמחק את האובייקט ונחזיר שגיאה -O(1)

אם לא נמצא אז נוסיף אותו -O(log N) כאשר N מספר האיברים בעץ השחקנים (הוספת איבר בעץ AVL).

נוסיף אותו לעץ השחקנים של הקבוצה -O(logK)+ O(log *NUMOFPLAYERS*) כאשר K מספר האיברים בעץ הקבוצות (חיפוש הקבוצה בעץ הקבוצות-O(logK) ו*NUMOFPLAYERS הוא מספר השחקנים השייכים לקבוצה* (הוספת איבר בעץ AVL-O(log *NUMOFPLAYERS*).

ע"י בדיקה לשדה הבוליאני של הקבוצה ,נעדכן אותו בהתאם , אם זיהו השחקן הראשון המתווסף לקבוצה זו נעדכן אותו לtrue ונוסיף את הקבוצה לעץ הקבוצות הלא ריקות –

O(log *NUMOGROUPS*)

*** נשים לב שלעץ הקבוצות הלא ריקות מתקיים : (K מספר כל הקבוצות במשחק (ריקות+לא ריקות))(*NUMOGROUPS השייכים לעץ הקבוצות הלא ריקות):*

NUMOGROUPS <=K

השוואה ועדכון במידת הצורך האיבר בעל הLEVEL הכי גדול בעץ של הקבוצה ועדכון מספר האיברים בעץ-O(1)

*** נשים לב שלכל קבוצה מתקיים : (N מספר כל השחקנים)(*NUMOFPLAYERS השייכים לקבוצה):*

NUMOFPLAYERS<=N

בסה"כ הוספת שחקן, זה

O(log k) + O(log N)+O (log *NUMOFPLAYERS*)+ O(log *NUMOGROUPS*)<=

O(log k) + O(log N) + O(log N)+ O(log k) <= O(log k) + O(log N)

במקרה הגרוע, כאשר k הוא מספר הקבוצות(האיברים בעץ)ו N הוא מספר כל השחקנים .

:RemovePlayer(void *DS, int PlayerID)

מחפשים את האיבר שיש להסיר בעץ של השחקנים -O(log N) כאשר N מספר האיברים בעץ השחקנים (חיפוש בעץ AVL).

אם לא נמצא נחזיר שגיאה -O(1)

ניגשים לעץ של השחקנים השייכים לקבוצה של השחקן דרך המצביע שמחזיק השחקן בעץ השחקנים לקבוצה שלו -O(1) (כי לא עשינו חיפוש לקבוצה בעץ אלא שמרנו מצביע ונגשנו באמצעתו)

מוחקים את האיבר מעץ השחקנים של הקבוצה ומעדכנים מספר השחקנים בקבוצה (log *NUMOFPLAYERS* O(

השוואה ועדכון במידת הצורך האיבר בעל הLEVEL הכי גדול בעץ של הקבוצה ועדכון מספר האיברים בעץ של השחקנים השייכים לקבוצה -O(1)

מוחקים את האיבר מעץ השחקנים של כל המשחק -O(log N) כאשר N מספר האיברים בעץ השחקנים

עדכון מספר האיברים בעץ של השחקנים -O(1)

סכ"ה:

O(log N)+ O(log N)+ O(1)+ O(1)+ O(log *NUMOFPLAYERS*)<= O(log N)

במקרה הגרוע, כאשר N הוא מספר כל השחקנים .

:IncreaseLevel (void *DS, int PlayerID, int LevelIncrease)

מחפשים את האיבר שיש לעדכן בעץ של השחקנים -O(log N) כאשר N מספר האיברים בעץ השחקנים (חיפוש בעץ AVL).

אם לא נמצא נחזיר שגיאה -O(1)

ניגשים לעץ של השחקנים השייכים לקבוצה של השחקן דרך המצביע שמחזיק השחקן בעץ השחקנים לקבוצה שלו -O(1) (כי לא עשינו חיפוש לקבוצה בעץ אלא שמרנו מצביע ונגשנו באמצעתו)

מחפשים ומעדכנים את הLEVEL של האיבר מעץ השחקנים של הקבוצה (O(log *NUMOFPLAYERS*

השוואה ועדכון במידת הצורך האיבר בעל הLEVEL הכי גדול בעץ של הקבוצה . –O(1)

שומרים המצביע לקבוצה השייך אליה השחקן. –O(1)

מוחקים השחקן מעץ השחקנים –O(log N)

מוספים שחקן עם אותם פרמטרים אך עם עדכון ה level שלו –O(log N)

סכ"ה:

O(log N)+ O(log N)+ O(1)+ O(1)+ O(1) + O(log *NUMOFPLAYERS*)+ O(log N)<= O(log N)

במקרה הגרוע, כאשר N הוא מספר כל השחקנים .

