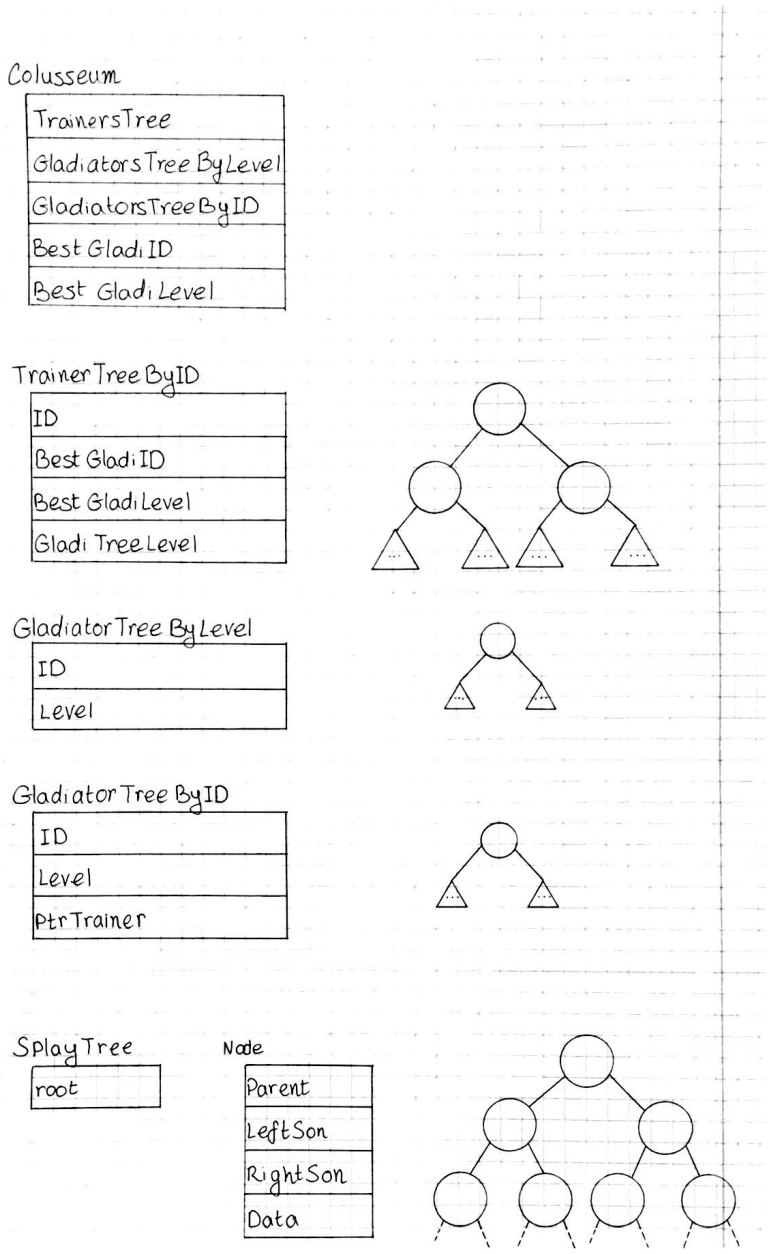
מבני נתונים 234218 - תרגיל בית רטוב 1

מבנה הנתונים:



לטיפוס הראשי מסוג Colosseum יש שני שדות לטובת הגלדיאטור הכי טוב, ועוד שלושה עצים:

- גלדיאטורים ממויין לפי ID.
- גלדיאטורים ממויין לפי Level+ID
- מאמנים ממויין לפי ID

בעץ הראשון קיימים איברים מטיפוס Gladiator, בהם יש ID, רמה ומצביע למאמן של אותו הגלדיאטור.
לטיפוס זה אופטרור השוואה ע"פ ID.

בעץ השני קיימים איברים מטיפוס GladiatorL, בהם יש ID ורמה של אותו הגלדיאטור.
לטיפוס זה אופרטור השוואה ע"פ Level תחילה ולאחר מכן ID (לפי הדרישה).

בעץ השלישי קיימים איברים מטיפוס Trainer, בהם יש ID, רמה, פרטי הגלדיאטור הטוב ביותר של אותו מאמן ועץ הגלדיאטורים של המאמן הממויין ע"פ Level+ID.
לטיפוס זה אופרטור השוואה ע"פ ID.

