מבני נתונים - תרגיל רטוב 1

15 ביוני 2022

1 תיאור מבנה הנתונים:

1.1 עץ AVL − Rank

למען פתרון התרגיל מימשנו עץ דרגות AVL גנרי המחזיק מידע נוסף, המשתמש בפעולת השוואה בעזרת פונקציית customCompare(T, bool (T אשר מסופקת בבנאי - או, בברירת המחדל, אופרטור השוואה $>$ המוגדר עבור אותו אובייקט גנרי. בסיס העץ הינו העץ מתרגיל רטוב 1 עם שיפורים לתמיכה בדרגות וסכימה של ערך נוסף.

העץ מורכב ממצביע לשורש שהוא מטיפוס Node גנרי, בנוסף מכיל שדה המציין את גודל העץ.

$Node$: מחלקה המייצגת צומת בעץ, מכילה שדות - left_value, right_value, right_size, left_size, height ומצביעים לצומת ימנית ושמאלית.

העץ תומך בפעולות הסטנדרטיות: הוספה, חיפוש, מחיקה ואיחוד בין עצים. בנוסף לאלו, העץ תומך בפעולות getValueSumUpTo שמחזירה את סכום השדות $value$ של כלל הצמתים שקטנים מצומת ספציפית - בנוסף מעבירה את דרגת הצומת במשתנה עזר.

1.2 טבלת ערבול

למען פתרון התרגיל מימשנו טבלת ערבול כפי שנלמד בהרצאה, פונקציית הערבול היא על פי שיטת הכפל כפי שראינו בהרצאה. הטבלה מטפלת בהתנגשויות עם probing linear, ושומרת את מספר האיברים הנמצאים בתוכה. במידת הצורך, מתבצעת הגדלה של המבנה על ידי יצירת מערך חדש וערבול מחדש של האיברים לתוכה בעזרת הגודל החדש.

הטבלה עצמה מחזיקה מערך של $HashNode$ שהיא מחלקה גנרית המחזיקה שדה של Id, שדה של מידע ושדה שמסמן את אותו התא כמחוק. המבנה תומך בפעולות הסטנדרטיות - הכנסה, הסרה, חיפוש, ואיחוד.

1.3 Union-Find

למען פתרון התרגיל מימשנו UF עם איחוד קבוצות וכיווץ מסלולים והאיחוד מתבצע לפי גודל ולכן בסך הכל הסיבוכיות המשוערכת היא $O(\log^*(n))$ לחיפוש ואיחוד. המבנה מומש בעזרת חמישה מערכים בגודל מספר האיברים - שלושה מערכים כפי שראינו בהרצאה (האיבר, גודל והורה) ושני מערכים נוספים התומכים בפעולות הנדרשות בתרגיל - האחד, $value$ שומר ערך של קבוצה ומבצע חישובים בדומה לשיטת הארגזים שראינו בתרגול. השני, set_representative שומר בפועל מי הוא הנציג של אותה קבוצה - כדי שנוכל לבצע איחודים בסיבוכיות נכונה ועדיין לשמור על נכונות בחירת הנציגים.

•

1.4 המערכת

המערכת תנהל שני אובייקטים - עובדים וחברות. כל חברה תכיל (אולי) עובדים, וכל עובד יהיה שייך לחברה. כלל האובייקטים במערכת ינוהלו, יווצרו, וימחקו על ידי המערכת. היחסים בין עובדים לחברות ישמרו בתור התייחסויות עם מצביעים בלבד.

למען עמידה בסיבוכיות הדרושה, המערכת תחזיק את החברות בUnionFind כשערך כל חברה ינוהל גם בעזרתו - ואת מספרן במשתנה נוסף. בנוסף, כלל העובדים בערכת יוחזקו בHashTable כאשר התחום הוא תעודות הזהות של העובדים - בנוסף נחזיק עץ דרגות מאוזן שיכיל את כלל העובדים שאינם מתמחים (שכר גבוהה מ0) ממוין לפי שכר, ועבור כל שאר העובדים (שכר 0) נחזיק משתנים נוספים אשר מונים את סכום הדרגותיהם ומספרם, כל זאת על מנת לעמוד בדרישות התרגיל ובש' ההגשה.

1.4.1 חברה:

כל חברה תכיל שדה מזהה, את ערכה ההתחלתי וְ- HashTable של מצביעים לכלל העובדים של החברה - ובנוסף, בדומה למערכת, תכיל עץ דרגות של העובדים בחברה עם שכר וסכום דרגות שאר העובדים ומספרם בנפרד.

1.4.2 עובד:

העובד יכיל שדה של מזהה, שכר, דרגה ומצביע לחברה בה הוא עובד.