: GetHighestLevel(void *DS, int GroupID, int *PlayerID)

אם GroupID שלילי אז מעדכנים את הערך שמחזיק השדה של highestLevelPlayer במשחק בPlayerID
אם לא נמצא ערך נעדכן -1 בPlayerID -O(1)

אחרת מחפשים את הקבוצה המתאימה בעץ הקבוצות של המשחק -O(log K) כאשר K מספר האיברים בעץ הקבוצות (חיפוש איבר בעץ AVL).

אם לא נמצאת נחזיר שגיאה -O(1)

אחרת מעדכנים את הערך שמחזיק השדה של highestLevelPlayer בקבוצה בעלת המזהה GroupID בPlayerID -O(1)

סכ"ה:

O(1)+ O(1)+ O(1) + O(log K) <= O(log K)

במקרה הגרוע, כאשר K הוא מספר הקבוצות .

:GetAllPlayersByLevel (void *DS, int *GroupID*, int **Players, int *numOfPlayers)

אם GroupID שלילי אז בודקים אם יש שחקנים במשחק אם לא מעדכנים את המצביעים בהתאם -O(1)

אם יש שחקנים אז עושים סיור INORDER (שמחזיר אותם מסודרים) על עץ השחקנים ושומרים אותם במערך הנתון . -O(N) כאשר N מספר האיברים בעץ השחקנים

אם GroupID איננו שלילי אז מחפשים את הקבוצה המתאימה בעץ הקבוצות -O(logK)

עושים סיור INORDER (שמחזיר אותם מסודרים) על עץ השחקנים של הקבוצה ושומרים אותם במערך הנתון (*NUMOFPLAYERS*)O- כאשר *NUMOFPLAYERS הוא מספר השחקנים השייכים לקבוצה*

במידה ויש שגיאה (אי מציאת הקבוצה בעלת הID הנתון) מחזירים שגיאה --O(1)

סכ"ה:

אם *GroupID*<0 אז $O(n)$ במקרה הגרוע, כאשר n הוא מספר השחקנים במערכת.

אחרת, $O(n_{\text{NUMOFPLAYERS}} + \log k)$ במקרה הגרוע, כאשר $n_{\text{NUMOFPLAYERS}}$ הוא מספר השחקנים ששייכים לקבוצה בעלת המזהה GroupID ו-k הוא מספר הקבוצות.

:Quit(void **DS)

מאחר והקצנו $O(k+n)$ מקום, עלינו לעבור על כל המקום כדי לשחרר אותו. לכן סיבוכיות הזמן היא בהתאם $O(k+n)$.

GetGroupsHighestLevel (void *DS, int numOfGroups, int **Players)

נשתמש בעץ הקבוצות הלא ריקות , באמצעות סיור INORDER על העץ רק עד קבלת **numOfGroups** מהאיברים נשמור אותם במערך של GROUPS בגודל **numOfGroups** . O(numOfGroups)

עוד מעבר על המערך לקבלת האיבר בשדה highestLevelPlayer ונשמור את ה ID של השחקן הזה במערך הנתון (אחרי הקצאה נכונה למערך זה). O(numOfGroups)

בס"ה: O(numOfGroups)

ReplaceGroup(void *DS, int GroupID, int ReplacementID)

חיפוש את שתי הקבוצות בעץ הקבוצות של המשחק -O(log K)*2 כאשר K מספר האיברים בעץ הקבוצות (חיפוש איבר בעץ AVL).

אחרי חיפוש נעשה סיור INORDER על כל עץ ונשמור את הערכים במערכים- $O(n_{group} + n_{replacement})$

סיור ה INORDER מחזיר אותם ממוינים , נעשה מיזוג בין המערכים ממיונים כפי שהוצג בתרגול מספר 5

-O(N*log k) כאשר N שווה לסכ"ה שחקנים בשני העצים ו K הוא מספר המערכים שעושים להם מיזוג כלומר 2 לכן זה

$O(n_{group} + n_{replacement})=O(N)$

בסכ"ה נקבל: $O(\log k + n_{group} + n_{replacement})$ במקרה הגרוע.