החלק היבש בתרגיל בית רטוב 2 במבני נתונים

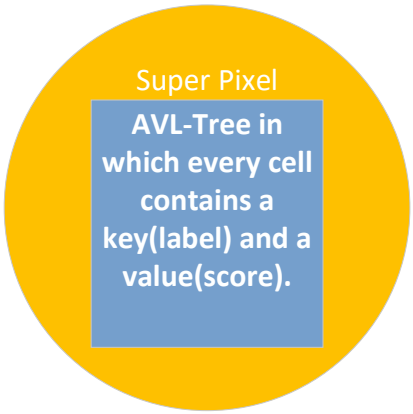
מגישים:

גיא אוחיון, ת"ז 315823856

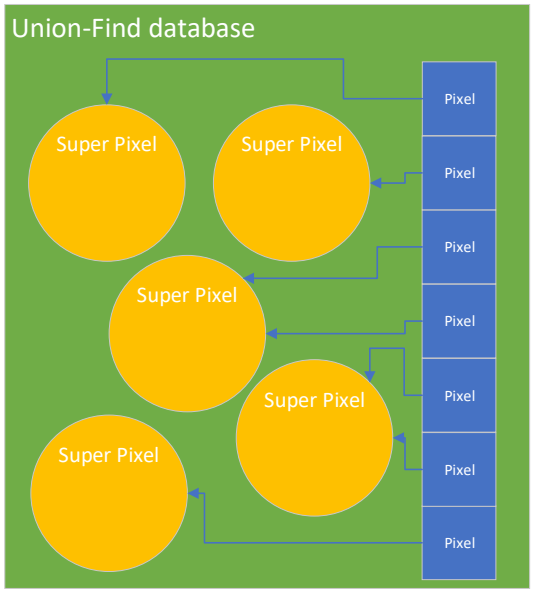
רון קנטורוביץ', ת"ז 207733015

תחילה נתאר את תכולת מבנה הנתונים והאובייקטים שאנו משתמשים בהם, ולאחר מכן נעבור לפירוט מעמיק לגבי המתודות על מנת להוכיח את הסיבוכיות שלהן. נתאר את מבנה הנתונים מהאובייקט הכי פנימי ועד המעטפת הכללית והחיצונית ביותר:

- Super Pixel: אובייקט שמייצג קבוצה אחת של פיקסלים, ומאחסן תיוגים וניקודים בעץ דרגות AVL (עבור כל תיוג יש ניקוד יחיד). התיוגים יהיו המפתחות בעץ, והניקוד יהיה הערך בכל צומת (כל ניקוד כמובן שייך למפתח ספציפי). בנוסף, בכל צומת x בעץ נשמור את הדרגה שלו, כאשר הדרגה היא הניקוד הגבוהה ביותר בתת העץ שהצומת x הוא השורש שלו. לכן, בשורש העץ קיים הניקוד הגבוהה ביותר, ולכן שליפה של הניקוד הגבוהה ביותר מהעץ מתבצעת בסיבוכיות זמן $O(1)$.

