**מבני נתונים**

**234218**

**תרגיל רטוב 2**

**מגישים: רוני אנגלנדר - 312168354,**

**נדב אורזך - 311549455**

**תאריך הגשה: 12.1.2019**

**תיאור מבני הנתונים**

מימשנו מבנה נתונים בשם ImageTagger המכיל מאגר של תמונות מתויגות ותומך בכל הפעולות הנדרשות:

**ImageTagger**

מבנה הנתונים ImageTagger מכיל בתוכו את השדות הבאים:

* Images[] – HashTable מסוג רשימה שמומשה בתרגיל הרטוב 1, שמכילה מצביעים לטיפוס מסוג Image (נפרט בהמשך), שנממש על פי אלגוריתם ה - chain hashing. למעשה images הוא מערך דינאמי של רשימות, כאשר הגדרנו את הגודל ההתחלתי שלו להיות 8 (#define **START\_SIZE** 8). כמו כן בהתאם לנלמד בכיתה נשתמש בפונקצית ערבול של שמפזרת באופן אחיד במערך בגודל size.
* size – משתנה מסוג int שמייצג את הגודל הנוכחי של המערך הדינאמי images.
* num\_images – משתנה מסוג int שמייצג את מספר התמונות הנמצאות באותו רגע במבנה.
* num\_pixel – משתנה מסוג int שיאותחל למספר קבוע באתחול המבנה שמייצג את מספר הפיקסלים בכל תמונה.

מבנה זה מקיים לפי הנלמד בהרצאה תחת הנחת הפיזור האחיד הפשוט סיבוכיות זמן משוערכת של הכנסה של תמונות חדשות והוצאה של תמונות ממבנה של O(1).

נסביר בקצרה על מספר מתודות של המבנה **ImageTagger**:

* void check\_and\_expand() – פונקציה שתיקרא לפני הוספה של כל Image חדש. הפונקציה בודקת האם יש צורך להרחיב את המערך בהתאם לפקטור עומס שהגדרנו כ-0.9, בהתאם לאלגוריתם הנלמד בכיתה. תחת הנחת הפיזור האחיד, אורכה הממוצע של שרשרת הוא פקטור העומס ולכן זמן חיפוש הוא לכל היותר Θ(1+load factor) ולכן אנו שומרים על סיבוכיות זמן המשוערכת הנדרשת. אם כמות התמונות במערך ביחס לגודלו תהיה גדולה או שווה ל-0.9 ניצור ונאתחל מערך חדש שגודלו כפול מהמערך הקודם, נכניס את כל התמונות מהמערך הקודם למערך החדש לפי פונקציות הערבול ביחס למערך החדש ונמחק את המערך הישן. נעדכן את images להיות המערך החדש ואת size להיות גודל המערך החדש.

