ראשית נתאר את מבנה הנתונים בו נשתמש בפתרון התרגיל: מבנה הנתונים יכיל קבוצה של כל השחקנים ומבנה $Union - Find$ של קבוצות.

מחלקות העזר "שחקן", ו-"קבוצה": עבור כל שחקן ישמרו המספר המזהה שלו, רמתו ,התוצאה שלו ובאיזה קבוצה הוא נמצא, ולקבוצה יהיה עץ של כל השחקנים שרמתם אינה 0 הממויין לפי התוצאה, לאחר מכן הרמה ולבסוף המספר המזהה של השחקן, עץ דרגות נוסף של שחקנים הממויין לפי השלב ואז המספר המזהה ובנוסף תחזיק טבלת ערבול של כל השחקנים.

מבנה הנתונים ישתמש במבני הנתונים הבאים:

1. עץ מסוג דרגות מסוג AVL אשר מומש בתרגיל הראשון (בתוספת דרגות הצמתים - כל צומת ידע מה כמות הצמתים בתת העץ החל ממנו). נשתמש לאורך התרגיל בכך שסיבוכיות הזמן של פעולות העץ היא ידועה ולפי מה שנלמד בהרצאות ובתרגולים - מציאת איבר\מחיקה\הכנסה בסיבוכיות של $O(log(n))$ ופעולות סיור על העץ -$inorder, postorder, preorder$ יבוצעו כולן ב- $O(n)$ כאשר n היא כמות האיברים בעץ. כמו כן, נשתמש בפעולת $combineTrees$ אשר מומשה בתרגיל הקודם לצורך איחוד שני עצים (חלק מפונקציית $ReaplceGroups$) וסיבוכיות הזמן שלה היא $O(n_1 + n_2)$ כאשר זו כמות האיברים הכוללת בשני העצים. כמו כן, הראנו בתרגולים ובהרצאות כי ה"שדרוג" של העץ להיות עץ דרגות אינו משנה את סיבוכיות הזמן והמקום של פעולות העץ. סיבוכיות המקום של העץ היא $O(n)$ כאשר n היא כמות האיברים בו.
2. טבלת ערבול - $Hash-Table$ בעלת פקטור עטמס של 1. הטבלה ממומשת כפי שנלמד בהרצאה כך שבכל תא יהיה איבר יחיד ובכך סיבוכיות הזמן הממוצעת על הקלט להכנסה\מציאה\מחיקה של איבר היא $O(1)$. כאשר הטבלה מלאה, נגדיל אותה פי 2, ולאחר מספר גדול של מחיקות ביחס לטבלה, נקצה טבלה חדשה ונוותר על הישנה ובה דגלים שאינם רלוונטיים יותר לחיפוש איבר בתאים ריקים. סיבוכיות המקום של הטבלה היא $O(n)$ כאשר n זו כמות האיברים בה.
3. מבנה $Union - Find$ שיאותחל לגודל k וישאר קבוע למשך כל ריצת התוכנית. המבנה ימומש כפי שנלמד בהרצאה 8 שקופיות 17-21, וישתמש בעצים הפוכים, באיחוד לפי גודל ובכיווץ מסלולים על מנת לפעול בסיבוכיות משוערכת של $log^*(k)$ כאשר k היא כמות האיברים בו. סיבוכיות המקום של המבנה היא $O(k)$.

כעת נתאר את מימוש כל אחת מהפעולות וננתח את סיבוכיות הזמן שלה.

את חישוב סיבוכיות הזמן המשוערכת הנוגעת לחלק שנובע ממספר הקבוצות - k, של הפעולות:

$$averageHighestPlayerLevelByGroup,\ mergeGroups,\ increasePlayerIDLevel,\ changePlayerIDScore,$$
$$getPercentOfPlayersWithScoreInBounds,\ removePlayer,\ addPlayer,\ getPlayersBound$$

כמו גם את חישוב הסיבוכיות המשוערכת הנוגעת לחלק של מספר השחקנים - n, של הפעולות:

$$removePlayer,\ increasePlayerLevel,\ changePlayerIDScore$$
$$addPlayer$$

נחשב בסוף במרוכז לאחר שנתאר כל אחת בנפרד.

תיאור הפעולות:

1. $\boldsymbol{void * init(intk, intsclae)}$: הפעולה תאתחל מבנה $Union - Find$ ריק בגדול k ל-k הקבוצות. סיבוכיות הזמן של הפעולה היא $O(k)$ כי יצירת כל קבוצה היא ב- $O(1)$.
2. $\boldsymbol{StatusType\ mergeGroups(void*\ DS, int\ GroupID1, int\ GroupID2)}$: הפעולה תמצא ותאחד את שתי הקבוצות במבנה ה-$Union-Find$ בסיבוכיות משוערכת של $O(log^*k)$ כפי שהוסבר בהרצאה 8 והצג למעלה, תאחד את עצי השחקנים של שתי הקבוצות בסיבוכיות $O(n)$ כאשר n זוהי כמות השחקנים בשתי הקבוצות ביחד על ידי שימוש במתודה $combineTrees$ שמוגדרת על העץ (תוך שהיא מעדכנת את דרגות העץ באופן רלוונטי), ותאחד את טבלאות השחקנים לכדי טבלה אחת, פעולה שממומשת על ידי המבנה $Hash-Table$ ומבוצעת ב-$O(n)$ בסיבוכיות משוערכת בממוצע