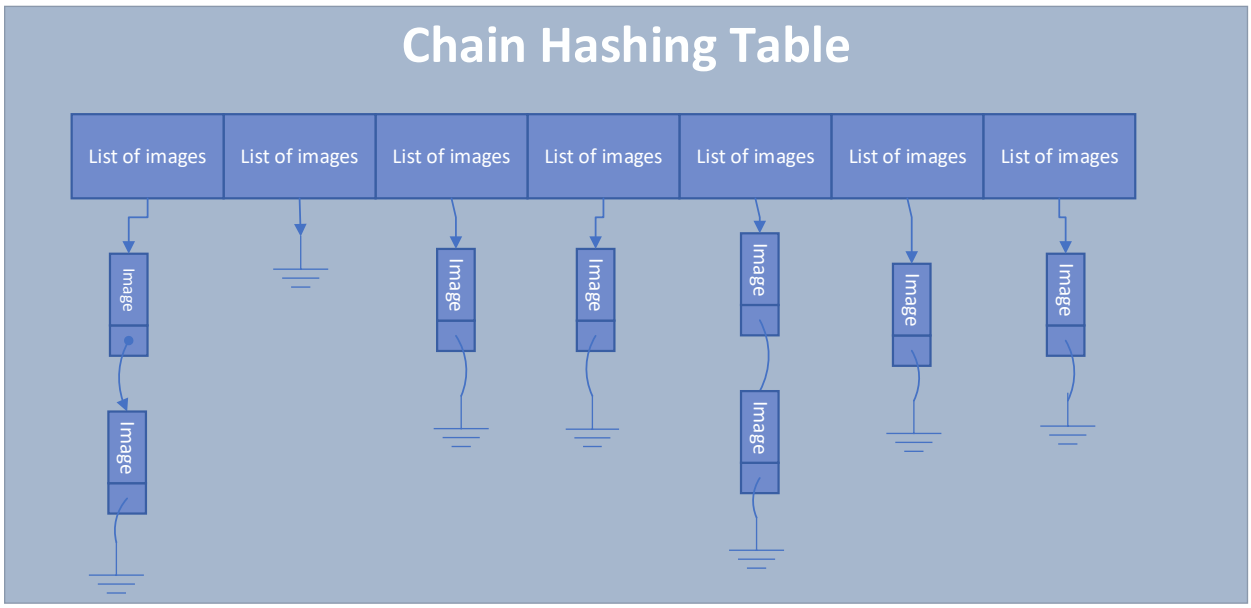


- Union-Find database: מאגר נתונים שמאחסן Super Pixels, תוך שימוש במבנה נתונים מסוג Union-Find עם עצים הפוכים וכיווץ מסלולים לאחר הפעולה Find (למדנו בקורס). בנוסף, במאגר הנתונים אנו שומרים את כל הפיקסלים ששייכים לתמונה כלשהי, כך שכל פיקסל משויך ל- Super Pixel אחד ויחיד.



- Image: מעטפת פשוטה למאגר הנתונים של ה-Super Pixels. בכל תמונה יש קבוצות שונות של Super Pixels, ובנוסף מזהה תמונה.

- Hash Table דינמי מסוג Chain Hashing שלמדנו בקורס. מאחסן תמונות לפי המזהה של כל תמונה (מספר שלם), תוך שימוש בפונקציית ערבול דינמית (מודולו גודל המערך).



כעת נסביר כיצד עובדות כל המתודות במבנה הנתונים:

- $Init(int\ pixels)$: נאתחל משתנה מהמחלקה Hash Table שייצרנו, כך שגודל המערך יהיה 50. בכל אחד מהתאים במערך נבצע השמה לפוינטר $null$. לבסוף, נחזיר מצביע למבנה הנתונים. אתחול מערך בגודל קבוע, וביצוע השמות אל מערך זה, מתבצעים **בסיבוכיות זמן ומקום $O(1)$ במקרה הגרוע.**
- $AddImage(void * DS, int\ imageID)$: נבדוק את תקינות הפרמטרים שהתקבלו, ונחזיר שגיאות מתאימות במידה ואחד מהן לא תקין בסיבוכיות זמן $O(1)$. אחרת, נחשב את האינדקס של התמונה המתאים ל-Hash Table על פי מזהה התמונה $imageID$ תוך שימוש בפונקציית ה-Hash המוגדרת במבנה הנתונים (ביצוע $i = imageID(mod)table\_size$). אם התמונה קיימת, נחזיר שגיאה ונסיים עם סיבוכיות משוערכת $O(1)$ כפי שהוסבר בקורס. אחרת, נייצר תמונה בעלת מזהה $imageID$, ובנוסף נכניס את התמונה לרשימה המקושרת הנמצאת באינדקס i של מערך ה- Hash Table השייך למבנה הנתונים. לאחר מכן, נבנה את מבנה הנתונים $Union - Find$ השייך לתמונה ומכיל את כל הסופר פיקסלים. מכיוון שיש לנו k פיקסלים, סיבוכיות הזמן לבנות את המאגר $Union - Find$ תהיה $O(k)$ (כי כל פיקסל הוא בפרט סופר פיקסל ריק בהתחלה). כעת נבדוק מהי כמות התמונות הכוללת ב-Hash Table לאחר הכנסת התמונה בהשוואה לגודל של ה-Hash Table עצמו, כדי לדעת האם נדרש להגדיל את הטבלה (אנו שומרים את גודל ה-Hash Table הכולל ואת כמות

