רטוב 2- חלק יבש

מגישים

מרואן סירייה 213769284
ג'וד מנסא 213716301

מבנה הנתונים שלנו:

מבנה הנתונים שלנו מכיל שלושה Hashtables.

הראשון יכיל את כל הפיראטים במערכת שלנו, הוא מאורגן באמצעות מבנה נתונים של LinkedList כדי לנהל את רשימת הפיראטים, כאשר כל פיראט מזוהה באמצעות מזהה ייחודי.

השני והשלישי יכילו את כל הציים, כאשר אחד מהם מיושם כמבנה של UnionFind, וכל צי הוא קבוצה ב-UnionFind.

מתודות עיקריות כמו insert,contains, ו-hashFunction משמשות להוספה וחיפוש של פיראטים וציים במערכת בזמן אמת. כל צי כולל מספר ספינות ומספר פיראטים המנוהלים באמצעות מחלקת Node, כאשר מתודת find מאפשרת למצוא את המנהיג של כל צי ולעדכן את הקשרים בין הציים. המערכת מספקת אפשרויות מורחבות כמו החזרת תכונות חשובות לכל צי ופיראט בזמן אמת, מעבר למבנה הבסיסי של הנתונים.

סיבוכיות מקום:

לכל פיראט או צי במערכת אנחנו מייצרים רק אובייקט אחד ושומרים בתוך Hashtables מצביעים לאובייקטים. n הוא מספר הפיראטים ו-m הוא מספר הציים, לכן סך הכל אובייקטים במערכת בכל זמן הריצה הוא m+n ולכן סיבוכיות המקום של התוכנית היא O(m+n).

הפעולות של המערכת:

לא הקצנו מקום באף פונקציה, לכן לכל אורך התוכנית, סיבוכיות המקום של כל הפעולות האלו הינה O(1).

1. oceans_t():

יוצר אובייקט מסוג oceans_t עם מספר פרמטרים: fleetTable, pirateTable ו-array. כל אחת מהפעולות המבוצעות בו לא תלויה בגודל של קלט כלשהו אלא מתבצעת במספר קבוע של פעולות: באתחול array יוצרים Hashtable בגודל קבוע (128), הפעולה הזאת נחשבת כבעלת סיבוכיות של O(1), שנית, באתחול הטבלאות fleetTable ו- pirateTable ישירות עם משתנים, גם הפעולה הזו נחשבת כבעלת סיבוכיות של O(1).

מכיוון שכל הפעולות בקטע הקוד הן בעלות סיבוכיות O(1), לכן סה"כ הסיבוכיות הכוללת של הבנאי היא גם O(1).

2. virtual ~oceans_t():

ההורס נועד לשחרר את הזיכרון שהוקצה דינמית עבור מבנה הנתונים oceans_t, שכולל Hashtable של ציים שמיוצגים על ידי מבנה ה-UnionFind.

ההורס עובר בלולאה על כל אחד מהאלמנטים:

- בכל איטרציה של הלולאה, ההורס קורא לפונקציה find(i) כדי למצוא את הצי המתאים ב-UnionFind.
- לאחר מציאת הצי, מתבצעת פעולת מחיקה delete fleet, שמסירה את הצי מהזיכרון.

לולאת ה- for רצה n פעמים (מספר הפיראטים), בכל איטרציה מתבצעת פעולת find והפעולה delete, כלומר הסיבוכיות של find כפולה במספר האיברים במבנה הנתונים של הציים שהוא מספר הציים m.

לכן סה"כ הסיבוכיות הכוללת של ההורס היא O(n+m).

3. StatusType add_fleet(int fleetId):

הפונקציה מבצעת מספר פעולות עיקריות כאשר מטרתה היא להוסיף צי חדש למערכת. תחילה בודקים את תקינות fleetId, זוהי פעולה המבוצעת בסיבוכיות של O(1). לאחר מכן, מתבצעת בדיקה האם הצי עם מזהה fleetId כבר קיים ב- Hashtable של הציים, פעולת החיפוש בטבלת HashTable מתבצעת בסיבוכיות משוערכת בממוצע של O(1). במידה והצי אינו קיים, הפונקציה מנסה להוסיף את הצי החדש ל-fleetTable. פעולת ההוספה ב-HashTable מתבצעת גם היא בסיבוכיות משוערכת בממוצע של O(1). במקביל, מתבצעת הוספה של הצי למבנה יחד עם יצירת אובייקט חדש מסוג UnionFind. פעולה זו מתבצעת בסיבוכיות של O(1) בממוצע.

