תרגיל בית רטוב 2 – יבש

הגשת:

רזי חליחל – 325056240

יארא זועבי – 212264048

תאריך:
3.8.2024

מבני נתונים 1 - 02340218

תוכן עניינים

רעיון כללי

בחרנו לממש את המחלקה הדרושה – Oceans_t – בעזרת שתי מחלקות חדשות (מחלקת פיראטים ומחלקת ציים) המימוש היה בעזרת שתי טבלאות ערבול דינמיות אשר הראשונה מכילה מידע על הפיראטים והשנייה מכילה מידע על הציים אשר מאחסנים אותם בעצים הפוכים ע"י union find כאשר בשורש של כל עץ הפוך (המעיד על צי אחד) שומרים מספר הספינות בצי. מימשנו את טבלת העירבול בעזרת מחלקה גינירת על מנת להשתמש בה בשתי המחלקות השונות שהוספנו.

מחלקת הפיראטים: כל אובייקט מהמחלקה ייצג פיראט אחד ייחודי בעל Id מסוים. לפיראט יש מצביע לצי שהוא שייך אליו.

מחלקת הציים: כל אובייקט מהמחלקה ייצג צי אחד ייחודי בעל Id מסוים. לכל צי בעץ ההפוך של טבלת הערבול יש מצביע לצי האב אם קיים, בנוסף לכל צי יש rank_modifier שהיא תכונה משותפת לכל הפיראטים בצי שבעזרתה ניתן לחשב את דרגת כל פיראט בצי.

מחלקת צומת בטבלת הערבול: כל צומת יכיל

- מצביעים לאיבר שמגיעה אחריו באותו תא בטבלת הערבול (אם קיים)
- מצביע לאובייקט שהוא מייצג

מחלקת טבלת הערבול: המחלקה תכיל מצביע למערך (טבלת הערבול) ופונקציונליות הוספת וחיפוש איברים בטבלה.

מחלקת ה Oceans_t: תכיל שני מערכי ערבול דינמיים, מערך פיראטים ומערך ציים שיכילו את כלל הפיראטים והציים.

עבור n פיראטים ו- m ציים, במימוש זה אנו יכולים לגשת לכל פיראט או צי בסיבוכיות משוערכת של $O(1)$ כפי שראינו בכיתה.