על הקלט. יש לשים לב בעת התייחסות לקבוצה שכבר אוחדה בעבר עם קבוצה אחרת, שחקני הקבוצה הם שחקני הקבוצות המאוחדות ואיחודה עם קבוצה נוספת יהיה איחוד של הקבוצה הנוספת אל הקבוצות שכבר אוחדו. כמו כן, פעולות מבנה ה-$Union-Find$ אינן תלויות בכמה שחקנים יש לכל קבוצה, אלא כמה קבוצות אוחדו ביחד לכדי קבוצה אחת. אין דבר זה משפיע על סיבוכיות פעולות העצים וזה נועד על מנת לממש את המבנה בצורה היעילה ביותר. בסה"כ, סיבוכיות הזמן של הפעולה היא $O(log^*k+n)$ כנדרש.

3. $StatusType\ addPlayer(void*\ DS, int\ PlayerID, int\ GroupID, int\ score)$: הפעולה תיצור שחקן חדש, תכניס אותו בקבוצת כל השחקנים לטבלת כל השחקנים בקבוצה ב- $O(1)$ במשוערך בממוצע על הקלט, תמצא את קבוצות השחקן בסיבוכיות של $O(log^*k)$ במשוערך בממוצע על הקלט ותכניס אותו לטבלת שחקני הקבוצה, גם כן ב-$O(1)$ משוערך בממוצע על הקלט. סה"כ, סיבוכיות הזמן המשוערכת בממוצע על הקלט של הפעולה היא $O(log^*k)$.

4. $StatusType\ removePlayer(void*\ DS, int\ PlayerID)$: הפעולה תמצא את השחקן בקבוצת כל השחקנים (בטבלת הערבול בקבוצה, סיבוכיות משוערת $O(1)$ בממוצע), משם תמצא את קבוצתו דרך השדה השמור אצלו במבנה ב-$Union-Find$ של הקבוצות (סיבוכיות משוערכת $O(log^*k)$ בממוצע) ותסיר אותו מטבלת השחקנים בקבוצה (סיבוכיות משוערת $O(1)$ בממוצע) ומעצי השחקנים במידה ורמתו אינה 0 (היא תחפש אותו לפי התוצאה שלו ולפי רמתו בשני העצים השונים, בסדר בו יכנסו שחקנים לעץ ותוך כדי ההוצאה תעדכן את דרגות הצמתים, סיבוכיות הזמן היא $O(log(n))$ כאשר יש n איברים בעץ). לבסוף הפעולה תסיר אותו מקבוצת כל השחקנים באותו האופן. בסה"כ, סיבוכיות הזמן של הפעולה היא $O(log^*k+log(n))$ כנדרש.

5. $StatusType\ increasePlayerIDLevel(void*\ DS, int\ PlayerID, int\ LevelIncrease)$: הפעולה תמצא את השחקן בקבוצת כל השחקנים (בטבלת הערבול בקבוצה, סיבוכיות משוערת $O(1)$ בממוצע), במידה ורמת השחקן היא 0, תעדכן את רמתו ותכניס אותו לעצי השחקנים שרמתם אינה 0, במידה והיא אינה 0, תעדכן אותו, תמצא אותו בעצים, תסיר אותו, תעדכן את רמתו ואז תכניס אותו מחדש (בכך לא תיפגע בתקינות סדר האיברים בעץ ובדרגות הצמתים). לאחר מכן היא תמצא את קבוצתו דרך השדה השמור אצל השחקן ותבצע את פעולה זו שוב באותו האופן רק בקבוצת השחקן ולא בקבוצת כל השחקנים. הסיבוכיות הכוללת של הפעולה נקבעת לפי מציאתו ועדכנו בטבלאות, בעצים ומציאת קבצתו במבנה הקבוצת ולכן היא בהס"כ תהיה $O\left(log^*k+log(n)\right)$כנדרש.

6. $StatusType\ changePlayerIDScore(void*\ DS, int\ PlayerID, int\ NewScore)$: הפעולה תמצא את השחקן בקבוצת כל השחקנים (בטבלת הערבול בקבוצה, סיבוכיות משוערת $O(1)$ בממוצע), תעדכן את התוצאה שלו, במידה ורמת השחקן אינה 0, תמצא אותו בעץ הרלוונטי, תסיר אותו, תעדכן את התוצאה שלו ואז תכניס אותו מחדש (בכך לא תיפגע בתקינות סדר האיברים בעץ). לאחר מכן היא תמצא את קבוצתו דרך השדה השמור אצל השחקן ותבצע את פעולה זו שוב באותו האופן רק בקבוצת השחקן ולא בקבוצת כל השחקנים. הסיבוכיות הכוללת של הפעולה נקבעת לפי מציאתו בטבלאות, בעצים, מציאת קבצתו במבנה הקבוצת ועדכון התאים במערך ולכן היא בהס"כ תהיה $O\left(log^*k+log(n)\right)$כנדרש.

