**מבני נתונים תרגיל רטוב 1 – חלק יבש**

**מבנה הנתונים הפנימי**

השתמשנו בעץ Splay כמבנה הנתונים הבסיסי שלנו בתרגיל. כתבנו אותו כעץ גנרי.

**שדות העץ:**

צמתי העץ הם Nodeים, הכוללים מצביעים לבנים הימני והשמאלי(גם הם Node) ומצביע לאב (גםNode ), ומידע גנרי (בעזרת טמפלייטים). לNode בנאי ריק, בנאי המקבל מידע והורס.

בשביל שהעץ יעבוד כראוי דרושים אופרטורי השוואה למידע.

השדה היחיד של העץ עצמו הוא Node השורש.

**המתודות הפרטיות של העץ:**

* **Zig**: מקבלת מצביע לצומת בעץ ומבצעת את גלגול הZig עליו, כפי שנלמד בכיתה, הסיבוכיות של הפעולה היא O(1) כי נעשות רק החלפות מצביעים.
* **Zag**: מקבלת מצביע לצומת בעץ ומבצעת את גלגול Zag עליו, כפי שנלמד בכיתה, הסיבוכיות של הפעולה היא O(1) כי נעשות רק החלפות מצביעים.
* **Splay**: מקבלת מצביע לצומת בעץ ומעבירה אותו לשורש העץ בעזרות גלגולי Zig וZag כפי שנלמד בכיתה, ולפי אותם תנאים (מבצעת הרכבות של גלגולי zig וzag לקבלת גלגולים מורכבים כמו ZigZag או ZigZig), תוך כדי השמירה על תכונות של עץ חיפוש. הסיבוכיות היא ---
* **GetSize**: מקבלת מצביע לשורש העץ ומחזירה את גודל העץ, פונק' רקורסיבית שקוראת לעצמה על תת העץ השמאלי והימני. סיבוכיות זמן O(log n) משוערך (כמו find כפי שנלמד בכיתה וכתוב בתרגיל), כנ"ל סיבוכיות מקום O(log n) משוערך, מאותן סיבות.
* **FindMax**: פונקציית עזר, מקבלת מצביע לשורש העץ ומחזירה מצביע למידע של הצומת המקסימלי, על ידי "הליכה" ימינה כל הדרך, בגלל שהעץ הוא עץ חיפוש. סיבוכיות זמן O(log n) משוערך.
* **GenericInorder**: פונקציית עזר, מקבלת מצביע לשורש העץ, ומופע של מחלקה בעלת אופרטור (). עוברת על העץ בInorder ומפעילה את אופרטור הסוגריים על המידע בכל צומת של העץ. סיבוכיות זמן O(n) בגלל שעוברים על כל הצמתים בעץ. סיבוכיות מקום O(log n) משוערך, לפי עומק העץ, בגלל שהפונקציה רקורסיבית.
* **FillInorder**: פונקציית עזר, מקבלת מצביע לשורש העץ, מערך מאותו סוג של המידע שנמצא בעץ, ואינדקס, ממלאת את העץ Inorder לפי סדר האיברים במערך (לכן אם המערך ממוין נקבל עץ תקין). סיבוכיות זמן O(n) בגלל שעוברים על כל הצמתים בעץ. סיבוכיות מקום O(log n) משוערך, לפי עומק העץ, בגלל שהפונקציה רקורסיבית.
* **RecCopy**: פונקצית עזר לבנאי העתקה, מקבלת מצביע לצומת שהוא שורש העץ שנרצה להעתיק, ומצביע לצומת האב של הצומת הנוכחית שמעתיקים (כך נוכל לסדר את האבות בזמן המעבר), הפונק' יוצרת צומת חדשה לפי הצומת הנוכחית ואז על כל תת-עץ עושה אותו דבר, וקח בצורה רקורסיבית נקבל העתק של העץ שקבלנו את שורשו. סיבוכיות זמן O(n) בגלל שמעתיקים את כל הצמתים בעץ. סיבוכיות מקום O(n) כי יוצרים עץ חדש עם n צמתים.