לבסוף, הפונקציה מחזירה את סטטוס הפעולה, מה שמתבצע גם כן בסיבוכיות של O(1).

בהתחשב בכך שכל הפעולות העיקריות בפונקציה הן בעלות סיבוכיות של O(1) משוערך בממוצע, סה"כ הסיבוכיות הכוללת של פעולת הוספת צי היא O(1) משוערך בממוצע על הקלט.

4. StatusType add_pirate(int pirateId,int fleetId):

תחילה בודקים את תקינות הקלט, ובודקים אם מזהה הפיראט והצי הם חיוביים, פעולה שמתבצעת בסיבוכיות O(1). מבצעים חיפוש בטבלת הציים (HashTable), כדי לוודא שהצי קיים במערכת. פעולה זו מתבצעת בסיבוכיות של O(1) בממוצע על הקלט. בהמשך, הפונקציה בודקת אם הפיראט כבר קיים בטבלת הפיראטים, שגם היא HashTable, ולכן החיפוש מתבצע שוב בסיבוכיות O(1) בממוצע.

לאחר הבדיקות הללו, הפונקציה בודקת אם מזהה הצי תואם לצי שנמצא במבנה הנתונים UnionFind. פעולה זו כוללת קריאה לפונקציה find של UnionFind, שהיא בעלת סיבוכיות משוערכת של O(log*m), כאשר m הוא מספר הציים במערכת. הוספת הפיראט ל-pirateTable והעדכון של הצי במערכת מתבצעים בסיבוכיות O(1) בממוצע.

לכן סה"כ הסיבוכיות הכוללת של פעולת הוספת פיראט היא O(log*m) משוערך בממוצע על הקלט.

5. StatusType pay_pirate(int pirateId,int salary):

תחילה, הפונקציה בודקת את תקינות הקלט, כלומר, אם מזהה הפיראט והשכר הם חיוביים, פעולה שמתבצעת בסיבוכיות O(1). לאחר מכן, מתבצעת בדיקה אם הפיראט קיים בטבלת הפיראטים, שהיא מבנה HashTable. חיפוש ב- HashTable מתבצע בסיבוכיות O(1) בממוצע. לאחר מכן, הפונקציה מעדכנת את השכר של הפיראט באמצעות קריאה לפונקציה update_salary, שהיא גם כן בעלת סיבוכיות O(1) בממוצע. לבסוף, הפונקציה מחזירה את התוצאה המתאימה בסיבוכיות O(1).

בשל העובדה שכל הפעולות בפונקציה מתבצעות בסיבוכיות O(1) בממוצע, סה"כ הסיבוכיות הכוללת של פעולת התשלום לפיראט היא O(1) בממוצע על הקלט.

6. output_t < int > num_ships_for_fleet(int fleetId):

תחילה, הפונקציה בודקת את תקינות הקלט, כלומר, אם מזהה הצי הוא חיובי, פעולה שמתבצעת בסיבוכיות O(1). לאחר מכן, היא בודקת אם הצי קיים בטבלת הציים, שהיא מבנה HashTable. פעולה זו מתבצעת בסיבוכיות O(1) בממוצע.

בהמשך, הפונקציה בודקת אם הצי נמצא במבנה הנתונים UnionFind על ידי קריאה לפונקציה find, אשר מזהה את מנהיג הקבוצה של הצי. פעולת find במבנה UnionFind מתבצעת בסיבוכיות O(log*m) בממוצע, כאשר m הוא מספר הציים במערכת. לבסוף, הפונקציה מחזירה את מספר הספינות בצי שנמצא, פעולה שמתבצעת בסיבוכיות O(1).

לכן סה"כ הסיבוכיות הכוללת של הפעולה היא O(log*m) משוערך בממוצע על הקלט.

7. output_t < int > get_pirate_money(int pirateId):

תחילה, הפונקציה בודקת את תקינות הקלט על ידי בדיקה אם מזהה הפיראט הוא חיובי, פעולה שמתבצעת בסיבוכיות O(1). לאחר מכן, היא בודקת אם הפיראט קיים בטבלת הפיראטים. החיפוש ב-HashTable מתבצע בסיבוכיות O(1) בממוצע.

