רטוב 2 - חלק יבש

16:00 Monday, 5 June 2023

נממש את חברת התקליטים בעזרת שלושה מבני נתונים

חברי מועדון

עץ AVL של חברי מועדון שמחלחל פרסים

נשתמש במימוש של עץ AVL סטנדרטי דומה לזה שנלמד בהרצאות עם תוספת של ערך מיוחד שנשמר בכל צומת השומר פרס שהתקבל על ידי כל התת עץ של הצומת. בנוסף נשמור את הטווח מינימום ומקסימום של תת עץ של צומת כערכי עזר שיעזרו לנו לממש את פעולת חילוק הפרס.

ניתן יהיה להשתמש בעץ זה על מנת לחלק פרס לכל החברי מועדון בטווח מספרי זהות מסוים. פעולה זו תמומש באופן רקורסיבי כאשר נתחיל מהשורש. כאשר נגיע לצומת שכל התת עץ שלו מוכל בטווח אז נשמור את הפרס שמגיע לכל התת עץ בצומת הנוכחי. אם אף חלק מהתת עץ לא מוכל בטווח לא נבצע כלום ואם רק חלק מהתת עץ הנתון מוכל בטווח אז נתקדם באופן רקורסיבי לשני הבנים (ונעדכן את ההנחה של חבר המועדון של הצומת הנוכחי אם הוא בטווח). באופן זה נתקדם באופן רקורסיבי אל עבר שתי הקצוות של הטווח המבוקש ונעדכן לאורך הדרך את כל ערכי הפרסים הרלוונטיים (ערך פרס של תת עץ כשבכל תת העץ מקבל פרס ורק ערך פרס של חבר מועדון עצמו כשהוא בן אבל לא בהכרח על התת עץ שלו). כיוון שתחום הפרס הוא חלק רציף של העץ קיימים לו למעשה רק גבולות גזרה ימניים ושמאליים. הגעה לכל אחד מהם תעשה בסיבוכיות $O(\log n)$ ולכן גם הסיבוכיות הכוללת של פעולת חלוקת הפרס תיהיה $O(\log n)$.

לעץ זה יהיה ניתן להכניס איברים (צירוף לקוח למועדון) ומציאת איברים (בירור הוצאות של חבר מועדון) בדומה לעץ AVL רגיל עם ההבדל שבעת גישה לאיבר במיקום מסוים בעץ יתבצע חלחול של ערכי פרס אל הצומת הזה. המשמעות היא שכאשר אנחנו מסיירים בצמתים בדרך אל צומת היעד אנו נאפס את ערך הפרס בכל צומת ו"נעביר אותו" אל החבר מועדון הנמצא בצומת ואל צמתי הבנים שלו. באופן זה אף צומת לא יאבד את הפרס שמגיע לו וגם כשנגיע אל הצומת אליו אנו מנסים לגשת כל הפרסים הרלוונטיים יחלחלו עד אליו וניתן יהיה לחשב את סך החובות שלו בהתחשב בפרסים שהוא קיבל. באופן דומה כאשר מכניסים חבר מועדון חדש לעץ נחלחל את הפרסים כך שחבר המועדון לא יקבל פרסים באופן רטרואקטיבית שחולקו לפני כניסתו למועדון. בנוסף נחלחל את הערכים של הצומת ושל בניו כאשר נבצע AVL rotation כדי למנוע בלבול של הפרסים בעת הפעולה. כיוון שכל פעולת חלחול היא בזמן קבוע וגובה עץ AVL הוא $O(\log n)$ ביצוע כל פעולות החלחול הרלוונטיות יהיה $O(\log n)$.

הבדל נוסף שיהיה ביחס לעץ AVL רגיל הוא שבעת הכנסה נצטרך לעדכן את ערכי טווח מינימום וטווח מקסימום של תת העץ בעת הוספת חבר מועדון. המינימום והמקסימום של צומת חדש בלי ילדים יהיה הערך של הצומת עצמו. טווח המינימום של צומת עם ילדים מוגדר להיות אותו הערך של הבן השמאלי של הצומת אם הוא קיים ושל הצומת עצמו אחרת. בזמן שטווח המקסימום מוגדר באופן סימטרי עבור המקסימום של הבן הימני. כל עדכון כזה נעשה בזמן קבוע. לפי הגדרה זו ניתן לעדכן את כל הערכים במסלול בין הצומת שהוכנס ועד לשורש העץ ב$O(\log n)$ כיוון שגובה עץ AVL הוא מסיבוכיות זו. במקרה של AVL rotation נצטרך לעדכן את הערכי מינימום והמקסימום של הצמתים המעורבים באופן דומה ב$O(1)$.