בנוסף, טיפוס עץ הSplayTree מכיל מצביע לNode הראשון בשורש.
לכל Node יש גישה(מצביע) אל שני הבנים שלו ואל האבא שלו. בנוסף כמובן יש לו מצביע לData.

סיבוכיות המקום הנדרשת$O(n+k)$, מתקיימת מכיוון שיש בColosseum שלושה עצים: שני עצים של גלדיאטורים(משני סוגים טיפוסים כל אחד) לכן סיבוכיות המקום היא לכל היותר $O(n)$. בנוסף קיים עוד עץ של מאמנים עם k איברים ובכל איבר יש $n_{trainerID}$ גלדיאטורים בעץ אחד, ומתקיים $\sum n_{trainerID} = n$ לכן סיבוכיות המקום של עץ המאמנים היא $O(n+k)$.
בנוסף במספר פונקציות יש הקצאת מערכים זמניים באורך n או k לכל היותר.
לסיכום, סיכויות המקום הכוללת היא $O(n+k)$.

מימוש הפונקציות הColosseum:

void* Init() -

פונקציית איתחול המערכת,
פונקציה זו יוצרת טיפוס חדש מסוג Colosseum.
בעת יצירת Colosseum חדש מאותחלים השדות שלו, בין היתר מאותחלים העצים שלו(עץ הID ועץ הLevel של הגלדיאטורים ועץ המאמנים).
שדות(level וID) הגלדיאטור הטוב ביותר מאותחלים לערך 1-.
סיבוכיות הפונקציה הינה ב$O(1)$, כי מספר הפעולות קבוע תמיד.

StatusType AddTrainer(Colosseum *DS, int trainerID) -

פונקציית הוספת מאמן,
פונקציה זו בודקת האם קיים כבר מאמן בעל ID זהה(במידה וכן זורקת שגיאה).
לאחר מכן, יוצרת Trainer חדש ומוסיפה לעץ המאמנים את האיבר החדש ע"י פעולת Insert לעץ.
פעולת יצירת הTrainer מאפסת את שדות המאמן וגם מאתחלת את עץ העצים ע"פ level.
בנוסף עולה מספר המאמנים.
סיבוכיות הפונקציה הינה ב$O(log(k)) \leq O(k)$ משוערך, כי מבוצע מעבר ראשוני על עץ המאמנים ב $O(log(k))$ משוערך, לאחר מכן יצירה מאמן ב$O(1)$ ולבסוף הוספה של של איבר לעץ ב$O(log(k))$.

StatusType BuyGladiator(Colosseum *DS, int gladiatorID, int trainerID, int level) -

פונקציית הוספת גלדיאטור,
פונקציה זו בודקת האם קיים מאמן כזה, והאם קיים גלדיאטור בעל ID זהה(במידה וכן זורקת שגיאה).
לאחר מכן, יוצרת Gladiator חדש עם הפרטים הרלוונטים וגם עם מצביע אל המאמן שמצאנו בעת החיפוש שלו. את טיפוס הGladiator מוסיפים לעץ הגלדיאטורים הראשי ע"פ ID בColosseum.
לאחר מכן יוצרת GladiatorL חדש עם הפרטים הרלוונטים ומוסיפים אותו לעץ הגלדיאטורים הראשי ע"פ Level ולעץ הגלדיאטורים של אותו מאמן.
לבסוף בודקים האם הגלדיאטור החדש הופך להיות הגלדיאטור הכי טוב, ע"פ דרישות level תחילה ולאחר מכן ID הקטן מבינהם. במידה ויש צורך לעדכן, מחליפים את הlevel והID.
באותו אופן מבצעים זאת לגלדיאטור הטוב ביותר בכל המערכת.
בנוסף עולה מספר הגדליאטורים במערכת ובמאמן באחד.
סיבוכיות הפונקציה הינה ב$O(log(n) + log(k)) \leq O(log(n) + k)$ משוערך, כי מבוצע חיפוש ראשוני על העץ המאמנים ועל עץ הגלדיאטורים הראשי ב$O(log(k) + log(n))$, לאחר מכן יוצרים גלדיאטור ב $O(1)$ ומוסיפים אותו לכל עץ ב$O(log(n))$ משוערך ולבסוף עדכון הגלדיאטור הטוב ב$O(1)$.