**מתודות הממשק של העץ:**

* **בנאי ריק:** מאפס את השורש להיות NULL, סיבוכיות של O(1).
* **בנאי העתקה**: מקבל רפרנס לעץ, וקורא לRecCopy שיוצרת העתק של העץ ומחזירה אותו. סיבוכיות זמן O(n) בגלל שמעתיקים את כל הצמתים בעץ. סיבוכיות מקום O(n) כי יוצרים עץ חדש עם n צמתים.
* **GetRoot**: הפונקציה מחזירה מצביע לשורש העץ (לא צריך להעביר לשורש, הוא כבר שם). סיבוכיות של O(1).
* **Insert**: מקבלת רפרנס למידע, הפונקציה יוצרת צומת חדש עם המידע הזה ומכניסה אותו לעץ כמו שנלמד בכיתה על עץ חיפוש רגיל במקומו, לאחר מכן מבצעת Splay, ומעבירה את הצומת החדש שהוכנס לשורש. הפונקציה מחזירה האם ההכנסה הצליחה או לא. הסיבוכיות זמן היא O(log n) משוערך, כי לפי ההנחה בתרגיל זאת הסיבוכיות של ההכנסה, וזאת גם הסיבוכיות של Splay.
* **Find**: מקבלת רפרנס למידע שנרצה למצוא את הצומת שלו (המפתח כלול במידע ולכן ההתאמה היא ייחודית), מבצעת חיפוש כפי שנלמד בכיתה בעץ חיפוש, אם הוא נמצא מחזירה מצביע אליו ובעזרת Splay מעבירה אותו לשורש, אחרת מחזירה NULL. הסיבוכיות זמן היא O(log n) משוערך, כי לפי ההנחה בתרגיל זאת הסיבוכיות של החיפוש, וזאת גם הסיבוכיות של Splay.
* **GenericInorder**: מקבלת ומופע של מחלקה בעלת אופרטור (). קוראת לפונקציית עזר בעלת אותו שם עם השורש של העץ ועם מופע המחלקה. סיבוכיות זמן O(n) בגלל שעוברים על כל הצמתים בעץ. סיבוכיות מקום O(log n) משוערך, לפי עומק העץ, בגלל שהפונקציה קוראת לפונקציה רקורסיבית.
* **FillInorder**: מקבלת מערך מאותו סוג של המידע שנמצא בעץ. קוראת לפונקציית העזר בעלת אותו שם עם המערך, שורש העץ ו0 (האינדקס הראשון במערך). סיבוכיות זמן O(n) בגלל שעוברים על כל הצמתים בעץ. סיבוכיות מקום O(log n) משוערך, לפי עומק העץ, בגלל שהפונקציה קוראת לפונקציה רקורסיבית.
* **FindMax**: קוראת לפונקציית העזר בעלת אותו שם ומחזירה את אותו ערך החזרה. סיבוכיות זמן O(log n) משוערך.
* **ConvertToArray**: מקבלת מערך מאותו סוג של המידע שנמצא בעץ, גודל המערך, מצביע לשורש העץ, אינדקס (בשביל המערך) ומופע של מחלקה בעלת אופרטור סוגריים המחזיר ערך בוליאני. הפונקציה באופן רקורסיבי עוברת Inorder על העץ, וממלאת את המערך (באינדקסים המתאימים, כך שיצא בצורה ממוינת) בכל המידע של הצמתים שעליהם אופרטור הסוגריים מחזיר ערך אמת. המערך הוקצה כבר בחוץ. הפונקציה מחזירה גם את הגודל האמיתי של המערך (כמה תאים מילאנו). סיבוכיות זמן O(n) כי עוברים על כל צמתי העץ.
* **GetData**: מקבלת רפרנס למידע, ומחזירה מצביע למידע של הצומת עם אותו מידע, אם הוא לא נמצא מחזירה NULL. סיבוכיות זמן O(log n) משוערך, כמו find.
* **GetSize**: קוראת לפונקציית העזר בעלת אותו שם עם שורש העץ ומחזירה את אותו ערך החזרה. סיבוכיות זמן O(log n) משוערך, כנ"ל סיבוכיות מקום O(log n) משוערך, מאותן סיבות.
* **הורס**: בצורה רקורסיבית, קורא להורס של השורש, שקורא בצורה רקורסיבית להורס על הבנים שלו וכך מוחק את כל העץ.

