מגישים: רן לוטם, דביר פרי

תיאור מילולי וגרפי של מבנה הנתונים:

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.

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.
* RAVL<Magi,Magi>availableMagiTree: 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 בO(logn) 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 בO(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.

תיאור גרפי של מבנה הנתונים:

