תרגיל בית 2

במבנה שלנו נשתמש בטיפוסים\מבני הנתונים הבאים:

רשימה מקושרת גנרית דו-כיוונית:
רשימה מקושרת גנרית דו-כיוונית התומכת בפעולות עבור $iterator$ המאפשרת מעבר על הרשומות במבנה. איטרטור יכול להצביע לאיבר במבנה או להצביע לסוף המבנה (להבדיל מהאיבר האחרון במבנה, הצבעה לסוף המבנה היא אינדיקציה לכך שהאיטרטור כבר לא מצביע על אף איבר במבנה.)

לרשימה קיים מצביע $head$ לתחילת הרשימה ומצביע $tail$ לסוף הרשימה.

כל חוליה ברשימה מכילה מצביע "אחורה" לאיבר הקודם ומבציע "קדימה" לאיבר הבא וכן מצביע המכיל את המידע המוכל באותה חוליה.

האיטרטורים תומכים בפעולות הבאות:

- קידום: ע"י אופרטור ++ : מתבצע ב-$O(1)$ ע"י מעבר למצביע לאיבר הבא.
- נסיגה: ע"י אופרטור-- : מתבצע ב-$O(1)$ ע"י מעבר למצביע לאיבר הקודם.
- במידה והאיטרטור מצביע לסוף המבנה, הפעלת האופרטור ++ אינה מוגדרת, במידה והאיטרטור מצביע לתחילת הרשימה, הפעלת האופרטור אינה מוגדרת.
- $Dereference$: ע"י האופרטור האונארי* - מחזיר את המידע בתוך האיטרטור – הוצאה ע"י המצביע למידע ב- $O(1)$.
- השוואה: אופרטור == ואופרטור =!. שני איטרטורים שווים אם הם מצביעים לאותו איבר באותה רשימה, או אם שניהם מצביעים לסוף הרשימה – מתבצע ב-$O(1)$ ע"י השוואת מצביעים ומקור רשימה.

פעולות ברשימה:

- $Iterator\ begin()$ - חזרת איטרטור לתחילת הרשימה. אם הרשימה ריקה, יוחזר איטרטור לסוף הרשימה. מתבצע ב-$O(1)$ ע"י המצביע head ששמור ברשימה.
- $Iterator\ end()$ - החזרת איטרטור לסוף הרשימה. כאמור, האיטרטור לא מצביע לאף איבר ברשימה. מתבצע ב-$O(1)$ ע"י המצביע tail ששמור ברשימה.
- $Void\ insert(const\ T\&\ data, Iterator\ iterator)$ - הערך בעל לרשימה חדש איבר הdata. אם האיטרטור מצביע לסוף הרשימה, אז האיבר החדש יוכנס כאיבר אחרון לרשימה. בכל מקרה, האיבר החדש יוכנס לפני האיבר אליו מצביע האיטרטור. אם האיטרטור הוא של רשימה אחרת, נזרקת חריגה – מתבצע ב-$O(1)$ ע"י שינוי של מס' סופי של מצביעים במקרה הגרוע.
- $Void\ remove(Iterator\ iterator)$- הסרת האיבר אליו מצביע האיטרטור מהרשימה. במידה והרשימה ריקה, או במידה והאיטרטור הוא של רשימה אחרת, או במידה והאיטרטור לא מצביע על איבר ברשימה (למשל מצביע לסוף שלה), נזרקת חריגה. מתבצע ב-$O(1)$ ע"י שינוי של מס' סופי של מצביעים במקרה הגרוע.
- $Iterator\ find(const\ Predicate\&\ predicate)$– חיפוש איבר ברשימה. $Predicate$ הוא טיפוס של $Function\ Object$, כאשר הפונקציה היא פונקציה המקבלת איבר ומחזירה ערך אמת. אם קיים איבר ברשימה המקיים את התנאי $predicate$, יחזור איטרטור עליו. במידה וכמה איברים מקיימים, יחזור הראשון המקיים את התנאי. במידה ולא קיים איבר כזה, יחזור איטרטור לסוף הרשימה – מעבר על כל אברי הרשימה במקרה הגרוע ובדיקת התנאי על כל איבר ברשימה (בהנחה כי בדיקת תנאי היא בדיקה של מספר סופי של פעולות), כלומר מתבצע ב- $O(n)$ כאשר n הוא מס' החוליות ברשימה.
- $int\ getSize()$ – החזרת מספר האיברים ברשימה ע"י מעבר על כל איבר ברשימה וספירת החוליות. מתבצע ב-$O(n)$ כאשר n הוא מס' החוליות ברשימה.

