מגישים: רן לוטם, דביר פרי

<u>תיאור מילולי של מבנה הנתונים</u>

המחלקה הראשית בשם MagicManager מכילה ארבעה מבני נתונים:

- UnionFind של חיות: ממומש כמערך של חיות, המייצג עצים הפוכים. כל חיה מכילה מידע על הרמה שלה (בפועל, סכום הרמות של החיות באזור, אם היא השורש של האזור), מספר החיות באזור בו היא נמצאת, ואת האינדקס במערך של החיה שהיא שורש האזור שלה. נעשה שימוש ב חיבור קבוצות לפי גודל ובכיווץ מסלולים.
  המתודות אותן מממש מבנה הנתונים הן:
  - אתחול מערך – יצירת מערך דינמי בגודל n (מספר החיות), ואתחול כל חיה עם הרמה שלה. סיבוכיות זמן היא לפיכך O(n).
  - מציאת חיה – פונקציית find בשילוב עם חיבור לפי גודל וכיווץ מסלולים, סיבוכיות $O(\log^* n)$ משוערכת.
  - הסרת מחסום בין חיות – מדובר בפעולת Union בסיבוכיות $O(\log^* n)$ משוערכת.
  - החזרת מספר החיות באיזור – מוצאים את החיה בזמן $O(\log^* n)$ משוערך, ומחזירים את גודל האיזור ששמור בשורש של אותו איזור בזמן O(1) (גישה למערך). סה"כ $O(\log^* n)$ משוערך.
  - בדיקה האם שתי חיות באותו האיזור – מוצאים את שתי החיות בזמן $O(\log^* n)$ משוערך, ומחזירים האם לשתיהן אותה חיה בשורש. סה"כ $O(\log^* n)$ משוערך.

- טבלת ערבול המכילה מצביעים לזואולוגים. כל זואולוג מכיל משתנה עבור ה-ID שלו, עבור הותק שלו (מסופק בעת היצירה שלו), הרמה שלו, ה-ID של החיה שהוא אחראי עליה (או "לא אחראי" אם עוד לא אחראי על חיה), והאינדקס שלו בטבלת הערבול. הטבלה בגודל דינמי, עם double hashing. פונקציות הערבול הן כמוסבר בתרגול, כאשר אם בשל הגדלת המערך נוצר מצב שקיים מכנה משותף גדול מ-1 לגודל הטבלה ולפונקציה r(x), r(x) מוגדל ב-1 עד שהמכנה המשותף הוא 1 (כלומר המספרים זרים זה לזה).
  המתודות אותן מממש המבנה:
  - מציאת החיה שעליה אחראי זואולוג מסויים – בעזרת ה-ID של הזואולוג ניתן למצוא אותו בזמן O(1) ממוצע על הקלט, ולהסתכל בשדה של החיה שהוא אחראי עליה.

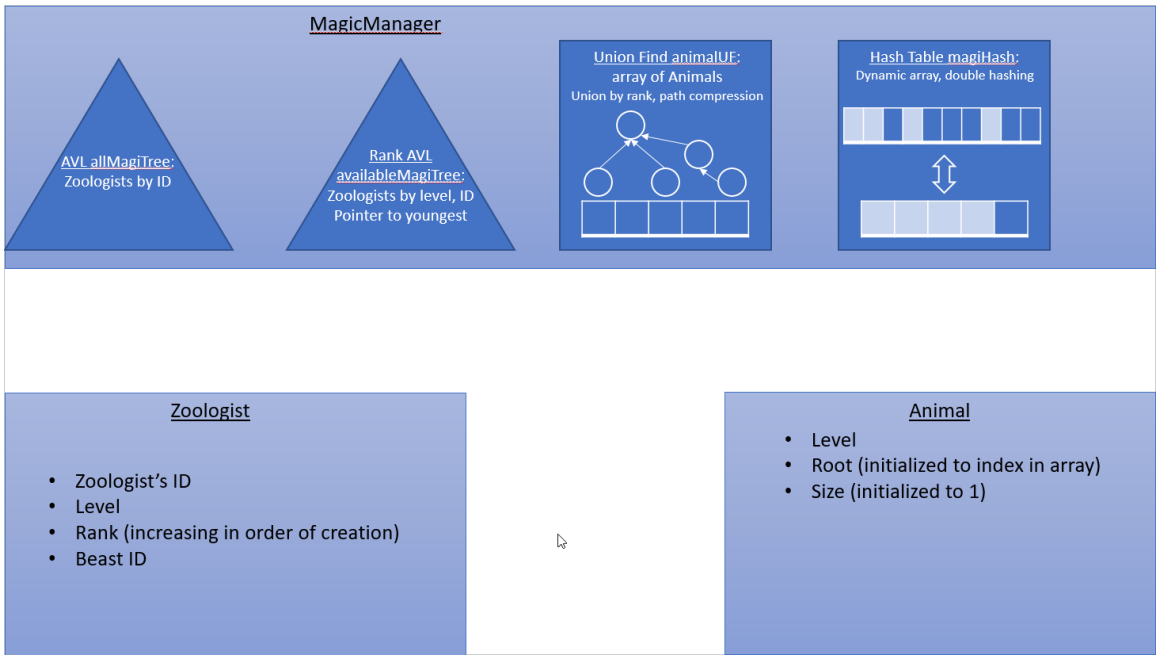
- availableMagiTree<Magi,Magi>RAVL: זהו עץ דרגות אשר מכיל את המאגים שלא הוקצו כאחראים על חיה מסויימת, מסודרים על פי הדרגה כקריטריון ראשון, ו-ID משני. בכל צומת שמור מידע נוסף שהוא מצביע למאגי הכי צעיר בתת העץ ,בעץ זה נמצאים רק המאגים אשר לא אחראים על אף איזור. בהוספה או הסרה הנתון על המאגי הכי צעיר יכול להשתנות מהצומת הרלוונטית ומעלה, ולכן מדובר ב$O(\log n)$ פעולות, לכן סיבוכיות פעולות אלה לא מושפעת.
  המתודות אותן מממש המבנה:
  - מציאת הזואולוג הצעיר ביותר שהרמה שלו מספיק גבוהה כדי להיות אחראי על איזור מסויים – בהנתן קריטריון כזה, נתחיל חיפוש בשורש העץ, ונשווה את הרמה שלו לקריטריון. נחלק לשני מקרים:
    - הרמה של הזואולוג קטנה מדי – נמשיך את החיפוש ימינה, אם אפשר. אם אי אפשר, נחזיר ערך ריק, כלומר אין זואולוג מתאים.

- הרמה של הזואולוג גדולה או שווה לקריטריון – במצב כזה אנחנו יודעים שהזואולוג הנוכחי מתאים, וכל מי שבבן הימני שלו יתאים (כי העץ ממויין לפי רמות), ובנוסף, יכול להיות שמישהו בבן השמאלי מתאים, בין אם כי הרמה שלו נמוכה יותר אך עדיין מספיק גבוהה, או שהרמה שלו שווה לזו של הזואולוג הנוכחי אך ה-ID שלו קטן יותר. לכן נשווה בין שלושה ערכים – הותק של הזואולוג הכי צעיר בבן הימני, הותק של הצומת הנוכחי, והותק של הזואולוג הכי צעיר בבן השמאלי שעונה על הקריטריון, אם יש כזה. נחזיר את הצעיר מביניהם. סך הכל מדובר ב(O(1 השוואות לאורך מסלול שאורכו חסום ע"י גובה העץ, ולכן מדובר בסיבוכיות של (O(logn.

- 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.

המתודות במבנה זה הינן סטנדרטיות בעץ AVL.

תיאור גרפי של מבנה הנתונים



סיבוכיות פונקציות של מבנה הנתונים

INIT: באתחול המערכת למעשה רק מאותחלים ארבעת מבני הנתונים לכן:

- אתחול של עץ דרגות ריק ב$O(1)$
- אתחול של עץ AVL ריק כנ"ל ב-$O(1)$
- אתחול של UNIONFIND כמו שראינו בהרצאה ב-$O(n)$
- אתחול 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 ו-double hashing). כמו שלמדנו זה נעשה ב-$O(1)$ משוערך בממוצע על הקלט.