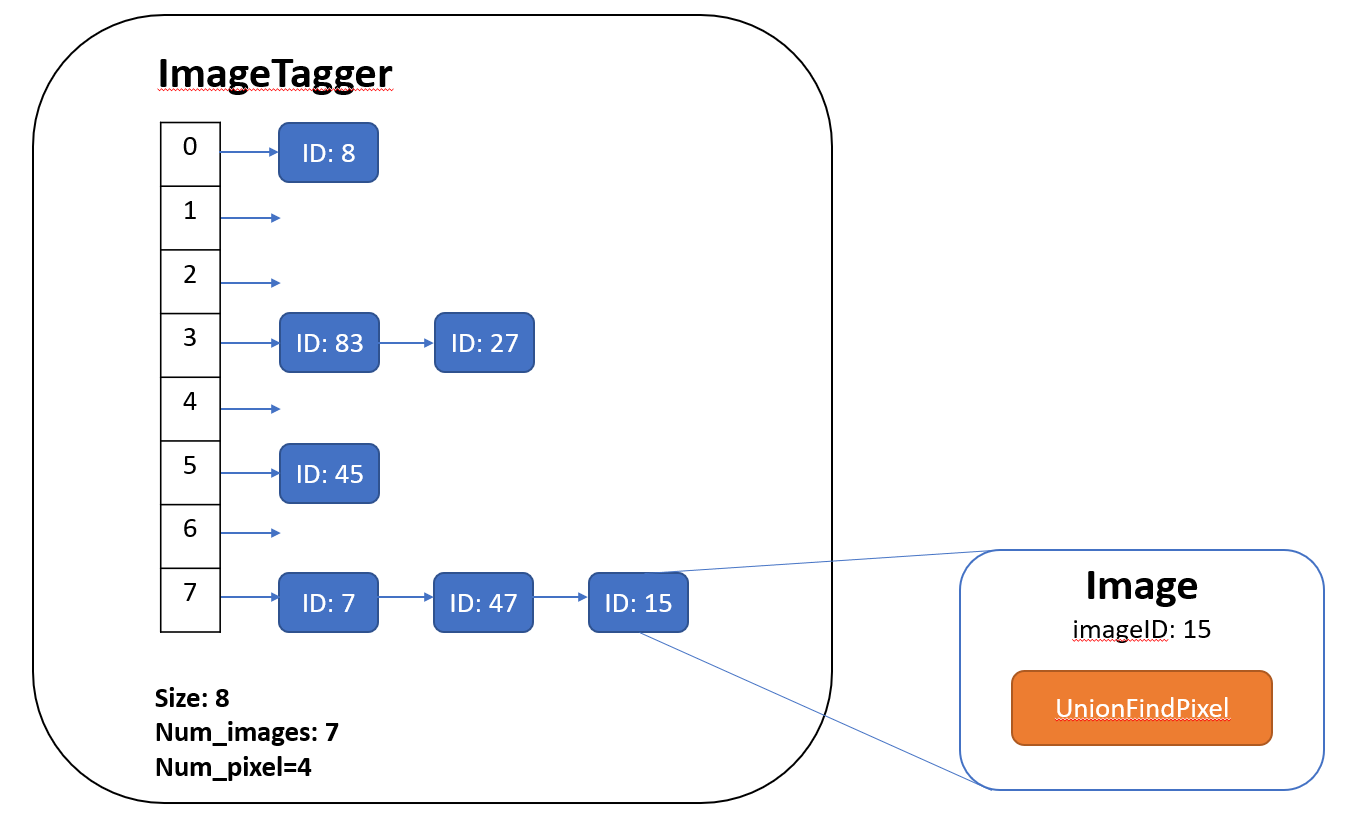
פעולת הסריקה של המערך להוצאת האיברים נעשה בסיבוכיות זמן של O(n) כאשר n הוא גודל המערך, הוצאה כל איבר מהרשימה מתבצעת על ידי הוצאת האיבר הראשון מהרשימה שבתא הנוכחי לכן מתקיימת בסיבוכיות זמן של O(1), והכנסה למערך החדש, שוב למקום הראשון ברשימה החדשה בתא המתאים לפי פונקצית הערבול, גם כן בסיבוכיות של O(1). לסיכום סיבוכיות זמן של פעולה זו היא O(n) במקרה הגרוע. באותו אופן, ניתן לראות כי סיבוכיות המקום היא לינארית במספר התמונות במערך n (כיוון שהמערך דינאמי וגדל/קטן בהתאם לn) כאשר כל תמונה מקיימת סיבוכיות מקום של מספר הפיקסלים בה (k) ומספר התיוגים השונים הקיימים סך הכל (m). לכן סיבוכיות המקום מקיימת O(n\*k\*m).

* void check\_and\_shrink() – פוקנציה שתיקרא לאחר כל מחיקת Image. הפונקציה בודקת האם יש צורך להקטין את המערך בהתאם לפקטור עומס שהגדרנו כ-0.25, בהתאם לאלגוריתם הנלמד בכיתה. המשך הפונקציה תתבצע בדומה לפונקציה check\_and\_expand ובאותו אופן סיבוכיות הזמן במקרה הגרוע היא O(n) וסיבוכיות המקום היא O(n\*k\*m).

**Image**

מבנה הנתונים Image מכיל בתוכו את השדות הבאים:

* imageID – משתנה מסוג int שמייצג את מזהה התמונה. מזהה זה ישמש גם כן כמפתח במילון מסוג רשימה שתואר במבני הקודם (תאי המערך images).
* pixels – אובייקט מסוג מצביע ל-UnionFindPixel, מבני שמקיים את עקרונות ה- Union Find כפי שנלמד בשיעור (נפרט על מבנה זה בהמשך).