**מבנה הנתונים של המערכת**

מבנה הנתונים הכולל הוא Colosseum ומשתמש במחלקות Gladiator ו-Trainer.

Gladiator- גלדיאטור הוא בעל השדות הבאים:

* **gladiatorID**: תעודת הזהות של הגלדיאטור, מזהה ייחודי. ערך חיובי.
* **level**: הרמה של הגלדיאטור. ערך חיובי.
* **trainer**: מצביע אל המאמן האישי של הגלדיאטור. נרחיב על שדה זה בהסבר מחלקת Colosseum.
* **compareByLevel**: שדה בוליאני. אם ערכו true אז נשווה בין שני גלדיאטורים באופן הבא: glad1 > glad2 אם הרמה של glad1 גדולה מהרמה של glad2. במידה והרמות שוות, הגלדיטור הטוב מביניהם יהיה בעל הID הקטן. אם ערך השדה false אז נשווה באופן הבא: glad1>glad2 אם ה-ID של glad1 גדול מ-ID של glad2.
* **factor**: שדה המציין פי כמה להכפיל את הרמה של הגלדיאטור. ערך ברירת המחדל הוא 1. נשנה את שדה זה כאשר נקרא לפונקציית המערכת UpdateLevels.

שני גלדיאטורים ייחשבו שווים אם הם בעלי ID זהה (מזהה ייחודי).

Trainer- מאמן הוא בעל השדות הבאים:

* **id**: תעודת הזהות של המאמן, מזהה ייחודי. ערך חיובי.
* **gladsID**: פוינטר לעץ של גלדיאטורים, כאשר המפתח הוא הID של הגלדיאטור.
* **gladsLevel**: פוינטר לעץ של גלדיאטורים, כאשר המפתח הראשי הוא הרמה של גלדיאטורים והמשני הוא הID. עץ זה ממוין לפי ההסבר עבור compareByLevel = true.
* **topGladiator**: שדה השומר את ה-id של הגלדיאטור הטוב ביותר ששייך למאמן. ערך ברירת מחדל יהיה -1 כאשר למאמן אין כלל גלדיאטורים.

שני מאמנים ייחשבו שווים אם הם בעלי ID זהה (מזהה ייחודי).

הערה לגבי topGladiator: את שדה זה נדאג לעדכן לאחר כל פעולה השמנה את מבנה הנתונים: הוספה/מחיקה של גלדיאטור או עדכון רמה של גלדיאטור. זאת באמצעות קריאה לפונקציה UpdateTopGladiator() אשר מוצאת את הגלדיאטור המקסימלי בעץ gladsLevel. אם למאמן יש גלדיאטורים, העץ לא ריק ולכן האיבר המקסימלי אכן קיים, נשמור את הID שלו כtopGladiator. אחרת, אין גלדיאטורים לכן ערך זה יקבל כברירת מחדל 1-. הפונקציה הנ"ל למעשה קוראת לFindMax() של עץ אשר סיבוכיות הזמן שלה .