– נסמן את מספר החוליות בעץ כ-n למטרות סיבוכיות.

עץ ספליי ($splay$):
זהו עץ ספליי הממומש כפי שראינו בתרגול.
כל גישה לצומת מגלגלת אותו לשורש (האלגוריתם כפי שנראה בתרגול).
כל עץ מקבל בבנייתו את סוגי הערכים שהוא שומר, ופונקציית השוואה בין ערכים מסוג זה.
העץ ממומש כמחלקה יורשת לעץ חיפוש בינארי רגיל (בקוד), ויש לו את הפעולות הבאות:

א. **מציאת צומת:** מחזיר את ערך הצומת עם המידע שהוכנס, ומגלגל אותו לשורש.
אם אין צומת עם המידע התאים, יוחזר מצביע ריק במקום, אך הצומת עם הערך הקרוב ביותר עדיין יגולגל לשורש – **סיבוכיות משוערכת $O(logn)$.**
ב. **הכנסת צומת:** מכניס צומת ומגלגל אותו לשורש – **סיבוכיות משוערכת $O(logn)$.**
ג. **הסרת צומת:** מגלגל את הצומת עם המידע התאים לשורש ומסיר אותו. אם אין צומת עם המידע המתאים הצומת עם הערך הקרוב ביותר עדיין יגולגל לשורש, אבל לא יוסר – **סיבוכיות משוערכת $O(logn)$.**
ד. ביצוע פעולה כלשהי על כל הצמתים: מבצע פעולה כלשהי על כל הערכים בצמתים, ישנם מספר סדרים שבהם נעבור על הצמתים: $preOrder, inOrder, invertedOrderpostOrder$, כאשר $invertedOrder$ זה כמו $inOrder$ רק הפוך, הוא עובר על הצתמים מהגדול לקטן. זה לא משנה את מבנה העץ – **סיבוכיות $O(n)$.**

– נסמן את מססר הצמתים בעץ כ-n למטרות סיבוכיות

גלדיאטור:
שומר שדות של המזהה היחודי שלו, הרמה שלו ומצביע למאמן שלו.

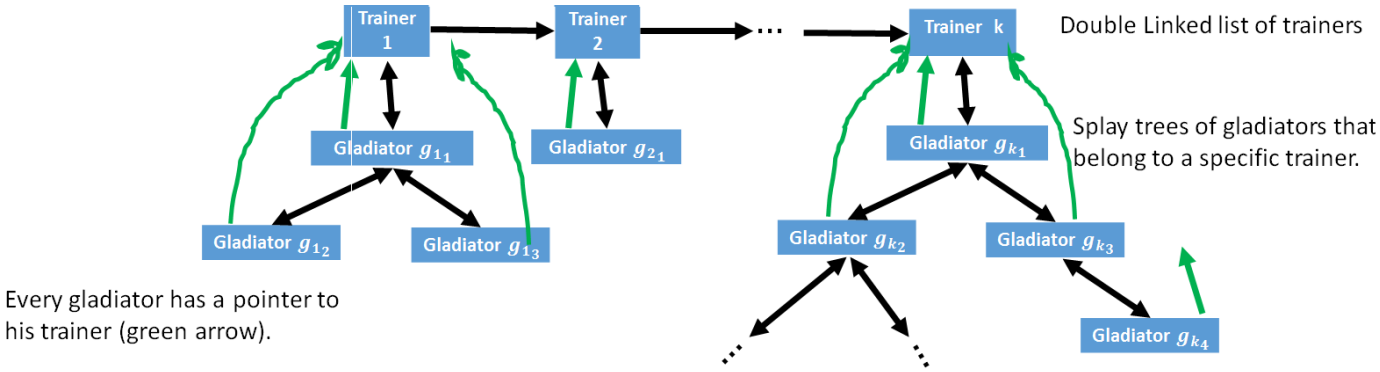
מאמן:
שומר שדות של המזהה היחודי שלו, הגלדיאטור עם הרמה הגבוהה ביותר שלו, מספר הגלדיאטורים שלו ועץ ספליי ממויין **לפי רמות** (מוסבר בהמשך) שמכיל מצביעים לכל הגלדיאטורים שלו.
כל הוספת או הסרת גלדיאטור מעץ המאמן גם מעדכנת את שדה מספר הגלדיאטורים ואם צריך גם מעדכנת את שדה הגלדיאטור עם הרמה הגבוהה ביותר.