1.5 ממשק

1.5.1 סיבוכיות זמן:

למען פשטות והקלת הקריאה עבור הקורא המסור, נציין שֶ-

- פעולות החיפוש, סיבוכיות ההסרה, החיפוש וההכנסה בHashTable העובדים היא כפי שראינו בהרצאה, תחת הנחת הפיזור האחיד הפשוט של שלושת הפעולות היא משוערכת להיות $O(1)$ בממוצע על הקלט.
- פעולות החיפוש והאיחוד בUF החברות מתבצעות בסיבוכיות, כפי שראינו בהרצאה, של $O(\log^* n)$ - זאת בעזרת כיווץ מסלולים ואיחוד לפי גודל.
- הסרה והוספה בעץ העובדים עם שכר היא $\log n$ – כאשר n הוא מספר העובדים במערכת. אלגוריתם החיפוש עובד לפי שמורת החיפוש בעץ ולכן כפי שנלמד בהרצאה יקח $\log n$, אלגוריתם ההוספה עובד גם כפי הנלמד בהרצאה ולכן שומר על אותה סיבוכיות הזמן. במקרה ההסרה, במקרה הגרוע נבצע לאורך המסלול עד לשורש תיקונים לBF ולכן הסיבוכיות היא אותה הסיבוכיות זמן. בנוסף במהלך חלק מהפונקציות אנו מבצעים דפוס החוזר על עצמו : בדיקה האם האיבר בעץ, הכנסה / הוצאה , נציין שאם נבצע L פעמים חיפוש בעץ עדיין מתקיים שעבור J, L מספר טבעי : $J \log k + L \log n = O(\log k + \log n)$

1.5.2 סיבוכיות מקום:

במערכת כולה יוחזקו n עובדים וk חברות - בנוסף לאובייקטים עצמם השמורים בזכרון, יהיו העצים והטבלאות הערבול המוכלים במערכת, ובחברות, תלויים לינארית באותם משתנים. סכום כל הצמתים והתאים של עצי וטבלאות העובדים בתוך החברות הינו $2n$ מכיוון שזו חלוקה של העובדים (כל עובד יכול להיות בחברה אחת וישנו עץ וטבלה אחת בכל חברה), בנוסף - בתוך המערכת, הUF המחזיק את החברות מחזיק מספר קבוע של מערכים בגודל כמות החברות שבמערכת, העץ וטבלת הערבול של העובדים גם כן מחזיקים מספר קבוע של צמתים/תאים כתלות לינארים במספר העובדים במערכת (נדגיש כי טבלת הערבול מבצעת צמצום אם התפוסה קטנה מ25%, ולכן גודלה תמיד לינארי בתפוסתה).

1. **init**:
בפעולה מאתחלת את מבנה הנתונים, יוצרת k חברות ריקות ומכניסה כל אחת מהן לקבוצה משלה ב$UnionFind$ החברות. בנוסף, יוצרת עץ ריק וטבלת ערבול ריקה של עובדים סך הכל, $O(k)$.

2. **AddEmployee**:
נבדוק את ערכי הפרמטרים, במידה ולא עומדים בתנאים נחזיר את השגיאה המתאימה. נבצע חיפוש למציאת החברה שעלינו להוסיף את העובד בUF החברות (כפי שפירטנו בסעיף 1.5.1 הסיבוכיות היא $O(\log^* k)$) אם החברה אינה נמצאת – נחזיר את השגיאה המתאימה. ניצור את העובד, ונבצע חיפוש בטבלת הערבול לבדוק אם העובד כבר קיים במבנה ($O(1)$), אם הוא אינו קיים נבצע הכנסה אך ורק לטבלת הערבול (שכרו 0), ונבצע הכנסה לחברה (שגם מבצעת הכנסה רק לטבלת הערבול). נוסף 1 למניית הזוטרים ונוסיף את דרגתו לסכימת דרגותהים של הזוטרים. סך הכל הסיבוכיות הינה $O\left(\log^*(k)\right)$.

3. **RemoveEmployee**:
נבדוק את תקינות הפרמטרים, אם יש בעיה נחזיר את ערך ההחזרה המבוקש. נחפש את העובד לפי המזהה, ונשיג את החברה שבה הוא עובד ממנו ($O(1)$). נסיר את העובד מכלל המבנים שבהם הוא נמצא, סך הכל $O(\log n)$. אם העובד בעל שכר 0, נדע כי הוא זוטר ולא שווה הכנסה לעץ ולכן רק נסיר את דרגתו מסכום הדרגות של הזוטרים.