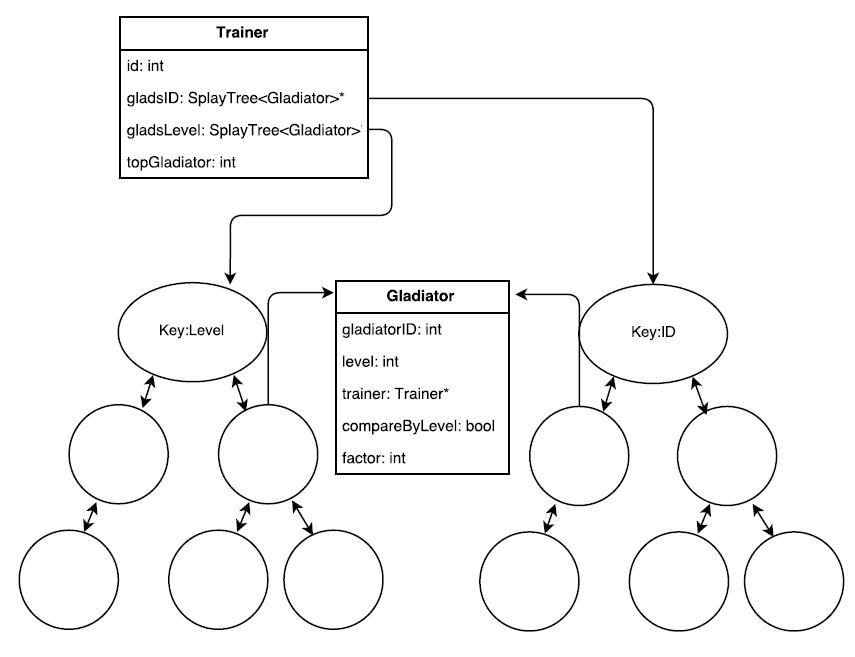
Colosseum- המערכת העיקרית שלנו היא קולוסאום, תשמור את השדות הבאים:

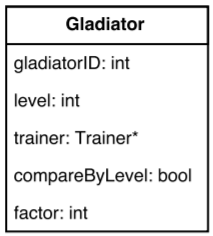
* **trainers**: מצביע לעץ של מאמנים, כאשר המפתח הוא הID של המאמן.
* **masterOfTheArena**: מצביע למאמן הכללי של הקולוסאום. מאמן זה למעשה אחראי על כל הגלדיאטורים אך לא נחשב למאמן האישי שלהם.

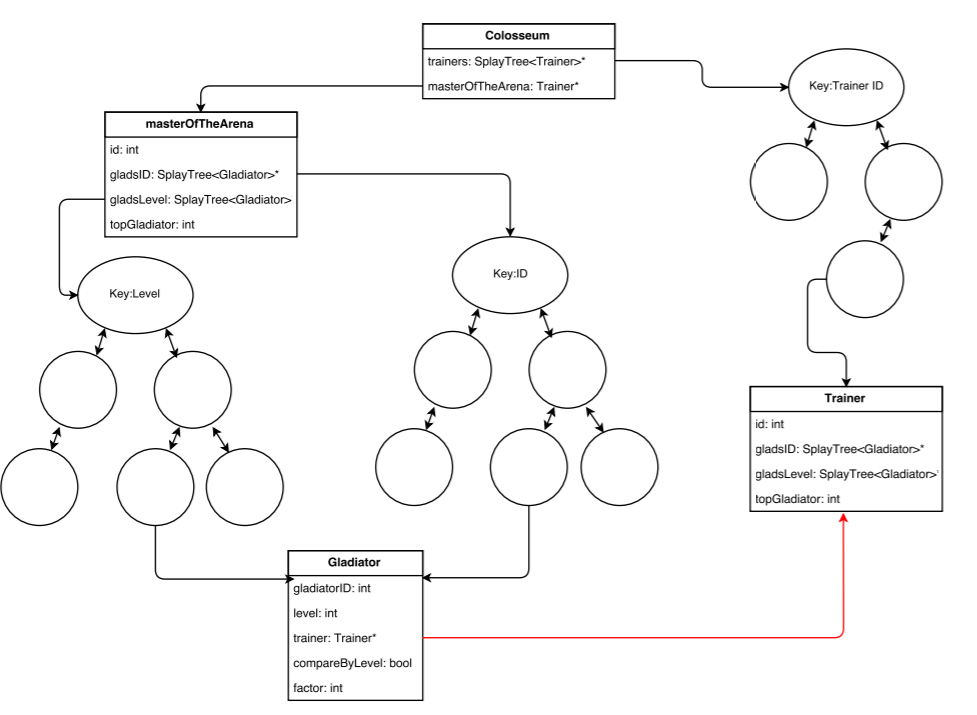
הסבר לגבי השדה trainer בגלדיאטור: כל גלדיאטור נשמור גם במאמן הכללי וגם תחת המאמן האישי שלו. כאשר נרצה להוסיף גלדיאטור, ראשית נמצא את המאמן האישי שלו בעץ trainers. נניח כעת כי המאמן קיים וכל הנתונים תקינים. ניצור גלדיאטור חדש כך שבשדה של trainer יהיה המצביע למאמן בעץ trainers. כיצד זה תורם לנו? בפונקציות הדרושות סיבוכיות זמן של o(n) ראשית נמצא את הגלדיאטור בעץ של המאמן הגדול (פעולת Find) ולאחר מכן באמצעות המצביע למאמן האישי נבצע פעולות מתאימות להעצים של המאמן. כל זה מבלי הצורך לחפש את המאמן.