קולוסיאום:
לקולוסיאום שדות של מספר הגלדיאוטרים שבו ומצביע לגלדיאטור עם הרמה הגבוהה ביותר שלו.
ישנן 2 דרכי השוואה בין גלדיאטורים:

א. **לפי מזהים יחודיים:** היחס בין שני גלדיאטורים הוא כיחס בין המזהים שלהם במקרה זה, לגדיאטור יש 'ערך' גדול מגלדיאטור אחר אם המזהה היחודי שלו הוא מספר גדול יותר.
ב. **לפי רמות:** היחס בין גלדיאטורים הוא כיחס בין הרמות שלהם במקרה זה, לגדיאטור יש 'ערך' גדול מגלדיאטור אחר אם רמה שלו היא מספר גדול יותר. אם לשני גלדיאטורים את אותה הרמה, נשווה **באופן הפוך** לפי מזהה יחודי - לגדיאטור יש 'ערך' גדול מגלדיאטור אחר אם המזהה היחודי שלו הוא מספר קטן יותר.

בנוסף אליהם, הוא בנוי משלושה מבנים נפרדים שמתקשרים זה עם זה:

1. <u>רשימה מקושרת (דו-כיוונית) של מאמנים שלכל אחד מהם $splay\ tree$ של הגלדיאטורים השייכים לו:</u>

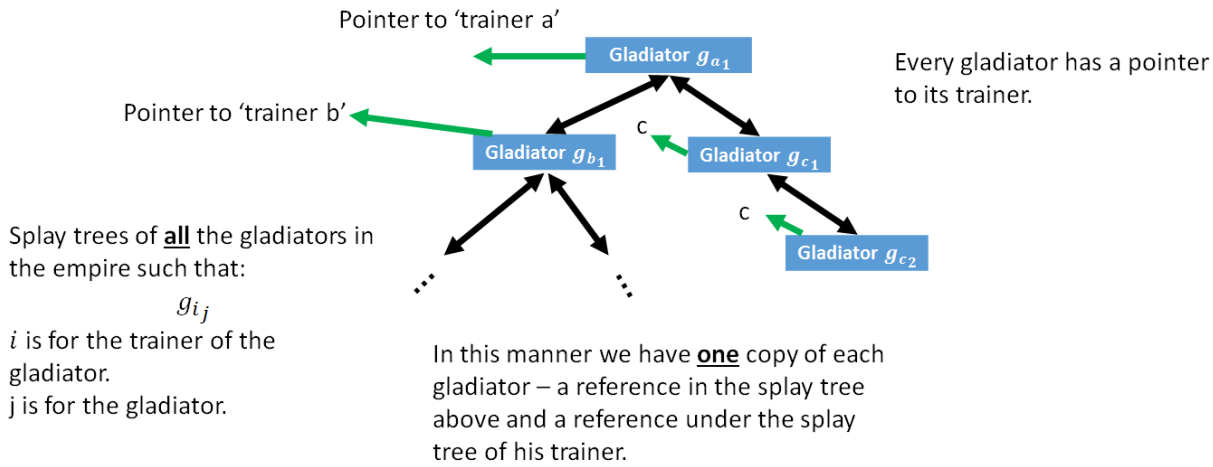


<u>המבנה בנוי מהדברים הבאים:</u>

א. רשימה מקושרת עם k חוליות, כאשר k הוא מספר המאמנים שנוספו למערכת עד כה. הרשימה מסודרת ע"פ סדר המזהים של המאמנים מהקטן לגדול.

