RAVL<Magi,Magi>availableMagiTree: זהו עץ דרגות אשר מכיל את המאגים מסודרים על פי הדרגה כאשר בכל צומת שמור מידע נוסף שהוא מצביע למאגי הכי צעיר בתת העץ ,בעץ זה נמצאים רק המאגים אשר לא אחראים על אף איזור

AVLTREE<int,Magi>allMagiTree : זהו עץ AVL אשר מכיל את כל המאגים שנמצאים במערכת (גם כאלו שאחראים על איזור ברגע זה וגם כאלו שאינם) ,עץ זה מסודר לפי מספר הזהות של המאגים.

סיבוכיות פונקציות :

INIT: באתחול המערכת למעשה רק מאותחלים ארבעת מבני הנתונים לכן אתחול של עץ דרגות ריק בO(1) אתחול של עץ AVL ריק כנ"ל ב-O(1), אתחול של UNIONFIND כמו שראינו בהרצאה בO(n) ואתחול טבלת ערבול ב-O(1) לכן בסה"כ O(n) במקרה הגרוע.

AddMagiZoologist: בהוספת מאגי למערכת, המאגי מוכנס לעץ allMagiTree, כמו שלמדנו הכנסה לעץ AVL ב O(log(k)) כמו כן המאגי מוכנס לעץ availableMagiTree כנ"ל הכנסה לעץ ב O(log(k))(כמו שראינו תוך כדי ההכנסה לעץ אנחנו גם מעדכנים את המידע הנוסף השמור בכל צומת בלי לפגוע בסיבוכיות זו), בנוסף המאגי מוכנס לטבלת ערבול (עם מערכים דינאמיים וdouble hashing ) כמו שלמדנו זה נעשה בO(1) משוערך בממוצע על הקלט לכן בסה"כ קיבלנו שעושים את זה בO(log(k)) משוערך בממוצע על הקלט כמו שראינו בתרגול.

RemoveMagiZoologist: בהסרה מהמערכת יש קריאה לreleaseMagi אשר כמו שנראה למטה נעשה בO(log(k)) במקרה הגרוע, בנוסף יש הסרה משני עצים שכל אחד נעשה בO(log(k)) והסרה מטבלת הערבול לכן בסה"כ נקבל כמו ב AddMagiZoologist שעומדים בO(log(k)) משוערך בממוצע על הקלט כמו שראינו בתרגול.

RemoveBarrier: זוהי למעשה אך ורק פעולת JOIN של unionfind עם כיווץ מסלולים ואיחוד לפי גודל לכן כפי שלמדנו זה נעשה בO(log\*(n)) משוערך כאשר בכל פעולה כזו נעדכן בשורש העץ ההפוך את מספר החיות שבאזור ואת סך כל רמות הסיכון שלהן, (כמובן שעדכון זה נעשה בO(1)) במידע זה נשתמש לפעולות אחרות.

AssignMagizoologistToCreature: בפעולה זו ראשית אנו מסתכלים בAnimalZoneUF ומחפשים את רמת הסיכון של האיזור מציאת האיזור היא למעשה פעולת FIND של unionfind עם כיווץ מסלולים ואיחוד לפי גודל לכן נעשית בO(log\*(n)) כאשר בכל פעולת UNION שמנו את המידע "רמת הסיכון של האיזור" בשורש העץ ההפוך לכן לאחר שמצאנו את האיזור הרלוונטי החזרת מידע זה נעשית בO(1) ,כאשר יש בידינו את מידע זה נחפש בעזרתו את את האיבר המינימלי שעומד בקריטריון זו בעץ availableMagiTree שכאמור מסודר לפי רמת כישוף (תיאור אלגוריתם זה נמצא בתיאור העץ availableMagiTree) לאחר שמצאנו את המינימלי לפי הindex שלו נוכל לגשת למערך של טבלת הערבול בO(1) ולעדכן למאגי את החיה שהוא אחראי עליה, להוציא את המאגי מהעץ availableMagiTree בO(log(k)) לכן בסה"כ ב O(log(k)+log\*(n))

ReleaseMagiZoologist: נמצא את המאגי בעץ allMagiTree בlog(k) דרך הindex שנמצא אצלו נוריד ממנו את האחריות לאיזור בטבלת הערבול בO(1) ונוסיף אותו לעץ availableMagiTree בO(log(k)) לכן בסה"כ בO(log(k)) במקרה הגרוע.

GetCreatureOfMagi: נמצא את המאגי בטבלת הערבול זהו חיפוש בטבלת ערבול עם מערכים דינאמיים וdouble hashing לכן נעשה בO(1) בממוצע על הקלט כפי שנלמד בהרצאה מכיוון שעידכנו את החיה שעליה אחראי המאגי בפעולות ReleaseMagiZoologist ו – AssignMagizoologistToCreature נוכל עכשיו למצוא את החיה המתאימה בO(1), לכן בסה"כ זה נעשה בO(1) בממוצע על הקלט.

AreCreaturesInSameArea: זוהי למעשה רק פעולת FIND של unionfind שמתבצעת כמו שלמדנו בO(log\*(n))-, פעם עבור החיה הראשונה ופעם עבור השניה מוצאים לאיזה אזור שייכת החיה לכן נעשה בסה"כ בO(log\*(n))-

GetSizeOfArea: זוהי למעשה רק פעולת FIND של unionfind שמתבצעת כמו שלמדנו בO(log\*(n)) כאשר בראש העץ ההפוך עדכנו כל פעולת UNION את מספר החיות שנמצאות באזור לכן נשיג זאת בO(1) וסה"כ ב-O(log\*(n)).

סיבוכיות מקום:

במבנה הכללי נמצאים ארבעה מבני נתונים:

* עץ הAVL allMagiTree שבו כל מאגי נמצא פעם אחת בלבד לכן O(k)
* עץ הדרגות שבו כל מאגי נמצא מקסימום פעם אחת לכן גם הוא O(k)
* טבלת ערבול שבה מוחזק מצביע עבור כל מאגי בדיוק פעם אחת לכן כנ"ל O(k)
* UF שבו כל חיה נמצאת פעם אחת בלבד לכן O(n)

בסה"כ O(k+n) מקום.