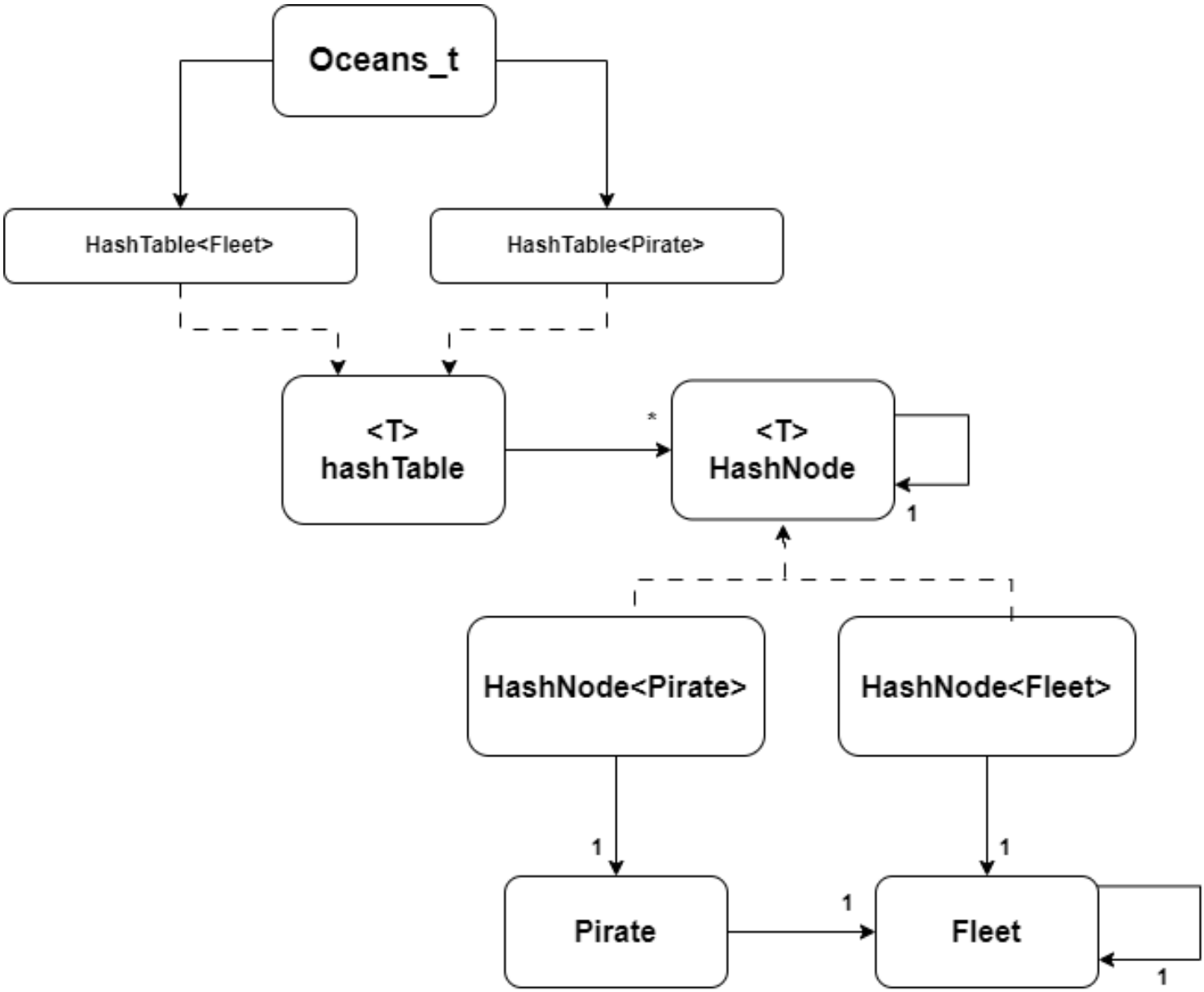
<u>עבור סיבוכיות המקום:</u>

אנו שומרים על כל פיראט פעם אחת ועל כל צי פעם אחת. לכל צי נשמור צומת ייחודית שנייצג אותה בעץ ההופכי אשר נצביע עליו מהתא המתאים בטבלת העירבול. ולכל פיראט נשמור אותו בתא מתאים בטבלת הערבול.

בסה"כ לכל צי יש לנו מצביע לתא המתאים שלו בטבלת הערבול ומצביע מהתא לצומת המתאימה בעץ ההופכי אשר שומרת מספר קבוע של שדות בעלי גודל חסום אזי נשמור על $O(m)$ זיכרון. ולכל פיראט יש מצביע לתא המתאים בטבלת הערבול אשר שומר על מספר שדות קבוע אזי נדרש $O(n)$ זיכרון. ביחד נדרש בסה"כ $O(n+m)$ זכרון.

בנוסף ברוב הפונקציות שמימשנו לא הגדרנו משתנים מקומיים נוספים חוץ ממספר סופי של משתני עזר בעלי מקום סופי כלומר סיבוכיות המקום עבורם היא $O(1)$. אבל עבור הפונקציה הדואגת להגדלת טבלת הערבול הדינמי היינו צריכים להקצות זיכרון חדש בגודל פעמיים n או m (תלוי בסוג המערך) אזי סה"כ הכל עבור פונקציה זו יוצא שסיבוכיות המקום $O(n+m)$, כלומר סיבוכיות המקום הכוללת של התוכנית נשארת $O(n+m)$.

UML


Oceans_t
HashTable<Fleet>
HashTable<Pirate>
<T>
hashTable
*
<T>
HashNode
1
HashNode<Pirate>
HashNode<Fleet>
1
1
Pirate
1
Fleet
1

פונקציות ראשיות

בפרק זה נפרט לכל פונקציה את האלגוריתם שעומד מאחוריה ואיך הוא עומד בסיבוכיות הזמן והמקום הדרושים. עבור פרק זה נשתמש בטענות על סיבוכיות פונקציות העזר שמימשנו אותם שנוכיח את הסיבוכיות שלהם בפרק הבא.

Oceans_t() – Constructor

משתמשים בבנאי הדיפולטיבי אשר קורא לבנאי הדיפולטיבי של מחלקת טבלת הערבול אשר מאתחל שתי טבלאות ערבול ריקות בגודל קבוע (בחירה שרירותית של הגודל), פעולה זו דורשת סיבוכיות זמן של $O(1)$ במקרה הגרוע.