ב. לכל מאמן ברשימה הנ"ל ישנו עץ $splay\ tree$, כפי שתואר קודם. העץ מכיל את כל הגלדיאטורים השייכים לאותו מאמן ספציפי. לכל גלדיאטור מצביע למאמן שלו (חץ ירוק). העצים מסודרים בסידור ראשי ע"פ הרמה של הגלדיאטור ובסידור משני ע"פ המזהה הייחודי של כל גלדיאטור (הסדר לפי המזהים הפוך). נראה כיצד נשמר מבנה זה בתיאור הפונקציות בהמשך.
<u>הערה:</u> אם במערכת כולה יש n גלדיאטורים אזי אם נחבר את גדלי כל ה- $splay\ trees$ במבנה הנ"ל נקבל כי בכולם יחד יש n צמתים ולכל אחד מהם מספר קבוע של מידע ששמור בו (נראה זאת גם במימוש הפונקציות). כמו כן, ישנם בדיוק k מאמנים ולכל אחד מהם מספר קבוע של מידע ששמור בו.
לכן **סיבוכיות המקום של תת-המבנה היא $O(n+k)$.**

2. $splay\ tree$ שמכיל את כל הגלדיאטורים במערכת – מסודרים ע"פ המזהה הייחודי שלהם:



המבנה בנוי מהדברים הבאים:

$splay\ tree$ שמכיל את **כל** הגלדיאטורים במערכת. כלומר, זה הוא האיחוד של כל העצים במבנה הקודם. העץ מסודר ע"פ המזהים הייחודיים שניתנו לכל גלדיאטור בהכנסה. נראה כיצד יישמר מבנה זה בתיאור הפונקציות בהמשך. בנוסף, לכל צומת (גלדיאטור) יש מצביע (מסומן בחץ ירוק) למאמן שלו.

הערה: ע"פ הגדרת המבנה, בעץ זה יהיו בדיוק n צמתים (המתארים גלדיאטורים). בכל צומת יש מספר קבוע של מידע ששמור בו ולכן **סיבוכיות המקום של תת-המבנה היא $O(n)$.**

3. $splay\ tree$ שמכיל את כל הגלדיאטורים במערכת – מסודרים ע"פ הרמה שלהם:

מבנה זה זהה למבנה השני למעט הסידור שנעשה קודם ע"פ הרמה של כל גלדיאטור ולאחר מכן ע"פ המזהה הייחודי שלו (כמו בעצים הקטנים של המאמנים).
הערה: ע"פ הגדרת המבנה, בעץ זה יהיו בדיוק n צמתים (המתארים גלדיאטורים). בכל צומת יש מספר קבוע של מידע ששמור בו ולכן **סיבוכיות המקום של תת-המבנה היא $O(n)$.**

במבנה המערכת הכללי מחזיק מצביע לשלושת המבנים, מצביע $topGladiator$ נוסף לגלדיאטור בעל הרמה הכי גבוהה במערכת וכן את מספר הגלדיאטורים הכללי במערכת $gladiatorsNum$.

סיבוכיות מקום של המבנה כולו היא: המבנה בנוי משלושה תת-מבנים שכל אחד מהם הוא מסיבוכיות מקום $O(n+k)$ לכל היותר ולכן סיבוכיות המקום של המבנה כולו הוא מסדר גודל $O(n+k)$.

מימוש:

void * Init() - אתחול של רשימה מקושרת ריקה (תת-המבנה הראשון), 2 עצים ריקים, אתחול מצביע $topGladiator$ ל-null ואתחול $gladiatorsNum = 0$. **סה"כ סיבוכיות זמן $O(1)$.**

StatusType AddTrainer(void * DS, int trainerID) – הוספת חולייה לרשימת המאמנים ע"פ ערך המזהה שניתן בפרמטר הפונקצייה – סיבוכיות זמן $O(k)$. אם הרשימה ריקה, נוסיף את החולייה לתחילת הרשימה (במידה והערך תקין כמובן). במידה והרשימה אינה ריקה, נחפש את המקום המתאים למאמן החדש ברשימה :

א. אם המאמן כבר קיים, לא נשנה את הרשימה.
ב. אם מדובר במאמן חדש, נצרף אותו למקום המתאים מבחינת גודל המזהה (מהקטן לגדול).