StatusType FreeGladiator(void *DS, int gladiatorID) -

פונקציית שיחרור/מחיקת גלדיאטור,
פונקציה זו בודקת האם קיים גלדיאטור עם הID הזה(במידה ולא זוקרת שגיאה).
לאחר מכן, מוחקת את הגלדיאטור משני העצים הראשיים(הlevel מתקבל מהחיפוש הראשון בעץ הID).
ע"י מציאת הגלדיאטור בעץ הראשי ע"פ ID מקבלת מצביע למאמן של אותו גלדיאטור, מוחקת את הגלדיאטור מעץ הlevel(הרמה של אותו גלדיאטור מתקבלת גם כן מהחיפוש הנ"ל בעץ הID).
אם הגלדיאטור שנמחק הינו הגלדיאטור הכי טוב של המאמן, נמצא הגלדיאטור הבא בטור ברמתו ע"י פונקציית הFind-Max של העץ.
לבסוף במידה והגלדיאטור הנמחק היה גם הכי טוב בColosseum עוברים על עץ המאמנים בשיטת presort ומוצאים את הגלדיאטור הכי טוב מבין הטובים של כל מאמן.
בנוסף מורידים את המספר הגלדיאטורים אצל המאמן ובסך הכל.
סיבוכיות הפונקציה הינה ב$O(log(n) + k)$ משוערך, כי מבוצע חיפוש ראשוני על עץ המאמנים ועל עץ הגלדיאטורים הראשי ב$O(log(n))$, מוחקים מכל עץ את הגלדיאטור הנ"ל ב$O(log(n))$ ולבסוף במידת הצורך מבוצע Find-Max ב$O(log(n))$ וגם מבוצע מעבר על כל המאמנים ב$O(k)$.

StatusType LevelUp(void *DS, int gladiatorID, int levelIncrease) -

פונקציית העלאת רמת הגלדיאטור,
פונקציה זו בודקת האם קיים גלדיאטור עם הID הזה(במידה ולא זוקרת שגיאה).
לאחר מכן, מעדכנת את הרמה של הגלדיאטור בעץ הראשי לפי ID.
בעץ הראשי לפי level נמחק הגלדיאטור ומוסף שוב מחדש עם הרמה החדשה.
באותו אופן נמחק ומוחזר הגלדיאטור בעץ של המאמן(יש מצביע למאמן בעץ הID).
לבסוף במידה והרמה החדשה של הגלדיאטור גורמת לו להפוך להיות הכי טוב, משתנה המידע אצל המאמן, וגם אצל הColosseum במידת הצורך.
סיבוכיות הפונקציה הינה ב$O(log(n))$ משוערך, כי מבוצע חיפוש ראשוני על עץ הID של הגלדיאטורים ב$O(log(n))$, לאחר מכן נמחק ומוחזר הגלדיאטור בשני העצים לפי level ב$O(log(n))$, ולבסוף מעודכנים ערכי הגלדיאטור הטוב ב$O(1)$.

StatusType GetTopGladiator(void *DS, int trainerID, int * gladiatorID) -

פונקציית מציאת הגלדיאטור הטוב ביותר,
במידה והtrainerID קטן מ0, מוחזר הגלדיאטור הטוב ביותר ששמור בColosseum ב$O(1)$.
אחרת, נבדק האם יש מאמן עם ID כזה(במידה ולא זורקת שגיאה).
מתוך המאמן מוחזר הגלדיאטור הכי טוב.
במידה ואין למאמן או לכל המערכת גלדיאטורים מוחזר 1-.
סיבוכיות הפונקציה הינה ב$O(log(k)) \leq O(k)$ משוערך, כי יש מעבר על עץ המאמנים ב$O(log(k))$ ואז מבוצעת פעולה ב$O(1)$.