**UnionFindPixel**

את מבנה הנתונים UnionFindPixelמימשנו באופן המוצע בהרצאה למימוש Union-Find בעזרת מערכים. כמו כן מימשנו מבנה זה בעזרת איחוד לפי גודל וכיווץ מסלולים. מבנה זה מכיל בתוכו את השדות הבאים:

* num\_of\_pixels - משתנה מסוג int שיאותחל למספר קבוע באתחול המבנה שמייצג את מספר הפיקסלים הקיימים במבנה.
* 3 מערכים בגודל num\_of\_pixels, כאשר עבור כל אינדקס בתחום, כל התאים במערכים אלו המתאימים לאותו אינדקס מייצגים פיקסל יחיד, השייך ל-SuperPixel אחד, כאשר כל ה-SuperPixels זרים אחד לשני. באתחול המבנה כל פיקסל נמצא ב-SuperPixel נפרד וללא תיוגים.
  + size – משתנה מסוג מערך של int. אם הפיקסל במקום ה-i הוא שורש של ה-SuprtPixel שלו אז בתא ה-i במערך הזה יופיע גודל הקבוצה אותה הוא מייצג. אחרת יופיע בתא זה -1 כי פיקסל זה שייך לקבוצה אך אינו שורשה.
  + parent – משתנה מסוג מערך של int. אם הפיקסל במקום ה-i הוא שורש של ה-SuprtPixel שלו אז בתא ה-i במערך הזה יופיע -1 מכיוון שלפיקסל זה אין אב. אחרת יופיע בתא זה מספר האינדקס שמייצג את האב של אותו פיקסל.
  + labels – משתנה מסוג מערך של Map\_tree (מילון מסוג עץ AVL כפי שמומש בתרגיל רטוב 1). למבנה ה-Map\_tree מהתרגיל הרטוב הקודם, הוספנו שדה לכל איבר בעץ שמכיל את הניקוד המקסימלי של תת העץ שאותו איבר הוא השורש שלו, ואת מספר הלייבל שמכיל את ניקוד זה. בכל הוספה, מחיקה או עדכון של איברים בעץ, אנחנו חוזרים רקורסיבית לשורש העץ ומעדכנים את הניקוד המקסימלי ומספר הלייבל המקסימלי של כל איבר עד לשורש. בצורה כזו, שורש העץ מכיל את הלייבל שמכיל את הניקוד המקסימלי בעץ.

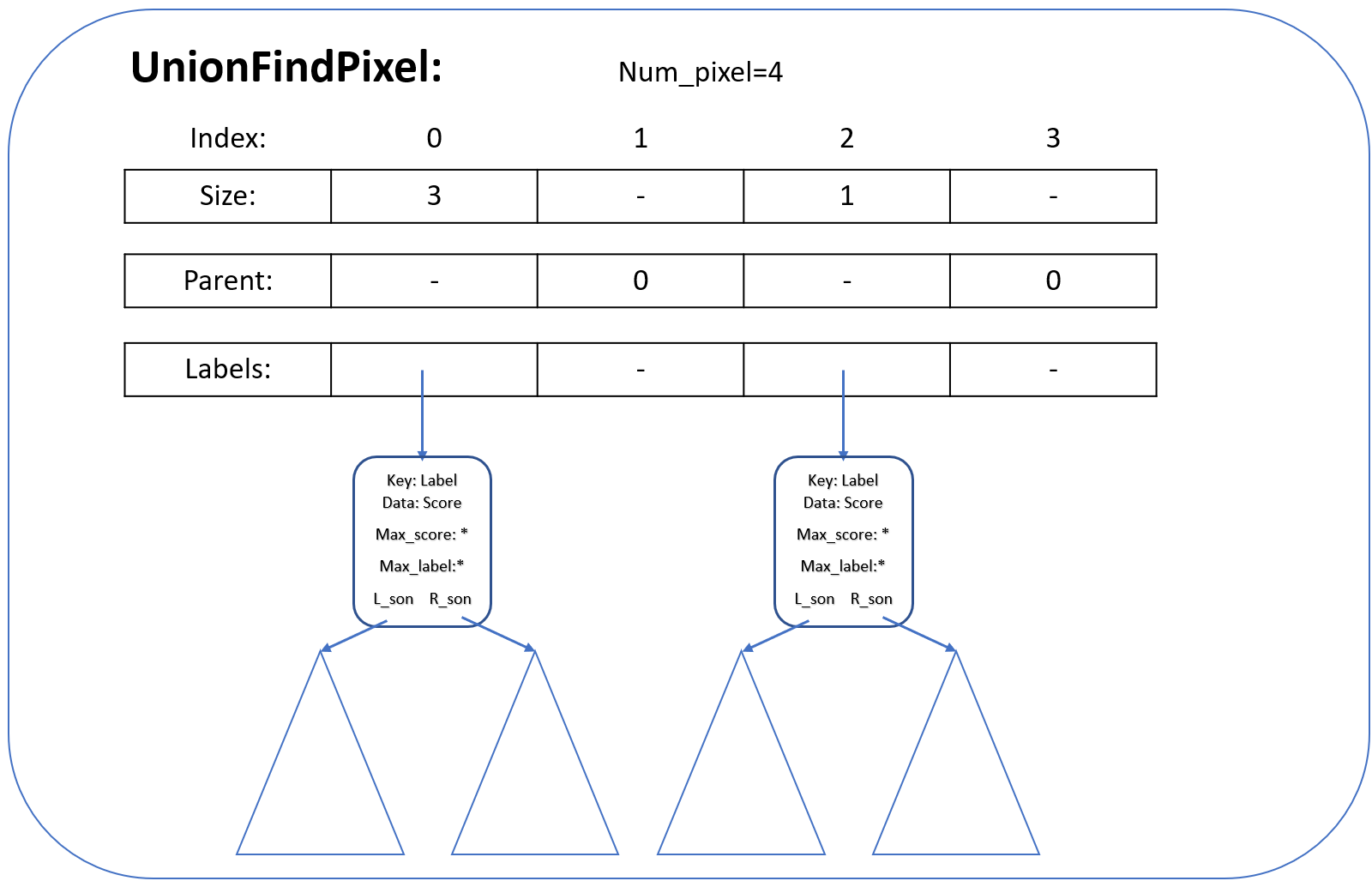
אם הפיקסל במקום ה-i הוא שורש של ה-SuprtPixel שלו אז בתא ה-i במערך הזה יופיע עץ שיכיל את כל הלייבלים ששייכים לאותו SuperPixel, כאשר כל איבר בעץ זה מיוצג על ידי מפתח שהוא מספר הלייבל, והמידע של אותו איבר יהיה ה-score של אותו לייבל. אחרת יופיע בתא זה עץ ריק.