במקרה השני, נרוץ במקרה הגרוע ביותר עד סוף הרשימה, כלומר k בדיקות. ולכן סיבוכיות הזמן במקרה השני היא $O(k)$ במקרה הגרוע. כלומר, **סיבוכיות הזמן של כלל הפונקציה במקרה הגרוע היא $O(k)$.**

StatusType BuyGladiator(void * DS, int gladiatorID, int trainerID, int level)- מתבצעת בדיקה האם מזהה הגלדיאטור תקין ולאחר מכן עדכון המבנה נעשה בחמישה שלבים:

א. ראשית נחפש את הגלדיאטור במבנה השני (מסודר ע"פ מזהה). במידה והוא נמצא שם, נצא מהפונקציה – סיבוכיות זמן משוערכת בעץ היא $O(\log n)$.
ב. כעת נחפש את המאמן במבנה הרשימה הראשון. אם המאמן אינו קיים נצא מהפונקציה. אם הוא קיים אזי נוסיף את הגלדיאטור לעץ המקושר אליו – נשמור מצביע p למאמן. בתהליך זה נדרש לחפש את המאמן – לכל היותר k השוואות במבנה של הרשימה המקושרת - ולכן סיבוכיות הזמן היא $O(k)$. כמו כן, נדרש לעדכן את העץ שבו נמצא הגלדיאטור. סיבוכיות מציאת המקום המתאים לגלדיאטור המשוערכת בעץ היא $O(\log n_{trainerID})$. במידה ונדרש עדכון ניעזר במצביע p ע"מ לעדכן את $numOfGladiators$ ואת $topGladiator$ (של המאמן והכללי של המערכת) במידה ונדרש. כעת נוסיף את הגלדיאטור החדש ע"י גישה ישירה לעץ של המאמן מתוך המצביע p . ב-$splay\ tree$ המקרה הגרוע תלוי במספר הצמתים בעץ אבל המקרה המשוערך הוא log של מס' הצמתים. כלומר, $\log n_{trainerID}$ כאשר $n_{trainerID}$ הוא מספר הצמתים (גלדיאטורים) בתת העץ וכן $n \geq n_{trainerID}$. סה"כ סיבוכיות זמן עבור עדכון מבנה זה היא $O(k + \log n_{trainerID})$. במקרה הגרוע כל הצמתים נמצאים תחת אותו מאמן, כלומר $O(k + \log n)$.
ג. במידה ובוצע שינוי במבנה הראשון, נעדכן גם את המבנה השני בהתאם ע"י הכנסת הגלדיאטור לתוך העץ – סיבוכיות עדכון משוערך של $splay\ tree$ בעל n צמתים הוא $O(\log n)$.
ד. במידה ובוצע שינוי במבנה הראשון, נעדכן גם את המבנה השלישי בהתאם ע"י הכנסת הגלדיאטור לתוך העץ – סיבוכיות עדכון משוערך של $splay\ tree$ בעל n צמתים הוא $O(\log n)$.
ה. נגדיל את $gladiatorsNum$ של המערכת ב-1.

בסה"כ עדכון משוערך של המבנה כולו מתבצע ב- $O(k + \log n)$.

S

tatusType FreeGladiator(**void** $*$ DS, **int** gladiatorID) - מתבצעת בדיקה האם מזהה הגלדיאטור תקין ולאחר מכן עדכון המבנה נעשה בשישה שלבים:

א. ראשית, נחפש את הגלדיאטור במבנה השני – במידה ולא נמצא, נצא מהפונקציה. במידה ונמצא, מצביע $trainer$ למאמן שלו ונבצע מחיקה של הגלדיאטור מהעץ במבנה השלישי – סה"כ סיבוכיות משוערכת למחיקה מעץ זה היא $O(\log n)$ יחד עם שמירת המצביע ופעולות חיפוש והסרה מהעצים.
ב. נשתמש במצביע ב- $trainer$ שמצביע אל המבנה הראשון ע"מ לדעת מה הייתה הרמה של הגלדיאטור שנמחק. נשתמש ברמה של הגלדיאטור ע"מ למצוא את מיקומו במבנה השלישי ונעדכן את מבנה זה ע"י מחיקה. זה נעשה באופן משוערך בסיבוכיות $O(\log n)$.
ג. באמצעות המצביע $trainer$ ששמרנו בשלב הראשון נפחית את $numOfGladiators$ של אותו מאמן ב-1 ולאחר מכן נמחק את הגלדיאטור מהעץ של המאמן. כאן הסיבוכיות המשוערכת תלויה במספר הצמתים בעץ של המאמן, כלומר $O(\log n_{trainerID})$.
ד. במידה והמצביע $topGladiator$ (של המאמן או של המערכת הכללית) מצביע לגלדיאטור שהסרנו מן העץ אזי סימן שהגלדיאטור ששיחררנו היה ברמה גבוהה ולכן כעת עלינו למצוא מחדש את אותו הגלדיאטור המדורג הגבוהה ביותר. אם היה מדובר ב-$topGladiator$ הכללי של המערכת אזי נעבור על כל המאמנים ונעדכן בהתאם (עם רמה מקסימלית בין ה-k) – חיפוש במבנה זה פועל בסיבוכיות של $O(k)$. כעת נעדכן גם את המאמן שכן נמחק גם ה-$topGladiator$ הקודם שלו בהתאמה – זה מתבצע ב-$O(\log n_{trainerID})$ שכן כבר נתון לנו המצביע ל-$trainer$. אם מדובר ב-$topGladiator$ של המאמן בלבד אזי נצטרך למצוא את ה-$topGladiator$ החדש שלו. שוב, ידוע לנו המצביע ל-$trainer$ ולכן סיבוכיות הריצה במקרה זה היא $O(\log n_{trainerID})$ בלבד.
ה. נמחוק את הגלדיאטור מהמבנה השני ונעדכן גם אותו (קודם מסירים ואז מוחקים) – משוערך $O(\log n)$.
ו. נקטין את $gladiatorsNum$ של המערכת ב-1.

בסה"כ עדכון משוערך של המבנה כולו מתבצע ב- $O(\log n + k)$.

StatusType LevelUp(void $*$ DS, int gladiatorID, int levelIncrease) - מתבצעת בדיקה האם מזהה הגלדיאטור ותוספת הרמה תקינים ולאחר מכן עדכון המבנה נעשה בשלבים הבאים:

א. נחפש את הגלדיאטור לפי המזהה בעזרת המבנה השני – חיפוש משוערך בעץ מתבצע ב-$O(\log n)$. אם לא נמצא כזה, נצא מהפונקציה.
ב. במידה ונמצא, נמחק את הגלדיאטור במבנה השלישי – משוערך $O(\log n)$.
ג. הגלדיאטור נמצא בראש המבנה השני (כתוצאה משלב א'). נשתמש במצביע שלו ע"מ להבין מי המאמן של אותו גלדיאטור. כעת נמחק את הגלדיאטור מהעץ של מאמנו שבמבנה הראשון - משוערך $O(\log n_{trainerID})$.
ד. נעדכן את הרמה של אותו גלדיאטור בהתאם לקלט הפונקציה. מכיוון שבחיפוש שביצענו בשלב א' כבר מצאנו רת הגלדיאטור ניתן לעדכן ב-$O(1)$ (כי הוא כבר זמין).
ה. נכניס את הגלדיאטור המעודכן לעץ שבמבנה השלישי - משוערך $O(\log n)$.
ו. נכניס את הגלדיאטור המעודכן לעץ שמתאים למאמן שלו במבנה הראשון - משוערך $O(\log n_{trainerID})$.

בסה"כ עדכון משוערך של המבנה כולו מתבצע ב- $O(\log n)$.