7. $getPercentOFPlayersWithScoreInBounds(\ DS, GroupID, socre, loweLevel, higherLevel, players)$ $StatusType$: הפעולה תעבור על העץ הממויין תחילה לפי תוצאה, תמצא כמה שחקנים יש בתוצאה המבוקשת כאשר רמתם היא בין שני החסמים (תעשה זאת בכך שתחפש את השחקן שתוצאתו היא המבוקשת ורמתו הכי קרובה מלמעלה לחסם התחתון, ואת השחקן שתוצאתו היא המבוקשת ורמתו הכי קרובה מלמטה לחסם העליון) כאשר את כמות השחקנים היא תחשב לפי דרגות הצמתים, תעשה אותה הפעולה בעץ הממויין רק לפי הרמה והמספר המזהה בו תקבל את כמות השחקנים הכוללת של השחקנים בין רמות אלה ותחזיר את היחס בין שתי הקבוצות. סיבוכיות הזמן הכוללת מתבטאת בחיפוש הקבוצה הרלוונטית ובה חיפוש של ארבעה איברים שונים בעצים וחישוב אמפירי ולכן בסה"כ תהיה $O\left(log^*k+log(n)\right)$ כנדרש.

8. $StatusType\ averageHighestPlayerLevelByGroup(\ DS, GroupID, m, avgLevel)$: פעולה זו תשתמש בשדה נוסף שיהיה שמור לצורך נוחיות העבודה אצל המחלקה "שחקן" ובה יהיה שמור משתנה מסוג $double$. הפעולה תרוץ על השחקנים הממויין לפי רמתם, כאשר תתחיל מהשורש ותחפש בתת העץ הימני כל פעם את הצומת בה כמות הצמתים (השחקנים) בתת העץ הנוכחי היא בדיוק הכמות של השחקנים הטובים ביורת (שרמתם הגבוהה ביותר) להם נרצה לחשב את הממוצע. השדה הנוסף של השחקן יחזיק את ממוצע השחקנים שנמצאים בתת העץ שהצומת כאשר הצומת בה הוא נמצא היא השורש. במידה לא קיימת צומת המחזיקה את המספר המדויק של השחקנים אותו נרצה לחשב, נמצא את האבא שלה (בהנחה שהצומת אינה

השורש כי אז הדרישה היא על כמות שחקנים הגודלה מזו שבעץ, במקרה זה נשתמש גם בשחקנים שרמתם 0 ובמידה וגם אז לא יהיו מספיק שחקנים בקבוצה תוחזר שגיאה). ממנו, נרד לבן הימני, נחשב את סכום הדרגות הכולל שם ונספור כמה שחקנים עברנו, אחר כך נלך לתת העץ השמאלי שלו ונתחיל את החיפוש מההתחלה כאשר השורש הוא הצומת בה אנו נמצאים כרגע (הבן השמאלי של האבא) וכומת השחקנים היא הכמות שחיפשנו בהתחלה פחות כמה שהיו בתת העץ השמאלי. לדוגמה, נניח ונחפש מה ממוצע הרמות של 9 השחקנים ברמה הגבוהה ביותר. בעצם נחפש את הצומת האינדקס הוא 9 ונחזיר את הערך שם. תמיד מתחילים מלמעלה והולכים ימינה לכן נניח ונעצרנו בצומת שגודל תת העץ בה הוא 11, ובבן הימני שלו יש 6. אז נולך לבן הימני, לוקח את הממוצע שיש בו, נכפיל ב-6, חוזר לאבא, הולך ממנו שמאלה ואז ימינה עד הסוף ומחפש צומת עם 3. אם אין, וקיימת נגיד צומת עם 5 ומימינה יש רק 2, נחוזר על התהליך. עדכון העץ יבוצע בכל פעם שנעלה את רמתו של שחקן, הכנסה ראשונית מבוצעת בפעם הראשונה שמעלים את רמתו מ-0. בעץ כן יהיו שחקנים, שימוינו לפי הרמה, ודרגת העץ היא תהיה "האיבר המיוחד" בו. סה"כ, סיבכיות הזמן של הפעולה היא מציאת הקבוצה הרלוונטית, ופעולת הסריקה של העץ המבוצעת בסיבוכיות זמן של $O\left(log(n)\right)$ כאשר n זו כמות האיברים בעץ, ולכן בסה"כ, סיבוכיות הזמן של הפעולה היא $O\left(log^{*}k + log(n)\right)$כנדרש.

9. $StatusType\, getPlayersBound(\, DS, GroupID, score, m, LowerBoundPlayers, HihgerBoundPlayers)$:

10. $void\, Quit(void\, **DS)$: