מבני נתונים - תרגיל רטוב 1

מבנה כללי:

מבנה מאגר התמונות שומר **עץ** AVL (מעתה ייקרא "$Images$") בעל מפתחות מספריים המסמלים את "מזהה התמונה" וערכים מסוג "תמונה".

כל ערך תמונה מכיל בתוכו מזהה תמונה, וכן **מערך ורשימה מקושרת דו כיוונית**.

המערך (מעתה ייקרא "$segments$") נועד לסמל את הסגמנטים של התמונה ואת התיוגים שלהם.

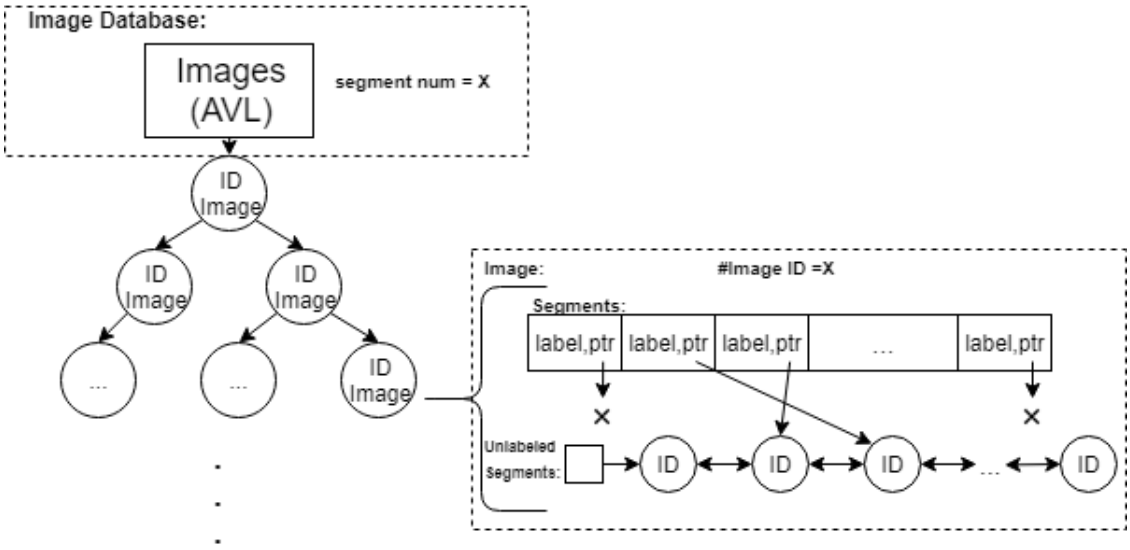
הרשימה המקושרת (מעתה תיקרא "$unlabeled$") נועדה להכיל מידע על הסגמנטים חסרי התיוג בתמונה.

המערך באורך זהה למספר הסגמנטים, כאשר כל תא במערך תואם באינדקס שלו למספר הסגמנט שהוא מייצג.

עבור כל תא במערך, הסגמנט שהוא מייצג יכול להיות משני סוגים:

- <u>עבור סגמנט מתויג:</u> התא המתאים במערך יכיל את הערך המספרי של התיוג ומצביע ל-$NULL$.
- <u>עבור סגמנט חסר תיוג:</u> התא המתאים במערך יכיל מצביע לחוליה מתאימה לסגמנט שתימצא ברשימה המקושרת, והערך המספרי בתא יהיה "ערך זבל" שאין להתייחס אליו כתיוג הסגמנט אם קיים בתא מצביע חוקי.

להלן תרשים הממחיש את המבנה הכללי:



מימוש הפעולות וסיבוכיות זמן:

$Init$:

מתבצע אתחול של העץ $Images$ כעץ ריק.
לאחר מכן שומרים בתוך המבנה את מספר הסגמנטים הקבוע לכל תמונה לשימוש עתידי.
מכיוון שאתחול עץ AVL והעתקת ערך מתבצעים ב־$O(1)$, זוהי גם הסיבוכיות הכוללת.

$AddImage$:

ראשית מתבצע חיפוש של התמונה בעץ. אם התמונה קיימת, מחזירים שגיאה.
אם התמונה שרוצים להוסיף היא אכן חדשה, מתבצעת יצירה של אובייקט התמונה שיש להכניס למבנה:
שומרים בתמונה את המזהה המספרי שלה, ולאחר מכן משום שביצירת תמונה, כל הסגמנטים בה חסרי תיוג, מכניסים בלולאה חוליה לרשימת $Unlabeled$ עבור כל סגמנט, המכילה את המזהה שלו.
במקביל, המצביע לכל חוליה נשמר בתא המתאים במערך $segments$.
מכיוון שהכנסת חוליה לרשימה מקושרת ושמירת במצביע שלה במערך המאפשר גישה ישירה מתבצעים ב־$O(1)$, נקבל שאתחול אובייקט תמונה מתבצע בסיבוכיות $O(n)$ כאשר עבור כל אחד מ־n הסגמנטים מתבצעות הפעולות הללו פעם אחת.
לאחר מכן מצביע לאובייקט התמונה החדש מוכנס לעץ $Images$ עם מזהה התמונה כמפתח.
סיבוכיות חיפוש והכנסה לעץ AVL כאשר במערכת k תמונות היא $O(\log(k))$.
לכן הסיבוכיות הכוללת היא: $O(\log(k)+n)$.

$DeleteImage$:

ראשית מתבצע חיפוש התמונה שיש למחוק בעץ $Images$ לפי המזהה שלה.
סיבוכיות החיפוש בעץ AVL בעל k תמונות: $O(\log(k))$.
אם התמונה נמצאה, מוציאים אותה מהעץ (בסיבוכיות $O(\log(k))$ בעץ AVL) ומוחקים את אובייקט ה־$Image$.
במחיקת האובייקט, מתבצע שחרור בלולאה של כל סגמנט במערך, ולאחר מכן שחרור כל חוליה ברשימה המקושרת.
מכיוון ששניהם מכילים לכל היותר n סגמנטים, ושחרור תא במערך או חוליה ברשימה מתבצע ב־$O(1)$, סיבוכיות ההריסה היא $O(n)$.
לכן הסיבוכיות הכוללת היא: $O(\log(k)+n)$.

$AddLabel$:

ראשית מתבצע חיפוש בעץ $Images$ של התמונה שלתוכה יש להוסיף תיוג.

חיפוש בעץ AVL בעל k תמונות הוא בסיבוכיות $O(\log(k))$.

אם התמונה נמצאה, מתבצעת גישה ישירה לתא במערך $segments$ שלה שמייצג את הסגמנט שיש לתייג.

אם מהבדיקה עולה שהסגמנט כבר מתויג (המצביע בתא $NULL$) מוחזרת שגיאה.

אם הסגמנט אינו מתויג, מתייגים את התא המתאים במערך ובאמצעות המצביע השמור בתא לחוליה המתאימה ברשימת $Unlabeled$ ניתן להוציא ולהרוס אותה בסיבוכיות $O(1)$ מכיוון שזו רשימה דו כיוונית.

לאחר מכן מסמנים את הסגמנט בתא המתאים כ"מתויג" על ידי השמה של $NULL$ למצביע החוליה בתוכו.

מכיוון שלאחר מציאת התמונה, כל הפעולות מתבצעות בגישה ישירה ובסיבוכיות $O(1)$, הסיבוכיות הכוללת היא: $O(\log(k))$ (ובפרט $O(\log(k)+n)$)

$GetLabel$:

ראשית מתבצע חיפוש בעץ $Images$ של התמונה שממנה רוצים לקרוא את התיוג.

חיפוש בעץ AVL בעל k תמונות הוא בסיבוכיות $O(\log(k))$.

אם התמונה לא נמצאה, מוחזרת שגיאה, ואחרת מבצעים גישה ישירה בתמונה למערך הסגמנטים ובודקים את מצב הסגמנט המבוקש.

אם הסגמנט המבוקש אינו מתויג, נדע זאת בסיבוכיות $O(1)$ מכך שהמצביע בתא המתאים במערך לא יהיה $NULL$, ונחזיר שגיאה.

אם הסגמנט המבוקש מתויג, נקרא את התיוג בגישה ישרה מהמערך ב־$O(1)$ ונחזיר אותו.

בסך הכול חיפוש ומספר קבוע של גישות ישירות הם בסיבוכיות כוללת של $O(\log(k))$.

$DeleteLabel$:

ראשית מתבצע חיפוש בעץ $Images$ של התמונה שבה מבקשים למחוק התיוג.

חיפוש בעץ AVL בעל k תמונות הוא בסיבוכיות $O(\log(k))$.

אם התמונה לא נמצאה מוחזרת שגיאה, ואחרת מבצעים גישה ישירה בתמונה למערך הסגמנטים ובודקים את מצב הסגמנט המבוקש.

בדומה לפעולה ב־$GetLabel$, אם הסגמנט אינו מתויג נוכל להחזיר שגיאה בסיבוכיות $O(1)$.

אם הסגמנט מתויג, כל שיש לעשות כדי לסמנו כ"לא מתויג" הוא ליצור חוליה חדשה ל־$Unlabeled$ עם מספר הסגמנט, להכניס אותה לרשימה בסיבוכיות $O(1)$ (כי זו הסיבוכיות שבה נתבקשנו לממש הכנסה לרשימה), ולאחר מכן להכניס את המצביע אליה לתא במערך $segments$ המסמל את הסגמנט שאת תיוגו מחקנו. מעתה בכל גישה למבנה הנתונים נוכל לראות שמצביע זה אינו $NULL$ ולכן לפי הצורה בה הגדרנו את המבנה, הסגמנט אכן אינו מתויג.

מספר קבוע של פעולות השמה והכנסת חוליה לרשימה מקושרת מתבצעים ב־$O(1)$.

לכן הסיבוכיות הכוללת היא $O(\log(k))$.

$GetAllUnlabeledSegments$:

ראשית מתבצע חיפוש בעץ $Images$ של התמונה שממנה יש למצוא סגמנטים.
חיפוש בעץ AVL בעל k תמונות הוא בסיבוכיות $O(\log(k))$.
אם התמונה אינה קיימת, נחזיר שגיאה ונבצע השמה של ערכי $NULL$ מתאימים ב־$O(1)$ למצביע שהועבר לפונקציה.
לאחר מכן נבצע ספירה של האזורים שאינם מתויגים בתמונה שנמצאה.
הספירה מתבצעת בסיבוכיות $O(1)$ משום שהרשימה המקושרת $Unlabeled$ ממומשת בחלק 1 באופן ששומר את שדה הגודל שלה ומחזיר אותו ב־$O(1)$.
לכן נוכל לדעת ב־$O(1)$ האם לא קיים בתמונה אזור פנוי (גודל $Unlabeled$ יהיה 0).
אם קיימים בתמונה אזורים פנויים, נקצה מקום למערך שיכיל את מספריהם (שאת אורכו השגנו לאחר הספירה).
נבצע סריקה של הרשימה $Unlabeled$ ונעתיק את תוכנה למערך (כלומר מקבלים מערך של כל מספרי הסגמנטים הלא מתויגים).
ההעתקה מתבצעת ב־$O(1)$ מכיוון שהערכים ב־$Unlabeled$ הם רק int.
לבסוף נחזיר את המערך החדש שבנינו שמכיל את כל הסגמנטים הלא מתויגים, דרך המצביע שהועבר לפונקציה.
לפי הגדרת הרשימה $Unlabeled$ כרשימה שמכילה רק את מספרי הסגמנטים הלא מתויגים: נקבל שאורכה s, ומכיוון שעבור כל חוליה שלה מתבצעות פעולות ב־$O(1)$ נקבל שסיבוכיות המעבר על הרשימה היא $O(s)$.
לכן הסיבוכיות הכוללת היא $O(\log(k)+s)$.

