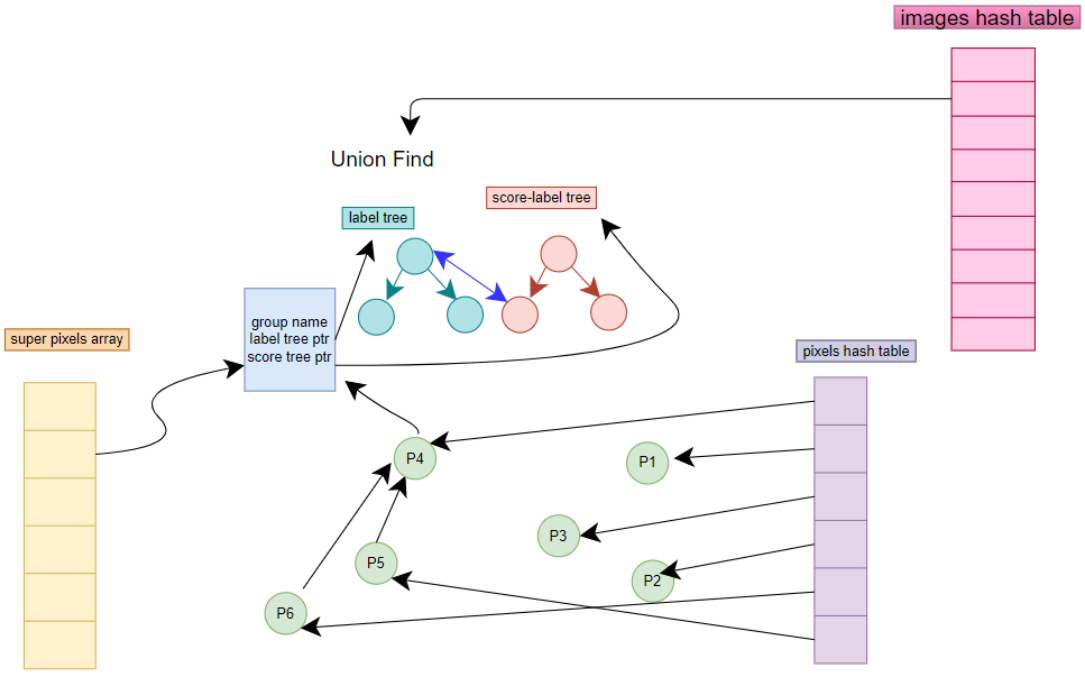
תרגיל בית רטוב 2

חלק יבש

תיאור מבנה הנתונים:

ניצור hash table בה נשמור את כל הimageID, שכל imageID יצביע למבנה של UF, במבנה נוסיף בשדה של המידע על הקבוצה עוד 2 שדות של מצביעים לשני עצים זהים, האחד ממוין לפי התיוגים והשני לפי ניקוד ואז תיוג, וכל צומת זהה בין שני העצים מקושרת באמצעות פוינטר דו-כיווני.



Init(int pixels):

ניצור Table hash חדשה של מצביעים לUF .
במידה והייתה בעיה בהקצאת הזיכרון נחזיר מצביע ריק.
אחרת, נחזיר מצביע אל ההאש.
סיבוכיות זמן : O(1)

AddImage(void* DS, int imagesID):

1. אם DS הוא NULL או imageID הוא קטן או שווה לאפס, נחזיר INVALID_INPUT.
2. נבדוק האם ה ImageID קיים בהאש (סיבוכיות משוערכת O(1))
3. אם הוא קיים, נחזיר FAILURE
4. ניצרו מבנה UF חדש, ונבצע MakeSet עבור כל אחד מהפיקסלים בתמונה (סיבוכיות זמן של O(k) משוערך)

5. אם הייתה בעיה בהקצאת הזיכרון נחזיר ALLOCATION ERROR
6. אחרת, נחזיר SUCCESS

סיבוכיות זמן: O(k)משוערך

DeleteImage(void* DS, int imageID):

1. אם DS הוא NULL או imageID הוא קטן או שווה לאפס, נחזיר INVALID_INPUT.
2. נבדוק האם ה ImageID קיים בהאש (סיבוכיות משוערכת O(1))
3. אם הוא לא קיים, נחזיר FAILURE
4. ניגש לUF אליו מצביע ה imageID בהאש
5. נעבור על המערך של הקבוצות – הסופר פיקסלים, עבור כל סופר פיקסל נבצע את הבאים:
   - ראשית נבדוק האם עצי התיוגים שלו ריקים (אם אחד ריק השני ריק)
   - אם לא, נמחק את כל התיוגים של הסופר פיקסל ע"י כך שנמחק את שני העצים בהם אנו שומרים את התיוגים (סיבוכיות זמן היא O של כמות הצמתות בעץ)
   - נעבור על כל הצמתות - הפיקסלים של הסופר פיקסל, נסיר אותן מההאש של הפיקסלים ולאחר מכן נמחק אותן. נעדכן את הגודל של הקבוצה להיות 0.
   - נמחק את המידע של הקבוצה ונשחרר את המצביע אליו במערך.
6. לאחר שסיימנו נשחרר את המצביע מההאש לUF הזה (את מה שנשאר למחוק מהUF נבצע אוטומטית ע"י דיסטרקטורים)
7. ונסיר מההאש של התמונות את התמונה imageID
8. במידע ותהיה בעיה בהקצאת זיכרון נחזיר ALLOCATION_ERROR
9. אחרת נחזיר SUCCESS