במידה והפיראט קיים, הפונקציה שולפת את סכום הכסף המשויך לו באמצעות קריאה לפונקציה getMoney של ה-pirateTable, שגם היא מתבצעת בסיבוכיות O(1) בממוצע, שכן מדובר בפעולה של גישה ישירה לערך בטבלת ה-Hashtable. לבסוף, הפונקציה מחזירה את סכום הכסף שנמצא.

כל הפעולות הללו מתבצעות בסיבוכיות O(1) בממוצע על הקלט, לכן סה"כ הסיבוכיות הכוללת של הפעולה היא O(1) בממוצע על הקלט.

8. StatusType unite_fleets(int fleetId1,int fleetId2):

הפונקציה unite_fleets מאחדת שני ציים במערכת. תחילה, היא מבצעת בדיקת תקינות קלט, הכוללת בדיקה אם מזהי הציים fleetId1 ו- fleetId2 הם חיוביים ושונים זה מזה. פעולה זו מתבצעת בסיבוכיות O(1). לאחר מכן, היא בודקת אם שני הציים קיימים בטבלת הציים (fleetTable), שהיא מבנה HashTable. הבדיקה הזו מתבצעת בסיבוכיות O(1) בממוצע. בהמשך, הפונקציה מוודאת שהציים שמורים נכון במבנה ה-UnionFind באמצעות קריאה לפונקציה findHead, שמחזירה את מנהיג הקבוצה של הצי. פעולה זו מתבצעת בסיבוכיות O(log*m) כאשר m הוא מספר הציים במערכת, שכן מדובר במבנה נתונים עם שיפורים כמו כיווץ מסלולים.
לאחר מכן, הפונקציה מאחדת את שני הציים באמצעות פעולת Union במבנה UnionFind, שגם היא מתבצעת בסיבוכיות O(log*m) בממוצע, והיא מאחדת לפי שיפור "איחוד לפי גודל" של ה- UnionFind. בסופו של דבר, הפונקציה מחזירה את התוצאה המתאימה בסיבוכיות O(1).
לכן, הסיבוכיות הכוללת של הפונקציה unite_fleets היא O(log*m) בממוצע, לכן סה"כ הסיבוכיות הכוללת של פעולת איחוד הציים היא O(log*m) משוערך בממוצע על הקלט.

9. StatusType pirate_argument(int pirateId1,int pirateId2):

הפונקציה pirate_argument מיועדת לבחון ויכוח בין שני פיראטים, להשוות בין הדירוג שלהם ולהתאים את השכר בהתאם להבדלי הדירוג. הפונקציה מתחילה בבדיקת תקינות הקלט, כלומר אם מזהי הפיראטים pirateId1 ו-pirateId2 הם חיוביים ושונים זה מזה. פעולה זו מתבצעת בסיבוכיות O(1). לאחר מכן, מתבצעת בדיקה אם שני הפיראטים קיימים בטבלת הפיראטים pirateTable, שהיא מבנה HashTable. החיפוש בטבלה מתבצע בסיבוכיות O(1) בממוצע.
בהמשך, הפונקציה מאתרת את הפיראטים ברשימות המקושרות שלהם בטבלת ה-HashTable. פעולה זו גם כן מתבצעת בסיבוכיות O(1) בממוצע.
לאחר מכן, הפונקציה בודקת אם שני הפיראטים שייכים לאותו צי על ידי קריאה לפונקציה findHead במבנה ה-UnionFind, שמחזירה את מנהיג הקבוצה של כל צי. פעולה זו מתבצעת בסיבוכיות O(log*m), כאשר m הוא מספר הציים במערכת, בגלל השיפורים שמבנה ה-UnionFind מספק (כמו כיווץ מסלולים).
לבסוף, הפונקציה מחשבת את הפרש הדירוגים בין הפיראטים ומעדכנת את השכר שלהם בהתאם, פעולות שמתבצעות בסיבוכיות O(1) בממוצע.
לכן סה"כ הסיבוכיות הכוללת של הפעולה היא O(log*m) משוערך בממוצע על הקלט.