~Oceans_t() – Destructor

ההורס של המחלקה ממומש ע״י ההורס הדיפולטי של המחלקה משום שכל הזיכרון ממומש ע״י מצביעים חכמים ומחלקות עם הורסים דיפולטים. ההורס יקרא להורס הדיפולטי של טבלאות הערבול אשר מוגדר למחוק את המערך שלו, ומשוש שאיברי המערך הם מצביעים חכמים אז הם ישתחררו אוטומטית. תהליך זה יעבור על כל מצביע חכם קיים ועל שתי הטבלאות כך שבסה״כ ישנם מצביעים כמספר הפיראטים ומספר הציים ושתי הטבלאות בסדר גודל של הפיראטים והציים בהתאם ולכן הסיבוכיות במקרה הגרוע היא $O(m+n)$.

add_fleet

כדי להוסיף צי, נשתמש בפונקציית ההוספה לטבלת הערבול (insert) אשר דורשת סיבוכיות זמן משוערכת בממוצע של $O(1)$ כפי שראינו בכיתה.

add_pirate

כדי להוסיף פיראט, נשתמש בפונקציית ההוספה לטבלת הערבול (insert) אשר דורשת סיבוכיות זמן משוערכת בממוצע של $O(1)$ כפי שראינו בכיתה.

אבל בנוסף נצטרך למצוא את הצי ששיך אליו הפיראט בעזרת פונקציית העזר get של טבלת הערבול אשר דורשת סיבוכיות זמן $O(1)$ משוערך בממוצע על הקלט (ראינו בהרצאה), ואת דרגתו של הפיראט בעזרת פונקציית העזר getHead שמחפשת את שורש הקבוצה (העץ ההפוך) של ה Union ששיך ה-fleet אליה כדי לדעת את סכ״ה מספר הפיראטים, פונקציה זו דורשת סיבוכיות זמן $O(\log^* m)$ משוערך בממוצע על הקלט. אזי סה״כ נקבל שסיבוכיות הזמן היא $O(\log^* m)$ משוערך בממוצע על הקלט.

pay_pirate

נשתמש בפונקציית עזר get שדורשת סיבוכיות זמן $O(1)$ משוערך בממוצע על הקלט אשר מחזירה מצביע לתא הפיראט שמחפשים בטבלת הערבול, ובעזרת setter נעדכן את ערך המטבעות אצל הפיראט. אזי סה״כ נקבל שסיבוכיות הזמן היא $O(1)$ משוערך בממוצע על הקלט.

num_ships_for_fleet

מכיוון שבכל עץ הופכי ב-Union Find שומרים את מספר הספינות בשורש, אזי נחפש את השורש של הצי הנתון בעזרת פונקציית העזר getHead שדורשת סיבוכיות זמן $O(\log^* m)$ משוערך בממוצע על הקלט.

get_pirate_money

נשתמש בפונקציית עזר get של טבלת ערבול הפיראטים, שדורשת סיבוכיות זמן $O(1)$ בממוצע על הקלט אשר מחזירה מצביע לפיראט שמחפשים בטבלת הערבול, ובעזרת getter נחזיר את ערך המטבעות אצל הפיראט. אזי סה"כ נקבל שסיבוכיות הזמן היא $O(1)$ בממוצע על הקלט.

unite_fleets

בפונקציה זו נרצה לאחד שני ציים ע"י חיבור שני עצים הפוכים. נעשה זאת ע"י מציאת שני השורשים של העצים ההפוכים בעזרת הקריאה פעמיים לפונקציית העזר getHead אשר כל קריאה לפונקציה זו דורשת סיבוכיות זמן $O(\log^* m)$ משוערך בממוצע על הקלט. בנוסף נעשה מספר קבוע וסופי של השוואות ועדכונים בשדות של שני השורשים (כגון מספר הפיראטים, ה-rank_modifer, מספר הספינות...) אשר דורשות $O(1)$ זמן במקרה הגרוע, ולכן סה"כ נקבל שסיבוכיות הזמן היא $O(\log^* m)$ משוערך בממוצע על הקלט.