SetLAbelScore(void* DS, int imageID, int pixel, int label , int score):

1. אם DS הוא NULL או imageID הוא קטן או שווה לאפס, נחזיר INVALID_INPUT.
2. אם pixel קטן מ0 או גדול שווה מpixels נחזיר INVALID_INPUT
3. אם label קטן או שווה לאפס או score קטן או שווה לאפס נחזיר INVALID_INPUT
4. נבדוק האם ה ImageID קיים בהאש (סיבוכיות משוערכת O(1))
5. אם הוא לא קיים, נחזיר FAILURE
6. ניגש לUF אליו מצביע הimageID בהאש. (סיבוכיות O(1) משוערך)
7. ניגש לקבוצה של הסופר פיקסלים של הפיקסל pixel באמצעות הפונקציה Find ((log*(k))
8. נבדוק האם קיים תיוג label בעץ . (סיבוכיות O(log(m))
9. אם כן, נשנה את הניקוד שלה. נמחק את הניקוד הקודם ונוסיף את הניקוד החדש מעץ הניקוד. (סיבוכיות O(log(m))
10. אם לא, נוסיף כזו לעץ עם הניקוד score (O(log(m) כפול 2 כיוון שיש שני עצים, 2 הוא קבוע ולכן אותו הדבר)
11. במקרה של בעיה בהקצאת הזיכרון נחזיר ALLOCATION_ERROR
12. אחרת נחזיר SUCCESS

סיבוכיות זמן: log*(k)+log(m) כנדרש.

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

1. אם DS הוא NULL או imageID הוא קטן או שווה לאפס, נחזיר INVALID_INPUT.
2. אם pixel קטן מ0 או גדול שווה מpixels נחזיר INVALID_INPUT
3. אם label קטן או שווה לאפס נחזיר INVALID_INPUT
4. נבדוק האם ה ImageID קיים בהאש (סיבוכיות משוערכת O(1))
5. אם הוא לא קיים, נחזיר FAILURE
6. נמצא את הסופר פיקסל של pixel באמצעות Find (O(log*(k)) משוערך)
7. ניגש לעץ התיוגים של הסופר פיקסל ונחפש בו את label (O(log(m))
8. אם אין צומת label בעץ נחזיר FAILURE
9. אם היא קיימת, נסיר אותה משני העצים (O(log(m)))
10. במקרה של בעיה בהקצאת הזיכרון נחזיר ALLOCATION_ERROR
11. אחרת נחזיר SUCCESS

סיבוכיות זמן: O(log*k + logm) כנדרש.

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

1. אם DS הוא NULL או imageID הוא קטן או שווה לאפס, נחזיר INVALID_INPUT.
2. אם pixel קטן מ0 או גדול שווה מpixels או label הוא NULL נחזיר INVALID_INPUT
3. נבדוק האם ה ImageID קיים בהאש (סיבוכיות משוערכת O(1))
4. אם הוא לא קיים, נחזיר FAILURE
5. נמצא את הסופר פיקסל של pixel באמצעות Find (O(log*(k)) משוערך)
6. נבדוק האם העץ של התיוגים שלו ריק.
7. אם כן, נחזיר FAILURE
8. ניגש לעץ הממוין לפי ניקוד ואז תיוג, וניגש לפוינטר שמצביע לאיבר המקסימלי. (O(1))
9. נשמור בlabel את התיוג עם הניקוד הכי גבוה.
10. במקרה של בעיה בהקצאת הזיכרון נחזיר ALLOCATION_ERROR
11. אחרת נחזיר SUCCESS

סיבוכיות זמן: O(log*k) כנדרש.

:MergeSuperPixels(void* DS, int imageID, int pixel1, int pixel1)

1. אם DS הוא NULL או imageID הוא קטן או שווה לאפס, נחזיר INVALID_INPUT.
2. אם pixel1 או pixel2 קטן מ0 או גדול שווה מpixels נחזיר INVALID_INPUT.
3. נבדוק האם ה ImageID קיים בהאש (סיבוכיות משוערכת O(1))
4. אם הוא לא קיים, נחזיר FAILURE
5. נבצע Find ל pixel1 ול pixel2 ונבדוק האם הם שייכים לאותה הקבוצה
6. אם כן, נחזיר FAILURE
7. אחרת, נבצע Union לשתי הקבוצות, וכאשר נאחד את העצים שלהן נעביר קודם את העצים למערכים ממוינים ע"י inorder, נבצע איחוד שלהם למערך ממוין אחד בO(m) ואז ניצור מהם עצים חדשים (נבצע זאת לכל אחד משני העצים שלנו). כאשר אנו מאחדים את המערכים אנו מעדכנים עבור תיוגים זהים את הניקוד להיות הסכום של הניקודים הישנים שלהם.
8. במקרה של בעיה בהקצאת הזיכרון נחזיר ALLOCATION_ERROR

9. אחרת נחזיר SUCCESS

סיבוכיות זמן: O(log*k+m) בנדרש

Quit(void** DS):

1. אם DS הוא NULL נסיים
2. נעבור על ההאש של התמונות, עבור כל תמונה ניגש לUF אליה היא מצביע.
3. עבור כל UF נמחק מכל סופר פיקסל את העצים של התיוגים שלו, ונמחק את כל הפיקסלים
4. נשחרר את הUF ואת ההאש לאחר שסיימנו
5. נשנה את הערך לNULL.

סיבוכיות זמן: $O(n \cdot k \cdot m)$ בנדרש.