**סיבוכיות מקום**

לקולוסאום יש מאמן כללי המחזיק בשני עצים בעלי n צמתים כל אחד (כאשר n = מספר הגלדיאטורים). כמו כן, הוא מחזיק בעץ של מאמנים בעל k צמתים כמספר המאמנים. כל מאמן מחזיק בשני עצים בהם חלק מסך כל הגלדיאטורים. ידוע כי אם נסכום על כל הצמתים של העצים תחת המאמנים נקבל את מספר הגלדיאטורים כפול 2 (כי יש 2 עצים לכל מאמן, ולכל גלדיאטור יש מאמן אישי אחד) לכן בסהכ קיבלנו שמספר הצמתים הכולל הוא 4n + k כלומר סיבוכיות המקום היא .

**דיאגרמות לתיאור המערכת**





הערה: כמו שהסברנו, המצביע trainer (באדום) מאפשר לנו גישה ישרה למאמן האישי, מבלי לחפש אותו בעץ trainers

**פונקציות המערכת**

במחלקה Trainer מימשנו:

* **InsertGladiator**: מחפשת את הID של הגלדיאטור העץ ID על ידי פונקציית Find בסיבוכיות O(log n). אם הגלדיאטור קיים, נחזיר שגיאה. אחרת, נוסיף אותו לעץ ID על ידי Insert בסיבוכיות O(log n)משוערך. אם הייתה שגיאת זכרון, נחזיר שגיאה. אחרת, נוסיף אותו גם לעץ Level. אם הפעם הייתה שגיאה, ראשית נמחוק את הגלדיאטור מעץ ID (על ידי Delete בסיבוכיות O(log n)) ולאחר מכן נחזיר שגיאה. חשוב למחוק את הגלדיאטור, כדי לשמור על מבנה המערכת. אם ההוספה התבצעה בהצלחה, נקרא לפונקציה UpdateTopGladiator שתעדכן את השדה של topGladiator בסיבוכיות O(log n) – הסברנו על זה מיקודם.

**בסה"כ אנחנו עושים סדרה של פעולות בסיבוכיות O(log n) משוערך לכן סיבוכיות הזמן היא O(log n) משוערך וסיבוכיות המקום היא O(1) – מקצים מספר סופי של גלדיאטורים בO(1).**

* **RemoveGladiator:** נמחוק את הגלדיאטורים משני העצים על ידי Delete של עץ בסיבוכיות O(log n) ונקרא לUpdateTopGladiator, ייתכן כי מחקנו את הגלדיאטור הטוב ביותר. **סיבוכיות הזמן היא O(log n) משוערך וסיבוכיות המקום O(1) לא הקצאנו מקום.**