$GetAllSegmentsByLabel$:

ראשית נבצע סיור על העץ $Images$ על מנת לחלץ ממנו מערך המכיל מצביעים לכל התמונות.
סיור מתבצע על עץ חיפוש בעל k תמונות ב־$O(k)$.
לאחר מכן נעבור על מערך התמונות בלולאה ועבור כל תמונה נסכום את מספר הסגמנטים שהם בעלי התיוג הרצוי.
עבור כל תמונה, הספירה מתבצעת על ידי מעבר בלולאה על מערך $segments$ שלה והחזקת מונה של מספר הסגמנטים שנמצאו מתויגים ובעלי התיוג הרצוי.
לכן עבור כל תמונה נוכל לספור את הסגמנטים הרלוונטיים בה ב־$O(n)$.
לכן הסכימה על כל k התמונות מתבצעת ב־$O(k \cdot n)$.
לאחר מכן נקצה שני מערכי מספרים בגודל שחילצנו מהסכימה (הקצאה ב־$O(1)$).
נבצע שוב איטרציה על מערך התמונות. עבור כל תמונה, נחלץ מתוכה מערך של כל הסגמנטים שהם בעלי התיוג הרצוי.
הדבר מתבצע על ידי איטרציה על מערך $segments$ והעתקת מספרי הסגמנטים הרצויים למערך נפרד.
לאחר בניית מערך הסגמנטים בעלי התיוג הרצוי, נבצע איטרציה עליו, ועבור כל סגמנט עם התיוג הרצוי, נעתיק את מספרו ואת מספר התמונה שבה הוא נמצא למערכים שבנינו.
על מנת לחלץ את מערך הסגמנטים הרצויים בכל תמונה, הסיבוכיות היא סיבוכיות איטרציה על מערך $segments$ וביצוע העתקות ב־$O(1)$ עבור כל תא בו, כלומר סיבוכיות $O(n)$.
לאחר מכן סיבוכיות איטרציה נוספת על מערך הסגמנטים הרצויים גם היא ב־$O(n)$ משום שמערך זה מכיל לכל היותר את כל הסגמנטים בתמונה.

לכן עבור כל תמונה מצבעים פעולות בסיבוכיות $O(n)$, כלומר הסיבוכיות הכוללת עבור כל התמונות היא $O(k \cdot n)$ מכיוון שעבור כל סגמנט מבצעים מספר קבוע של פעולות ובדיקות ב־$O(1)$.

מכיוון שבעץ $Images$, התמונות שמורות ממוינות לפי המזהים שלהם מתכונות עץ חיפוש, וסיור $Inorder$ שומר על תכונה זו כאשר אנו מעתיקים את התמונות למערך, וכמו כן בתוך כל תמונה אנו מבצעים איטרציה על מערך ה־$segments$ מההתחלה לסוף, נקבל שמערכי התוצאה ממוינים כמבוקש לפי מזהה תמונה בסדר עולה, ועם סידור פנימי בסדר עולה של סגמנטים בתמונה משותפת.

לבסוף נהרוס את מערך מצביעי התמונות שחילצנו מהעץ איבר־איבר. מכיוון שהמערך מכיל מצביעים לתמונות, ולא את התמונות עצמן, נוכל להרוס כל מצביע (תא במערך) בסיבוכיות $O(1)$. כלומר המערך כולו נהרס בסיבוכיות $O(k)$: אורך המערך.

בסך הכול ביצענו רצף פעולות בסיבוכיות $O(k + k \cdot n + k \cdot n + k) = O(k \cdot n)$.

$Quit$:

מתבצע מעבר על העץ $Images$ ושחרור כל תמונה בתוכו.

במימוש מחלק 1 של העץ, סיבוכיות השחרור של מבנה העץ בעל k התמונות היא $O(k)$.