מבני זה מקיים לפי הנלמד בהרצאה ובהתאם למימוש שלנו:

* איתחול המבנה מאחל 3 מערכים בגודל קבוע, כאשר עבור 2 המערכים הראשונים מבוצעות k פעולות (כאשר k הוא מספר הפיקסלים בתמונה), ועבור המערך השלישי מבוצעות k פעולות אתחול של מילון מסוג עץ, שלפי תרגיל רטוב 1 מבוצע במקרה הגרוע ב-O(1). לכן, איתחול בסיבוכיות זמן של O(k) במקרה הגרוע.
* חיפוש בסיבוכיות זמן של O(logn) במקרה הגרוע.
* איחוד בסיבוכיות זמן של O(1) במקרה הגרוע.
* כאשר הסיבוכיות המשוערכת של חיפוש ואיחוד היא O(log\*n)

נסביר בקצרה על מספר מתודות של המבנה **UnionFindPixel**:

* merge\_trees(tree1,tree2) – פונקציה שממזגת שני עצי AVL שונים בהתאם לאלגוריתם הנלמד בהרצאה- ראשית נכניס כל עץ למערך ממויין על ידי סריקה שלו inoreder – מבוצע בסיבוכיות זמן של O(n) כאשר n הוא מספר האיברים (לייבלים) בשני העצים יחד. נמזג את שני המערכים למערך אחד ממוין על ידי סריקה של שני המערכים – סיבוכיות זמן O(n). ניצור עץ שלם באופן רקורסיבי שכל איבריו ריקים שגובהו log(n) – סיבוכיות זמן O(n). נחשב את מספר העלים שעלינו למחוק מצד ימין שכמות האיברים בעץ תהיה זהה לגודל המערך ונקבל עץ כמעט שלם. כעת נסרוק את העץ החדש inorder באותו אופן ונכניס את תאי המערך הממוין שיצרנו לעץ ונקבל עץ AVL תקין חדש. סיבוכיות הזמן של פונקציה זו היא במקרה הגרוע O(n). סיבוכיות מקום נוסף: בפונקציה אנו יוצרים 2 מערכים המייצגים את הלייבלים וממזגים אותם למערך אחוד של סה"כ הלייבלים. סיבוכיות המקום של המערכים הינה O(n). לאחר מכן אנו יוצרים עץ כמעט שלם בגודל מספר הלייבלים ומכניסים אליו את איברי המערך האחוד, לכן סיבוכיות המקום של העץ הינה O(n). כלומר בסה"כ סיבוכיות המקום הנוסף במיזוג שני העצים היא O(n).