**מעתה, כאשר נכתוב כי אנחנו מוסיפים/מוחקים מעץ של מאמן הכוונה היא לקריאה לפונקציות האלו.**

1. **Init**: אתחול של מבנה נתונים חדש, באמצעות קריאה לבנאי הריק של Colosseum. כלומר, יצירה של עץ מאמנים ריק (באמצעות הבנאי שיוצר עץ ריק) וכן יצירה של מאמן כללי (העצים של מאמן חדש הם ריקים) – **כל אלה בסיבוכיות מקום וזמן של O(1).**
2. **AddTrainer**: יצירה של מאמן חדש עם מזהה הID שלו (סיבוכיות ריצה ומקום O(1)). ראשית נבדוק האם המאמן הנ"ל קיים בעץ המאמנים, על ידי פונקציית Find (סיבוכיות זמן O(log n). אם המאמן כבר קיים, נחזיר שגיאה מתאימה. אחרת, נוסיף אותו לעץ המאמנים על ידי Insert בסיבוכיות זמן של O(log n) משוערך. **בסה"כ בצענו שתי קריאות לפונקציות שסיבוכיות הזמן שלהם O(log n) משוערך ולכן סיבוכיות הזמן היא O(log n) משוערך וסיבוכיות המקום O(1) (במקרה הגרוע, הקצאנו מקום למאמן חדש בעץ).**
3. **BuyGladiator**: נחפש את הID של הגלדיאטור תחת העץ id של המאמן masterOfTheArena, זאת על ידי פעולת Find בסיבוכיות זמן O(log n) משוערך. אם הגלדיאטור כבר קיים, נחזיר שגיאה מתאימה. אחרת, נחפש את המאמן הדרוש בעץ המאמנים על ידי Find בסיבוכיות O(log k). אם המאמן לא קיים, נחזיר שגיאה. אחרת, נוסיף את הגלדיאטור למאמן הכללי על ידי InsertGladiator. אם ההוספה נכשלה, נחזיר שגיאה, אחרת נוסיף את הגלדיאטור למאמן האישי שלו. סיבוכיות ההוספה היא גם כן O(log n) משוערך(למעשה מדובר בלוג של מספר הגלדיאטורים השייכים למאמן, ערך זה חסום על ידי n). אם ההוספה נכשלה, ראשית נמחוק את הגלדיאטור מהמאמן הכללי על RemoveGladiator בסיבוכיות זמן של O(log n) משוערך ולאחר מכן נחזיר שגיאה.

בסה"כ במקרה הכי גרוע ביצענו : חיפוש של הגלדיאטור בעץ הגדול, חיפוש של המאמן, הוספה של הגלדיאטורים לשני העצים של המאמן הכללי, הוספה לשני העצים של המאמן האישי. במקרה ויש כשלון בהוספה האחרונה, נבצע מחיקה משני העצים של המאמן הכללי: 7\*O(log n) + O(log k) שזה **סדר גודל של O(log n + log k) משוערך יותר טוב מהנדרש. סיבוכיות המקום היא O(1), הקצאנו לכל היותר מספר סופי של גלדיאטורים בO(1).**