StatusType GetTopGladiator(void * DS, int trainerID, int * gladiatorID) – מתבצעת בדיקה האם מזהה הגלדיאטור תקין ולאחר מכן עדכון המבנה נעשה בשלבים הבאים:

א. אם $trainerID < 0$ אזי יש לנו את המצביע הכללי השומר במערכת של הגלדיאטור עם הרמה הכי גבוהה באימפריה – **שליפה ב-$O(1)$ כנדרש.**
ב. אחרת, נחפש את המאמן הנ"ל ברשימה שנמצאת במבנה 1 – מתבצע בסיבוכיות $O(k)$ במקרה הגרוע. אם לא נמצא המאמן, נחזיר כישלון. אם נמצא המאמן נפריד למקרים:
   a. אם למאמן אין גלדיאטורים נחזיר 1-.
   b. אם למאמן יש גלדיאטורים נחזיר את התוצאה מתוך ה-$topGladiator$ האישי שלו, שע"פ הגדרת המבנה מסודר ע"י סידור ראשוני של רמה וסידור משני של $gladiatorID$. מתבצע ב-$O(1)$.

בסה"כ הפונקציה מתבצעת בסיבוכיות זמן $O(k)$.

StatusType GetAllGladiatorsByLevel(void * DS, int trainerID, int ** gladiators, int * numOfGladiator) – מתבצעת בדיקה האם המצביעים תקינים ולאחר מכן עדכון המבנה נעשה בשלבים הבאים:

א. אם $trainerID < 0$ אזי נרוץ על העץ במבנה השלישי (מסודר לפי רמות). ראשית, נקבע את ערך $numOfGladiator$ כ-$gladiatorsNum$ ונקצה מערך בגודל זה – מתבצע ב-$O(1)$.
נרוץ על צמתי העץ מהגדול לקטן ($invertedOrder$) ונוסיף את המזהים הייחודיים של כל גלדיאטור לפי סדר זה למערך (אותו סדר בתוך המערך). מעבר על n צמתים וביצוע פעולות בסדר גודל של $O(1)$ על כל צומת – **לכן בסה"כ סיבוכיות זמן $O(n)$.** נקבע את ערך $gladiators$ כמערך שיצרנו.
ב. אם $trainerID > 0$ נרוץ על הרשימה שבמבנה הראשון עד שנמצא אותו. אם לא נמצא, נצא מהפונקציה. אם נמצא, נקבע את ערך $numOfGladiator$ כמספר הגלדיאטורים של המאמן המתאים ונקצה מערך בגודל זה – מתבצע ב-$O(1)$. נרוץ על צמתי העץ של המאמן מהגדול לקטן ($invertedOrder$) ונוסיף את המזהים הייחודיים של כל גלדיאטור לפי סדר זה למערך (אותו סדר בתוך המערך). מעבר על $n_{trainerID}$ צמתים וביצוע פעולות בסדר גודל של $O(1)$ על כל צומת – לכן בסה"כ סיבוכיות זמן $O(n_{trainerID})$. נקבע את ערך $gladiators$ כמערך שיצרנו. **בסה"כ בנוסף לריצה על המאמנים זה יעשה בסיבוכיות זמן $O(n_{trainerID} + k)$.**

StatusType UpgradeGladiator(void * DS, int gladiatorID, int upgradedID) - מתבצעת בדיקה האם מזהיי הגלדיאטור תקין ולאחר מכן עדכון המבנה בשלבים נעשה הבאים:

א. נחפש את הגלדיאטור לפי המזהה בעזרת המבנה השני – חיפוש משוערך בעץ מתבצע ב-$O(\log n)$. אם לא נמצא כזה, נצא מהפונקציה.
ב. במידה ונמצא, נמחק את הגלדיאטור במבנה השלישי – משוערך $O(\log n)$.
ג. הגלדיאטור נמצא בראש המבנה השני (כתוצאה משלב א'). נשתמש במצביע הדו כיווני שלו ע"מ להבין מי המאמן של אותו גלדיאטור. כעת נמחק את הגלדיאטור מהעץ של מאמנו שבמבנה הראשון - משוערך $O(\log n_{trainerID})$.
ד. נעדכן את הרמה של אותו גלדיאטור בהתאם לקלט הפונקציה. מכיוון שמצאנו אותו בשלב א' העלה ניתן לעדכן אותו ב-$O(1)$.
ה. נוציא את הגלדיאטור מהעץ שבמבנה השני, ואז נכניס אותו - משוערך $O(\log n)$.
ו. נכניס את הגלדיאטור המעודכן לעץ שבמבנה השלישי - משוערך $O(\log n)$.
ז. נכניס את הגלדיאטור המעודכן לעץ שמתאים למאמן שלו במבנה הראשון - משוערך $O(\log n_{trainerID})$.