סך הכל הוספנו פעולות ב$O(\log n)$ אל פעולת ההכנסה בנוסף לאותה סיבוכיות של הפעולות הסטנדרטיות בעץ AVL ולכן בסך הכל הפעולות האלה יקחו $O(\log n)$ בעץ.

לקוחות

עבור אכסון הלקוחות נשתמש במבנה נתונים Hash Table וכפי שנלמד בהרצאות - פעולות find , insert, remove מתבצעות בזמן ממוצע על הקלט) . $O(1)$במקרה שלנו דרישות התרגיל לא מכילות הוצאה של לקוחות לכן לא מימשנו.(remove

מימוש: Hash Table

(1מכיוון שהמפתח של האיברים הם ID ניתן להניח את הנחת הפיזור הפשוט, כלומר אין תלות בין המפתחות. בחרנו בפונקצית ערבול ID mode m כאשר m הוא גודל הטבלה (שומרת על הנחת הפיזור האחיד הפשוט simple hash uniform, ממפה איבר אקראי מהתחום בהסתברות אחידה m/1 לכל אחד מהתאים בטווח, ובאופן בלתי תלוי בשאר האיברים.

(2על מנת לשמור על הסיבוכיות הנחשקת נבחר פקטור עומס קטן שווה ל 1 , כלומר בכל רגע נתון גודל הטבלה יגדל כך שתמיד יהיה גדול או שווה לכמות הלקוחות בה.

(3את הטבלה נממש בעזרת מערך דינמי כפי שנלמד בתרגול, והפיתרון שלנו עבור התנגשויות ערבול הוא שכל תא במערך יכיל עץ AVL שבו יוחזקו האיברים שהתמפו לאותו תא ע"י פונקציית הערבול. על מנת לשמור על פקטור

העומס תקין' בכל רגע שבו כמות האיברים תהיה שווה לכמות התאים בטבלה נרחיב את הטבלה פי 2 באופן דינמי. הרחבה דינמית זאת מתבצעת ע"י אתחול טבלה חדשה כפולה בגודל, ומעבר על כל התאים בטבלה הישנה ובהם מעבר על כל איברי העץ וערבולם לטבלה החדשה.

הדרישות הן $O(1)$ **משוערך** בממוצע על הקלט. בגלל שבתוחלת בכל תא יש איבר אחד גישה אליו היא ב $O(1)$משוערך בממוצע על הקלט, והפעולה הגדולה של ההגדלה הדינמית מתבצעת אחת להמון פעולות והיא "נבלעת" בין יתר הפעולות הקטנות ונשמרת הסיבוכיות המשוערכת .

תקליטים

מבנה union find של תקליטים ששומר גובה
נממש מבנה union find בדומה לזה שהוצג בבעיית הארגזים בתרגול. ניקח מימוש unionfind סטנדרטי כפי שהעובר בתרגול של עץ הפוך שהגישה לכל איבר היא במערך. נגדיר כל איבר להיות טיפוס שכולל את כל הפרטים הרלוונטים של התקליט, השדות הרגילים של unionfind (מצביע להורה ומספר חברי הקבוצה בשורש) ומספר שדות מיוחדים.

- delta - ההפרש בין הגובה של צומת לבין גובה צומת ההורה שלו בעץ
- stack_height - ערך שיהיה אקטיבי (כלומר בשימוש) אך ורק עבור שורש של ערימה כלשהי ויהיה שווה לגובה של כל הערימה ביחד.
- stack_count - ערך שיהיה אקטיבי אך ורק עבור שורש של ערימה. הערך יהיה שווה לעומדה שעליה כל הצמתים בקבוצה נערמים.

כל פעם שנשים ערימה של תקליטים על ערימה של תקליטים נבצע שתי פעולות find ופעולת union על מנת לחבר את השורשים כפי שנלמד. בעזרת המימוש האופטימלי (קיצור דרך + חיבור קבוצה קטנה אל גדולה) מתקבל סיבוכיות $O(\log^* n)$ משוערכת עבור הפעולה. בנוסף לפעולות הללו אנחנו גם נעדכן את השדות המיוחדים שלנו על מנת לשמור את תקינותם.

- אם נשים את קבוצה a על קבוצה b והקבוצה b עם יותר איברים מa אז השורש של a יצביע לשורש של b. נעדכן את ערך הdelta של השורש שלa להיות גדול יותר בגובה של ערימה b. המשמעות של זה תיהיה שהגובה האמיתי של כל תקליט שמוביל לשורש של a (שזה כל תקליט בערימת a) יעלה בגובה הכולל של ערימה b.
- אם נשים את קבוצה a על קבוצה b והקבוצה a עם יותר או אותו מספר של איברים מb אז השורש של b יצביע לשורש של a.
נעדכן את ערך הdelta של השורש שלa להיות גדול יותר בגובה שלb אבל גם נחסר מהdelta של השורש של b את הdelta של a. שתי הפעולות הללו ביחד יגרמו לכך שהגובה של כל הצמתים שהיו לפני זה בקבוצה a יעלה בגובה שלb בזמן שהגובה של איברי קבוצה b לשעבר לא ישתנה.
- בכל פעם שנבצע קיצור מסלול לשורש נעדכן את הdelta של הצומת להיות הסכום של כל הדרך כדי לשמר את הגובה של התקליט.