1. **FreeGladiator**: נחפש את הגלדיאטור בעץ ID של המאמן הכללי על ידי Find בסיבוכיות O(log n) משוערך . אם לא קיים, נחזיר שגיאה. אחרת, ניגש לשדה trainer של הגלדיאטור ובכך ניגש למאמן האישי שלו בO(1) במקום לחפש אותו בעץ המאמנים. נמחוק את הגלדיאטור משני העצים של המאמן האישי ומשני העצים של המאמן הכללי: **4 פעולות מחיקה (סיבוכיות כל מחיקה O(log n) משוערך) + חיפוש = O(log n) משוערך. לא הקצאנו זיכרון לכן סיבוכיות המקום O(1).**
2. **LevelUp**: נחפש את הגלדיאטור בעץ ID של המאמן הכללי על ידי Find בסיבוכיות O(log n) משוערך. אם לא קיים, נחזיר שגיאה. אחרת, ניגש לשדה trainer של הגלדיאטור ובכך ניגש למאמן האישי שלו בO(1) במקום לחפש אותו בעץ המאמנים.
3. **GetTopGladiator**: אם הID של המאמן שהועבר הוא -1, אז נשים במשתנה פלט את הID של הגלדיאטור ששמור לנו כtopGladiator במאמן הכללי פעולה בסיבוכיות O(1). אם הID גדול מ0 אז נחפש את המאמן המתאים בסיבוכיות של O(log k) משוערך ואז נשים במשתנה פלט את הID של הגלדיאטור ששמור לנו כtopGladiator במאמן הזה, פעולה בO(1). סה"כ אם ID=-1 אז הסיבוכיות תהיה O(1), אם ID>0 אז הסיבוכיות תהיה ( O(log k, יותר טוב מהמבוקש.
4. **GetAllGladiatorsByLevel**: אם הID של המאמן שהועבר הוא -1, אז נקבע את המאמן הכללי כמאמן המבוקש, אם הID>0 נחפש את המאמן עם הID הזה בO(log k) משוערך, ונקבע אותו כמאמן המבוקש, אם לא קיים כזה נחזיר כי הפעולה נכשלה. כעת לפי המאמן שקבענו נקצה מערך חדש לפי גודל העץ של המאמן הזה ובעזרת פונקציית ConvertToArray של העץ שממוין לפי רמות, סיבוכיות של O(, כש הוא מספר הצמתים בעץ שמתאים למאמן, נקבל כעת מערך של הגלדיאטורים של המאמן, ממוין, כי המעבר הוא inorder על עץ חיפוש, לפי הרמה של הגלדיאטור ולאחר מכן לפי הID שלו, כעת במעבר על המערך (O() נמיר את הגלדיאטורים לID שלהם וכך ניצור מערך בגודל של הID'ים ממוין לפי הדרישות. במקרה ש ID=-1, ולכן סה"כ הסיבוכיות מקום וזמן תהיה O(n) כפי שנדרש, אם ID>0 אז , ולכן סך הכל נקבל שהסיבוכיות זמן היא O(log k+ (יותר טוב מהנדרש), וסיבוכיות מקום O(.
5. **UpgradeGladiator**: ראשית נחפש בעץ הID של המאמן הכללי את הגלדיאטור עם הID שנתון בO(log n), אם הוא לא נמצא נחזיר שהפעולה נכשלה. אחרת, נחפש את את המאמן המתאים לו ב O(log k), כעת נבדוק האם הID של הגלדיאטור קיים, נחפש בO(log n) משוערך אם לא אז נחזיר שהפעולה נכשלה. אם הוא קיים אז נבדוק שupgradeID **לא** קיים במערכת כבר, ושוב נחפש ב O(log n) משוערך, אם כבר קיים כזה נחזיר שהפעולה נכשלה. אם לא קיים כזה, נסיר את הגלדיאטור מהמאמן הכללי והאישי ב O(log n) משוערך, ואז נוסיף גלדיאטור עם אותם נתונים רק שעכשיו הID שלו מעודכן לupgradeID, גם ההכנסה ב O(log n). לכן סך הכל סיבוכיות הפעולה היא O(log n) משוערך.
6. **UpdateLevels**:
7. **Quit**: הפונקציה מוחקת את כל מבנה הנתונים. קוראת להורס של המאמן הכללי ולהורס של עץ המאמנים, ובכך מוחקת את כל המידע במבנה הנתונים בצורה רקורסיבית. סך כל הצמתים שאנחנו שומרים הוא O(n+k), כי אנחנו שומרים מספר קבוע של עותקים של כל גלדיאטור, ושומרים בדיוק k+1 מאמנים, וצריך לעבור על כולם כדי לשחרר אותם מהזכרון. סה"כ הסיבוכיות היא O(n+k).