pirate_argument

בפונקציה זו נצטרך לחשב את הדירוג של כל פיראט ולכן נצטרך למצוא את השורשים של הציים השייכים להם שני הפיראטים (כי צריכים את ה-rank_modifier ששמור בשורש בנוסף ל-rank_modifier של הצי המקורי שלו שהינו מצביע אליו על מנת לחשב דרגת הפיראט), נעשה זאת בעזרת שתי קריאות לפונקציית העזר getHead אשר כל קריאה לפונקציה זו דורשת סיבוכיות זמן $O(\log^* m)$ משוערך בממוצע על הקלט. בנוסף נעשה מספר סופי של חישובים והשוואות על מנת לחשב את הדרגה של כל פיראט ואת הפרש המטבעות בין שני הפיראטים ובסוף נעדכן את המטבעות של כל פיראט בעזרת שתי קריאות ל- setters ולכן סה"כ פעולות אלו דורשות $O(1)$ זמן. אזי סה"כ מתקיים כי סיבוכיות הזמן היא $O(\log^* m)$ משוערך בממוצע על הקלט.

פונקציות ומבני עזר

Getters & Setters

כל מתודות ה-get וה-set של המחלקות שהגדרנו פועלות ב-$O(1)$ מפני ושהם ניגשות אך ורק לשדה המחלקה שלהם.

insert

פונקציה זו מכניסה איבר כלשהו לטבלת הערבול בעזרת פונקציית ערבול מודלו גודל המערך של טבלת הערבול, וכפי שראינו בהרצאה ולפי הנחת הפיזור האחיד פעולה זו דורשת סיבוכיות זמן משוערך בממוצע של $O(1)$. בנוסף, עבור כל הכנסה של איבר נדאג לבדוק את גודל המערך ובמקרה ושיש להגדיל את המערך מתקיים כי פעולה זו דורשת סיבוכיות זמן משוערך בממוצע של $O(1)$ (נסביר פעולה זו בהמשך). אזי סה"כ מתקיים כי פעולת ההכנסה לטבלת הערבול דורשת זמן משוערך בממוצע של $O(1)$.

Get

פונקציה זו מחזירה מצביע לאיבר שמחפשים עליו בטבלת הערבול, אשר עושה זאת בעזרת פונקציית ערבול מודלו גודל המערך של טבלת הערבול, וכפי שראינו בהרצאה ולפי הנחת הפיזור האחיד פעולה זו דורשת סיבוכיות זמן משוערך בממוצע של $O(1)$.

getHead

פונקצייה זו מחזירה מצביע לשורש של עץ הפוך, ומכיוון שדאגנו לממש את העצים ההפוכים כפי שראינו בכיתה (איחוד לפי גודל וכיווץ מסלולים) כך שמחברים את כל הצמתים במסלול לשורש ישירות לאחר מציאת השורש (ועדכון שדה הrank_modifier- כפי שראינו בתרגול) מתקיים שפעולה זו דורשת סיבוכיות זמן $O(\log^* m)$ משוערך בממוצע על הקלט.

טבלאות ערבול דינמיות

משתמשים בטבלאות ערבול דינמיות על מנת להבטיח מקום פנוי בטבלת הערבול להכנסת איברים חדשים לאורך כל התוכנית, ולכן עבור כל הכנסה של איבר במערך נבדוק את גודל המקום התפוס במערך, אם גודל המקום התפוס שווה ל- 0.75 מגודל כל המערך אזי נצטרך להגדיר מערך חדש בגודל של פי שתיים מגודל המערך הנוכחי ואז נעביר את כל האיברים למערך החדש. פעולה זו דורשת $O(n)$ זמן במקרה הגרוע (עבור מערך בגודל n) אבל ראינו בכיתה שפעולה זו דורשת סיבוכיות זמן משוערך בממוצע של $O(1)$.

חישוב סיבוכיות משוערכת

עבור הפונקציות בעלות סיבוכיות משוערכת $O(1)$ (הוספת צי ושילום לפיראט) נדרש חיפוש בטבלת ערבול וכמו שראינו בהרצאה חיפוש זה דורש סיבוכיות משוערכת של $O(1)$.

עבור פונקציות בעלות סיבוכיות משוערכת $O(\log^* m)$ (הוספת פיראט, ספירת מספר הספינות, איחוד ציים וסכסוך פיראטים) כל אחת מהם משמשת בפונקציית ה-getHead ששקולה ל-Find של Union Find אשר ראינו בכיתה, ופונקציית האיחוד בנוסף פועלת כמו Union שראינו. לכן עבור סדרת k קריאות של הפונקציות אנו מקבלים במקרה הגרוע $O(k \cdot log^* m)$ לכן הסיבוכיות המשוערכת בממוצע על הקלט תהיה $O(\log^* m)$.