אך בנוסף עבור כל תמונה בעץ קיימת סיבוכיות של שחרור השדות בתמונה:

עבור כל תמונה בעץ נעבור על מערך הסגמנטים בלולאה ונשחרר את הזיכרון המוקצה לכל תא ב־$O(1)$.

כלומר סיבוכיות ההריסה של המערך באורך n היא $O(n)$.

לאחר מכן נעבור חוליה־חוליה על הרשימה המקושרת $Unlabeled$ ונהרוס כל חוליה בסיבוכיות $O(1)$.

הרשימה מכילה לכל היותר את כל n הסגמנטים בתמונה, לכן הסיבוכיות של הריסתה היא $O(n)$.

בסך הכול סיבוכיות הריסה של כל תמונה היא $O(n)$, ולכן סיבוכיות הריסת כל k התמונות היא $O(k \cdot n)$, וכזו גם הסיבוכיות הכוללת.

סיבוכיות מקום:

במבנה בעל k תמונות קיים עץ AVL ששומר אותן, ולכן סיבוכיות המקום שלו היא $O(k)$.

בנוסף, סיבוכיות המקום בכל תמונה היא סיבוכיות שמירה של מערך באורך n הסגמנטים, רשימה שמכילה לכל היותר את כל הסגמנטים ולכן באורך לכל היותר n, וכן מספר קבוע של ערכים ומצביעים.

לכן סיבוכיות המקום של אחסון תמונה היא $O(n)$, ולכן סיבוכיות אחסון כל k התמונות, שהיא הסיבוכיות הכוללת של המבנה, היא $O(k \cdot n)$.

כעת נותר להוכיח שסיבוכיות המקום של רקורסיות והקצאות מערכים (פרט למערכים השמורים במבנה) בפעולות אינן חורגות מסיבוכיות זו:

$Init$:

אין הקצאת מערכים או מקום נוסף, וללא רקורסיות.

$Quit, AddImage, DeleteImage, AddLabel, GetLabel, DeleteLabel$:

בכל הפונקציות הללו אין הקצאת מקום נוסף.

הרקורסיה שמתבצעת בהן היא רקורסיה בפעולות הריסה, חיפוש, הכנסה והוצאה לעץ AVL.

מכיוון שהעץ בעומק k, מתכונות עץ AVL הפעולות הללו הן בעומק לכל היותר גובה העץ, כלומר $\log(k)$.

לכן סיבוכיות המקום הנוסף ברקורסיה בכל הפעולות הללו היא $O(\log(k))$ ובפרט $O(k \cdot n)$.

$GetAllUnlabeledSegments$:

כמו בפעולות הקודמות, ברקורסיה מבצעים פעולות על עץ AVL בסיבוכיות מקום $O(\log(k))$.

בנוסף מתבצעת הקצאה של מערך הסגמנטים הלא מתויגים, שהוא לכל היותר באורך $k \cdot n$ (אם כל הסגמנטים בכל התמונות אינם מתויגים).

לכן סיבוכיות המקום הנוסף היא $O(k \cdot n)$.

$GetAllSegmentsByLabel$:

באופן דומה לפעולה הקודמת, קיימת סיבוכיות מקום רקורסיבית של $O(\log(k))$.

בנוסף מקצים מערך זמני לשמירה של כל התמונות, באורך k ולכן בסיבוכיות מקום נוסף $O(k)$.

מערך זה נהרס בסוף הפעולה, ולכן הסיבוכיות אינה מצטברת במבנה.

בנוסף מתבצעת הקצאה של המערכים המוחזרים מהפונקציה, שהם לכל היותר מערך של כל k התמונות ומערך של כל $k \cdot n$ הסגמנטים.

כלומר סיבוכיות המקום הנוסף היא עדיין $O(k \cdot n)$.

בסך הכול הראינו שסיבוכיות המקום במבנה וסיבוכיות המקום הנוסף בפונקציות היא $O(k \cdot n)$, ולכן זו סיבוכיות המקום הכוללת כנדרש.