StatusType GetAllGladiatorsByLevel(void *DS, int trainerID, int **gladiators, int *numOfGladiator) -

פונקציית מציאת כל הגלדיאטורים ע"פ רמה,
במידה והtrainerID קטן מ0, מוחזר רשימת כל הגלדיאטורים במערכת ע"י מעבר Presort "הפוך" ב$O(n)$.
אחרת, נבדק האם יש מאמן עם ID כזה(במידה ולא זורקת שגיאה).
מתוך המאמן שנמצא מוחזרת רשימת הגלדיאטורים שלו ע"י מעבר Presort "הפוך" ב$O(n)$.
פונקציית הPresort ממימוש העץ.
סיבוכיות הפונקציה הינה ב$O(n_{trainerID} + log(k)) \leq O(n_{trainerID} + k)$ משוערך, כי יש מעבר על עץ המאמנים ב$O(log(k))$, ולאחר מכן מעבר Presort ב$O(n_{trainers})$.

StatusType UpgradeGladiator(void *DS, int gladiatorID, int upgradedID) -

פונקציית עדכון ID הגלדיאטור,
פונקציה זו בודקת האם קיים גלדיאטור עם הID הזה(במידה ולא זוקרת שגיאה).
לאחר מכן, הפונקציה מוצאת בעץ הראשי לפי ID את הגלדיאטור, שומרת את הפרטים ומוחקת את האיבר. בנוסף נמחק הגלדיאטור גם מהעץ הראשי לפי level וגם בעץ של המאמן(יש מצביע אליו מהחיפוש הראשון).
לאחר מכן נוצר גלדיאטור חדש והוא מוכנס לכל העצים עם הID החדש.
לבסוף נבדק אם הID הישן של הגלדיאטור הינו אחד של הגלדיאטור הכי טוב הן אצל המאמן והן אצל הColosseum, במידה וכן מעודכן הID שלו.
סיבוכיות הפונקציה הינה ב$O(log(n))$ משוערך, כי יש חיפוש ראשוני של הגלדיאטור ב$O(log(n))$, לאחר מכן מחיקה שלו משלושת העצים ב$O(log(n))$, ולבסוף הוספה שלו חזרה ב$O(log(n))$ לשלושת.
יש גם בדיקה של הגלדיאטור הכי טוב ב$O(1)$.

StatusType UpdateLevels(void *DS, int stimulantCode, int stimulantFactor) -

פונקציית עדכון רמת כל הגלדיאטורים המתאימים לקריטריון,
הפונקציה עוברת בכל עץ של גלדיאטורים על כל הטיפוסים בInOrder "הפוך"(ימין לשמאל). ושומרת אותו במערך בגודל העץ(n או k) לאחר מכן יש מיון של האיברים לשני מערכים, כאשר מערך אחד לטובת אלה שיש לפקטר והשני לטובת היתר. מפקטרים תוך כדי(במערך המפוקטרים המיוון נשמר, factor>1).
לבסוף מוחקים את העץ הישן(המיון בו לא מתאים לעדכון, נהרסת השמורה), ומכניסים מחדש את כל האיברים לעץ בשיטת marge-sort בין שני מעריכים ממויינים.
סיבוכיות הפונקציה הינה $O(n+k)$ מכיוון שיש מעבר על כלל האיברים בכל עץ ללא לולאות מקוננות, פרט לזאת של עץ המאמנים אך שם סכום כל הגלדיאטורים בכל המאמנים הוא n לכן זה עדיין מתקיים.

void Quit(void **DS) -

פונקציית סגירת המערכת,
הפונקציה משתמשת בהורסים של הטיפוסים.
הורס של העץ מוחק כל פעם את השורש, ומשייך את הבנים שלו לשני טיפוסים חדשים מסוג SplayTree, וכך חוזר חלילה(שימוש רקורסיבי בהורס של העץ) עד הגעה לעץ ריק.
סיבוכיות הפונקציה הינה $O(n+k)$ מכיוון שיש מעבר יחיד על כל האיברים במקנה הנתונים(על כל איבר בעץ יש מעבר יחיד לטובת מחיקה).

מימוש הפונקציות הSplayTree:

המימוש התבצע ע"פ הPDF שצורף,
בנוסף יש מימוש של פונקציית InOrder "הפוך" מימוש דומה לזה מהתרגול לכן מתקיים $O(n)$ כמספר האיברים בעץ.