בסה"כ עדכון משוערך של המבנה כולו מתבצע ב- $O(\log n)$.

StatusType UpdateLevels(void * DS, int stimulantCode, int stimulantFactor) - ראשית, נעשה בדיקה האם הקלטים תקינים ולאחר מכן נעדכן את המבנה באופן הבא:

א. ניצור שני מערכים ריקים בגודל n (הערך נתון באמצעות $gladiatorsNum$), ונשמור מונים $changeNum, sameNum$ של מספר הגלדיאטורים לכל מערך (שניהם מתחילים ב-0) – בסיבוכיות $O(1)$.

ב. נרוץ על העץ במבנה השלישי (מסודר לפי רמות).
נרוץ על צמתי העץ מהגדול לקטן ($invertedOrder$) ונוסיף את הגלדיאוטרים לפי סדר זה למערכים (אותו סדר בתוך כל המערך, נעדכן ונשתמש ב-$changeNum, sameNum$ בהתאם כל פעם), כאשר במערך אחד יאוכסנו כל אלה שנצטרך לשנות את רמתם, ובשני אלו שרמתם לא תשתנה. כל מערך מסודר בסדר יורד של רמות. מעבר על n צמתים וביצוע פעולות בסדר גודל של $O(1)$ על כל צומת – לכן בסה"כ סיבוכיות זמן $O(n)$.

ג. נכפיל את רמת כל הגלדיאטורים שבמערך של הגלדיאטורים שצריך לשנות אותם בפקטור $stimulantFactor$ – מתבצע בסיבוכיות $O(n)$.

ד. ניצור מערך חדש באורך n ונכניס את הגלדיאטורים אליו באופן ממויין כמו בשני המערכים האחרים (לפי רמה ואז לפי מזהה), כאשר נתייחס לגדלי שני המערכים כ-$changeNum, sameNum$ בהתאם. מערך זה גם יהיה ממויין לפי סדר יורד של רמות. נמחק את שני המערכים שייצרנו קודם – סיבוכיות זמן של $O(n)$ (איחוד שני מערכים ממויינים למערך ממויין בגודל n).

ה. נמחק את כל צמתי העץ שבמבנה השלישי, ונרוץ על הרשימה שבמבנה הראשון ונמחק את כל צמתי העצים של האמנים – סיבוכיות זמן $O(n+k)$ מכיוון שרצים על כל הגלדיאטורים והמאמנים.

ו. כעת נוסיף את כל הגלדיאטורים בסדר היורד (שנמצא כבר במערך) לעץ שבמבנה השלישי, כשזה יוצר שרוך וכל הוספה היא 2 פעולות (לשים את הגלדיאטור כבן השמאלי של השורש, ואז להפוך אותו לשורש החדש – נעשה אוטומטית ע"י מימוש עץ splay, ואלה מספר קבוע של פעולות). בנוסף לכך נוסיף כל גלדיאטור לעץ של המאמן המתאים (גישה מיידית באמצעות מצביע), וגם זאת ייצור שרוך וכל הוספה היא 2 פעולות באותו האופן. סיבוכיות זמן - $O(n)$.

ז. נמחק את המערך שיצרנו – סיבוכיות זמן $O(1)$.

בסה"כ הפונקציה מתבצעת בסיבוכיות זמן $O(n+k)$.

void Quit(void ** DS) - נמחק את כל מבני הנתונים, הם לוקחים מקום בסדר גודל של $O(n+k)$ (סיבוכיות המקום של המבנה) ולכן **זה ייעשה בסיבוכיות זמן $O(n+k)$.**