עבור השורש החדש של הקבוצה המאוחדת

- נעדכן את ערך גובה הערימה הכולל להיות הסכום של הגבהים הכוללים של שני הערימות.
- נעדכן את העמודה להיות העמודה של הקבוצה שעליה הערימו את הערימה השנייה.

כיוון שכל הפעולות הללו הן בסיבוכיות $O(1)$ זה אומר שהסיבוכיות של הפעולה של לערום ערימה אחת של תקליטים על אחרת זהה לסיבוכיות של אותה הפעולה במבנה unionfind רגיל ולכן הסיבוכיות נשארת $O(\log^* n)$.

פעולות של החברה

newMonth
נקצה unionfind של תקליטים ב$\mathrm{O(m)}$ ונשחרר את המבנה הקיים ב$\mathrm{O(m_{prev})}$. נאפס את ההוצאות של כל החברי מועדון על ידי סיור בעץ חברי המועדון ב$\mathrm{O(n)}$. סך הכל סיבוכיות כוללת תיהיה $O(m_{prev} + m + n) = O(\max(m_{prev}, m) + n)$ שזה תואם את הסיבוכיות הנדרשת (לפי תשובה בפיאצה).

addCustomer
נוסיף לקוח ל hash table של הלקוחות בעזרת פעולת ההכנסה שנעשית בסיבוכיות משוערכת של $O(1)$.

getPhone, isMember
נשיג מידע רלוונטי על לקוח על ידי כך שנמצא אותו ב hash tableשל הלקוחות בסיבוכיות משוערכת של $O(1)$ ואז נחזיר את המידע.

makeMember

נהפוך לקוח לחבר מועדון על ידי כך שתחילה נמצא אותו ב hash table של הלקוחות בסיבוכיות זמן גרוע של $\mathrm{O}(\log n)$. זה קורה בסיבוכיות הזאת מכיוון שגישה לתא המתאים במערך היא ב $\mathrm{O}(1)$ ובתא יש עץ AVL שחיפוש בו בסיבוכיות זמן גרוע של $\mathrm{O}(\log n)$. ואז נכניס מצביע ללקוח לעץ חברי המועדון בסיבוכיות זמן גרוע של $\mathrm{O}(\log n_{members})$. כיוון שחברי מועדון זה תת קבוצה של כל הלקוחות אז הסיבוכיות זמן גרועה כוללת של שני הפעולות היא $O(\log n)$ כנדרש.

buyRecord

ניגש למידע על תקליט במערך של הunionfind ב$\mathrm{O}(1)$ ניגש ללקוח ב hash table הלקוחות בסיבוכיות זמן גרוע של $O(\log n)$. זה קורה בסיבוכיות הזאת מכיוון שגישה לתא המתאים במערך היא ב $\mathrm{O}(1)$ ובתא יש עץ AVL שחיפוש בו בסיבוכיות זמן גרוע של $\mathrm{O}(\log n)$. נעלה את מונה הקניות בתקליט ואם הלקוח הוא חבר נעלה לו את החובות במחיר התקליט. סך כל הסיבוכיות הכוללת של $O(\log n)$

addPrize

נשתמש בפעולה המיוחדת שהוגדרה על עץ חברי המועדון על מנת לחלק הנחה לכל חברי המועדון בטווח הנתון ב$O(\log n)$

getExpenses

ניגש למידע של חבר מועדון בעץ חברי המועדון ב$O(\log n)$ ונחזיר את החובות שלו.

putOnTop

נבצע פעולת union של unionfind בין שני הקבוצות של שני התקליטים ונעדכן את הגבהים להיות תקינים בסיבוכיות משוערכת $O(\log^* n)$ כפי שתואר בחלק על מבנה התקליטים.

GetPlace

נבצע פעולת find על התקליט. נמצא את ערך העמודה שרשום בשורש ונקצר את המסלול לאורך הדרך בסיבוכיות משוערכת של $O(\log^* n)$.

אחרי שקיצרנו את המסלול עם מציאת העמודה, נמצא את גובה התקליט על ידי סכימת הdelta של התקליט ושל ההורה שלו (שזה השורש אחרי שקיצרנו את הדרך) ב$O(1)$.

שתי הפעולות ביחד יוצאות $O(\log^* n)$ משוערך.