**מימוש הפעולות הנדרשות**

**void\* Init(int pixels)**:

הפונקציה מאתחלת ImageTagger חדש. באתחול המבנה ניצור מערך דינאמי מגודל קבוע כפי שצוין (START\_SIZE 8). כמו כן נאתחל את שאר השדות במבנה זה, גם הם בגודל קבוע. לכן:

**סיבוכיות זמן במקרה הגרוע: O(1) | סיבוכיות מקום נוסף: O(1)**

**StatusType AddImage(void \*DS, int imageID):**

הפונקציה מוסיפה תמונה חדשה למבנה, כאשר כל פיקסל נמצא ב-SuperPixel נפרד והוא ללא תיוגים. ניצור תמונה חדשה לפי הקלט המתקבל, ונאתחל את השדה מסוג UnionFindPixel בסיבוכיות זמן של O(k) כפי שתיארנו בפירוט המבנה. לאחר מכן, נבדוק האם יש צורך להרחיב את הטבלת הערבול בהתאם לפקטור עומס שהגדרנו, בעזרת check\_and\_expand. נכניס את התמונה לטבלת הערבול, לפי המתואר לעיל פעולה זו מתבצעת בסיבוכיות משוערכת של O(1).

כמו כן, ניתן לראות כי יצרנו תמונה חדשה שמכילה k פיקסלים ובמקרה הגרוע בו הרחבנו את המערך הדינאמי שלנו יצרנו עוד n תאים למערך (כאשר n הוא מספר התמונות במערך) לכן קיבלנו:

**סיבוכיות זמן משוערכת: O(k) | סיבוכיות מקום נוסף: O(n + k)**

**StatusType DeleteImage(void \*DS, int imageID):**

הפונקציה מוחקת תמונה מהמבנה לפי המזהה תמונה שמתקבל, ומוחקת את כל השדות שלה. נמצא את התמונה ונסיר אותה. לאחר מכן, נבדוק האם יש צורך להקטין את המערך בהתאם לפקטור עומס שהגדרנו על ידי שימוש בפונקציה check\_and\_shrink. לפי האמור לעיל פעולות אלו מבוצעות בסיבוכיות משוערכת של O(1). נמחק את 2 שדות המערכים הראשונים של התמונה שבגודל k – מספר הפיקסלים בתמונה. עבור המערך השלישי, מערך העצים, מחיקת כל עץ נעשית בסיבוכיות זמן במקרה הגרוע של O(d) כאשר d הוא מספר האיברים בעץ. לכן למחיקת המערך תידרש סיבוכיות זמן במקרה הגרוע של כאשר m הוא מספר הלייבלים בתמונה סך הכל. בנוסף, בפעולה זו לא בוצעו הקצאות זיכרון חדשות. לכן:

**סיבוכיות זמן משוערכת: O(k + m) | סיבוכיות מקום נוסף: O(1)**

**StatusType SetLabelScore(void \*DS, int imageID, int pixel, int label, int score):**

הפונקציה מתייגת את הסופר-פיקסל שמכיל את הפיקסל בעל המזהה pixel בתמונה imageID בתיוג label עם הניקוד score . נמצא את התמונה המתאימה למזהה שהתקבל במערך הדינאמי שלנו, לפי הנלמד בהרצאה תחת הנחת הפיזור האחיד, זמן חיפוש של תמונה במערך בממוצע הוא O(1). לאחר שמצאנו את התמונה ניגש לשדה ה-UnionFindPixel שלה ונחפש את ה-SuperPixel המתאים, לפי האמור לעיל מתבצע בסיבוכיות משוערכת של O(log\*k) כאשר k הוא מספר הפיקסלים בתמונה. לאחר שמצאנו את שורש ה-SuperPixel המתאים ניגש לבדוק את עץ הלייבלים שלו, נחפש את הלייבל המתאים, ונעדכן את ה-score שלו או נוסיף לייבל חדש לפי הדרישות. פעולות אלו על העץ מבוצעות בסיבוכיות זמן במקרה הגרוע של O(log(m)) כאשר m הוא מספר הלייבלים בעץ של ה-SuperPixel, לפי ההרצאה. לא הוקצה זכרון נוסף בפונקציה זו. לכן:

**סיבוכיות זמן משוערכת: O(log\* k + log m) | סיבוכיות מקום נוסף: O(1)**

**StatusType ResetLabelScore(void \*DS, int imageID, int pixel, int label):**

הפונקציה מוחקת את התיוג label בסופר פיקסל המכיל את הפיקסל עם המייצג pixel בתמונה המתאימה למזהה imageID. ראשית, נמצא את התמונה המתאימה למזהה שהתקבל במערך הדינאמי שלנו, לפי הנלמד בהרצאה תחת הנחת הפיזור האחיד, זמן חיפוש של תמונה במערך בממוצע הוא O(1). לאחר שמצאנו את התמונה ניגש לשדה ה-UnionFindPixel שלה ונחפש את ה-SuperPixel המתאים, לפי האמור לעיל מתבצע בסיבוכיות משוערכת של O(log\*k) כאשר k הוא מספר הפיקסלים בתמונה. לאחר שמצאנו את שורש ה-SuperPixel המתאים נמחק את התיוג המתאים למזהה label, לפי סיבוכיות העץ שנבנה בתרגיל הרטוב הקודם, פעולה זו מתבצעת בסיבוכיות זמן במקרה הגרוע של O(log(m)) כאשר m הוא מספר הלייבלים בעץ של ה-SuperPixel. לא הוקצה זכרון נוסף בפונקציה זו. לכן:

**סיבוכיות זמן משוערכת: O(log\* k + log m) | סיבוכיות מקום נוסף: O(1)**

**StatusType GetHighestScoredLabel(void \*DS, int imageID, int pixel, int \*label):**

הפונקציה תחזיר את המזהה של התיוג בעל ה-score המקסימלי בסופר פיקסל המכיל את pixel בתמונה עם המזהה imageID. ראשית, נמצא את התמונה המתאימה למזהה שהתקבל במערך הדינאמי שלנו, תחת הנחת הפיזור האחיד, זמן חיפוש של תמונה במערך בממוצע הוא O(1). לאחר שמצאנו את התמונה ניגש לשדה ה-UnionFindPixel שלה ונחפש את ה-SuperPixel המתאים, לפי האמור לעיל מתבצע בסיבוכיות משוערכת של O(log\*k) כאשר k הוא מספר הפיקסלים בתמונה. כעת, ניגש למערך המכיל את עצי הלייבלים של המבנה במקום המתאים לסופר פיקסל, ולפי מבנה העץ שתיארנו שורש העץ מכיל את המזהה המתאים ללייבל עם הניקוד המקסימלי. נעתיק אותו ל- \*lable בסיבוכיות זמן במקרה הגרוע של O(1). לא הוקצה זכרון נוסף. לכן:

**סיבוכיות זמן משוערכת: O(log\* k) | סיבוכיות מקום נוסף: O(1)**

**StatusType MergeSuperPixels(void \*DS, int imageID, int pixel1, int pixel2):**

הפונקציה תאחד את 2 הסופר פיקסלים המכילים את pixel1 ו-pixel2 בתמונה עם המזהה imageID. ראשית, נמצא את התמונה המתאימה למזהה שהתקבל במערך הדינאמי שלנו, תחת הנחת הפיזור האחיד, זמן חיפוש של תמונה במערך בממוצע הוא O(1). לאחר שמצאנו את התמונה ניגש לשדה ה-UnionFindPixel שלה ונחפש את ה-SuperPixel המתאים ל-pixel1 ול-pixel2, לפי האמור לעיל מתבצע בסיבוכיות משוערכת של O(log\*k) כאשר k הוא מספר הפיקסלים בתמונה. כעת נבצע איחוד של עצי הלייבלים של שני הSuperPixel על ידי שימוש בפונקציה merge\_trees המתוארת לעיל, שפועלת לפי ההסבר בסיבוכיות זמן של O(m) במקרה הגרוע כאשר m הוא מספר התיוגים בשני הסופר פיקסלים ביחד. כמו כן פונקציה זו מקיימת סיבוכיות מקום נוסף של O(m) לכן:

**סיבוכיות זמן משוערכת: O(log\* k + m) | סיבוכיות מקום נוסף: O(m)**

**void Quit(void \*\*DS):**

הפעולה משחררת את המבנה. הפונקציה סורקת את כל המערך הדינאמי מוציאה את כל התמונות (n),עבור כל תמונה מוחקת את כל הפיקסלים שלה (k) ומוחקת את כל התיוגים שלה (m מספר התיוגים בכל המערכת). לא הוקצה זכרון נוסף. לכן:

**סיבוכיות זמן משוערכת: O(n\* k \* m) | סיבוכיות מקום נוסף: O(1)**