לכן בסה"כ קיבלנו שעושים את זה ב$O(\log(k))$ משוערך בממוצע על הקלט, כמו שראינו בתרגול.

RemoveMagiZoologist: בהסרה מהמערכת יש קריאה ל-releaseMagi, אשר כמו שנראה למטה נעשה ב$O(\log(k))$ במקרה הגרוע. בנוסף יש הסרה משני עצים שכל אחת נעשית ב$O(\log(k))$. ההסרה מטבלת הערבול לא נעשית ע"י חיפוש בה, אלא בעזרת שדה ה-index שנשמר בזואולוג כשנוצר. כך מתאפשרת גישה ישירה אל התא בטבלה בו נמצא הזואולוג והדלקת הדגל deleted באותו תא, ב-$O(1)$. (הדגל deleted לפי שיטת double hashing).

לכן בסה"כ נקבל כמו ב-AddMagiZoologist שעומדים ב$O(\log(k))$ משוערך בממוצע על הקלט כמו שראינו בתרגול.

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

סה"כ הסיבוכיות $O(\log^*n)$.

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 ב$O(\log^*(n))$, כאשר בכל פעולת UNION שמנו את המידע "רמת הסיכון של האיזור" בשורש העץ ההפוך. לכן, לאחר שמצאנו את האיזור הרלוונטי החזרת מידע זה נעשית ב$O(1)$. כאשר יש בידינו מידע זה, נחפש בעזרתו את האיבר המינימלי שעומד בקריטריון זו בעץ availableMagiTree שכאמור מסודר לפי רמה כקריטריון ראשי ו-ID כקריטריון משני. תיאור אלגוריתם זה נמצא בתיאור העץ availableMagiTree. לאחר שמצאנו את המינימלי, לפי ה-index שלו נוכל לגשת למערך של טבלת הערבול ב$O(1)$ ולעדכן למאגי את החיה שהוא אחראי עליה, ולהוציא את המאגי מהעץ availableMagiTree ב$O(\log(k))$.

לכן סה"כ הסיבוכיות היא $O(\log(k)+\log^*(n))$.

ReleaseMagiZoologist: נמצא את המאגי בעץ allMagiTree ב$\log(k)$. דרך הindex- שנמצא אצלו נוריד ממנו את האחריות לחיה בטבלת הערבול ב$O(1)$. נוסיף אותו לעץ availableMagiTree ב$O(\log(k))$.

לכן בסה"כ ב-$O(\log(k))$ במקרה הגרוע.

GetCreatureOfMagi: נמצא את המאגי בטבלת הערבול זהו חיפוש בטבלת ערבול עם מערכים דינאמיים ו double hashing לכן נעשה ב$O(1)$ בממוצע על הקלט כפי שנלמד בהרצאה מכיוון שעידכנו את החיה שעליה אחראי המאגי בפעולות ReleaseMagiZoologist ו – AssignMagizoologistToCreature נוכל עכשיו למצוא את החיה המתאימה ב$O(1)$.

לכן בסה"כ זה נעשה ב$O(1)$ בממוצע על הקלט.

AreCreaturesInSameArea: זוהי למעשה רק פעולת FIND של unionfind שמתבצעת כמו שלמדנו ב-$O(\log^*(n))$. פעם עבור החיה הראשונה, ופעם עבור השניה. משווים את שורשי העצים, וזה ערך ההחזרה של הפונקציה.

לכן נעשה בסה"כ ב-$O(\log^*(n))$.

GetSizeOfArea: זוהי למעשה רק פעולת FIND של unionfind שמתבצעת כמו שלמדנו ב$O(\log^*(n))$ כאשר בראש העץ ההפוך עדכנו בכל פעולת UNION את מספר החיות שנמצאות באזור.

לכן נשיג זאת ב$O(1)$ 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.