4. **AcquireCompany**:
נמצא את החברה הרוכשת והחברה הנרכשת בUF (1.5.1) ונוודא שאכן שתי החברות בקבוצות זרות. במידה והרכישה יכולה להתבצע נבצע מיזוג של החברות, כלומר מיזוג בין שני הקבוצות בUF בתוספת ה$factor$, שיבצע עדכון הערך של קבוצת החברה הרוכשת בUF. בנוסף, נבצע מיזוג בין החברות שיעביר את כלל העובדים מטבלת הנרכש לרוכש ויבצע מיזוג בין העצים. סך הכל $O(\log^* k + n_{\text{Acquirer}} + n_{\text{Target}})$ בשל שימוש בפונקציית merge וחיפוש החברות המתאימות. נדגיש כי המבנים שמכילים עובדים בעץ הנרכש מתרוקנים ולכן לא משפיעים על סיבוכיות המקום, על אף שהחברה נותרת במערכת.

5. **employeeSalaryIncrase**:
נמצא את העובד המבוקש בטבלה, נשמור את השכר הנוכחי שלו לפני העדכון. נשיג את החברה שהוא עובד בה ב$O(1)$ מהשדה השמור בעובד. נסיר את העובד מעץ המשכורות של המערכת ושל החברה, נעדכן את שכרו ונחזיר אותו לשני העצים. לאחר מכן, אם שכרו הקודם היה 0 - כלומר הוא היה זוטר, נוריד את הדרגה שלו מסכום הדרגות ונקטין את מספר הזוטרים ב1 כן בחברה וכן במערכת. סך הכל, $O\left(\log n\right)$ בממוצע על הקלט.

6. **PromoteEmployee**:
נמצא את העובד, נגדיר את דרגתו אם הפרמטר חיובי, אחרת נצא ונחזיר הצלחה. אם העלנו את דרגתו, נוציא ונחזיר אותו לעצים של השכר אם הוא בעל שכר גבוהה מ0. אחרת, נעלה את סכימת הדרגות של הזוטרים בפרמטר הנתון. סך הכל, בדומה ל$salaryIncrease$ הסיבוכיות היא $O\left(\log n\right)$.

7. **SumOfBumpGradeBetweenTopWorkersByGroup**:
לאחר בדיקת נכנות הקלט, נמצא את העץ הנכון לפי הפרמטר $companyID$ כמפורט בתרגיל. אם אין חברה מתאימה, נחזיר שגיאה. לאחר מכן, נשתמש בפונקציית העץ $getHighetMValueSum$ כדי לקבל את הערך הרצוי - סך הכל, זהו חיפוש בינארי בעץ דרגות מאוזן ולכן $O(\log^* k + \log n)$ בשל חיפוש החברה.

8. **AverageBumpGradeBetweenSalaryByGroup**:
לאחר בדיקת נכונות הקלו, נמצא את העץ הנכון לפי הפרמטר כמפורט בתרגיל. אם אין חברה מתאימה, נחזיר שגיאה. לאחר מכן, נשתמש בפונקציית העץ $findNode$ בכדי למצוא את ה$node$ הקרוב ביותר לתחום העליון - מימין, ולתחום התחתון - משמאל. זאת נעשה על ידי "חיפוש" עובד עם Id מינימאלי ומקסימלי בהתאם עם השכר הרצוי. לאחר שמצאנו את הצמתים הקרובים, נבצע תיקונים בהתאם להימצאותם בטווח. כעת, נשתמש בפונקציית $getValueSumUpto$ עבור שני הצמתים בקצוות הטווח ונחסיר את הקטן (בעצם - סכום הדרגות הקטנות מהטווח הקטן) מהגדול, ונקבל את הטווח הרצוי בדיוק. בדומה לכך, עם פרמטר שהעברנו לפונקציה נקבל גם את הכמות בכל תחום כזה. נחשב את הממוצע כפי שלמדנו בטכניון ונחזיר אותו. סך הכל, מתבצעים 4 חיפושים בעץ מאוזן בנוסף לחיפוש החברה המתאימה (אולי) לכן בסך הכל $O\left(\log^* k + \log n\right)$

9. **GetAllEmployeesBySalary**:
נבחר את עץ העובדים הממומין לפי השכר המתאים עפ"י ה companyID, במידת הצורך על פי חיפוש בעץ החברות. נקבל מערך inOrder של העץ לפי השכר בשימוש בפונקציה של העץ, ממנו ניצור מערך של תעודות זהות ונחזיר את המערך הזה. סך הכל סיבוכיות $O(\log k + n_{company})$ כשcompanyId > 0, אחרת זהו מערך כל העובדים ולכן $O(n)$.

10. **companyValue**:
נמצא את החברה המתאימה ב$UnionFind$ ונחזיר את הערך שלה שחישבנו בתוך המבנה כפי שראינו בתרגיל בשיטת הארגזים בתרגול. סך הכל $O(\log^* k)$.

11. **Quit**: תתבצע מחיקה של כלל העובדים במערכת לפי עץ העובדים הראשי, ומחיקה של כלל החברות במערכת לפי עץ החברות הראשי. כלל העצים מורכבים ממצביעים והינם תתי עצים של עצים אלו, לכן זו תהיה מחיקה של כל הזכרון שהוקצה דינאמית.