התמונות הקיימות במערכת, ולכן נקבל את גודלו ואת כמות התמונות בסיבוכיות $O(1)$. אם ה- Hash Table מלא (כלומר אם יש כמות תמונות שהיא בדיוק הגודל הכולל של הטבלה), **נגדיל** את גודלו פי 2, נעדכן את פונקציית ה-Hash בהתאם, ונעדכן את כל התמונות שהיו בטבלה הקודמת לטבלה המתאימה לפונקציית ה-Hash החדשה (בדיוק כפי שלמדנו בתרגול – מערך דינמי של Hash Table). כלומר, נוציא את כל התמונות מהטבלה הקודמת, נגדיר פונקציית Hash חדשה שמתאימה לגודל המערך החדש, ונכניס את כל התמונות מחדש לטבלה תוך שימוש בפונקציית ה-Hash החדשה. הוצאה והכנסה מחדש למערך נכללת בסיבוכיות משוערכת $O(1)$ למבנה ה-Hash Table (כפי שלמדנו בתרגול), ולכן אינה הורסת את הסיבוכיות של מתודה זאת. ייצור תמונה, ביצוע כמות סופית של חישובים, והכנסת איבר לראש רשימה מקושרת מתבצעים בסיבוכיות זמן ומקום $O(1)$ במקרה הגרוע. בנייה של מבנה $Union - Find$ בגודל k מתבצעת בסיבוכיות זמן ומקום $O(k)$, כפי שלמדנו בקורס. מכיוון שאנו משתמשים ב$Chain\ Hasihng$, סיבוכיות הזמן המשוערכת בממוצע על הקלט עבור מחיקה והוספה של איברים ל-Hash Table היא $O(1)$. לכן, סיבוכיות הזמן המשוערכת של מתודה זו תושפע מ-k, כמות הפיקסלים בתמונה. כלומר, סיבוכיות הזמן המשוערכת בממוצע על הקלט היא $O(k)$, כנדרש.

- $DeleteImage(void * DS, int\ imageID)$: נבדוק את תקינות הפרמטרים שהתקבלו, ונחזיר שגיאות מתאימות במידה ואחד מהן לא תקין בסיבוכיות זמן $O(1)$. נחשב את האינדקס של התמונה המתאים ל-Hash Table על פי מזהה התמונה $imageID$ תוך שימוש בפונקציית ה-Hash המוגדרת במבנה הנתונים (נסמן את האינדקס ב-k). אם התמונה לא קיימת, נחזיר ערך שגיאה. אחרת, נשלוף את התמונה מהרשימה המקושרת המתאימה לה בטבלת ה-Hash, ונמחק את התמונה. נתחיל בלמחוק את כל העצים בסופר פיקסלים השייכים לתמונה: בתמונה יש בסה"כ m תיוגים, ולכן הגודל הכולל של כל העצים הנמצאים בסופר פיקסלים השייכים לתמונה הוא m, ולכן סיבוכיות הזמן למחיקת כל העצים בכל הסופר פיקסלים היא $O(m)$. נמחק כעת את מבנה הנתונים $Union - Find$ שמאחסן את כל הסופר פיקסלים. התחלנו עם k סופר פיקסלים ולכן יש לנו לכל היותר k קבוצות, ולכן סיבוכיות הזמן למחיקה היא $O(k)$. נוציא את התמונה מהרשימה המקושרת שהיא נמצאת בה, נעדכן את הרשימה בהתאם, ואז נמחק את התמונה. כעת נבדוק מהי כמות התמונות הכוללת ב-Hash Table לאחר הוצאת התמונה בהשוואה לגודל של ה-Hash Table עצמו, כדי לדעת האם נדרש להקטין את הטבלה (אנו שומרים את גודל ה-Hash Table הכולל ואת כמות התמונות הקיימות במערכת, ולכן נקבל את גודלו ואת כמות התמונות בסיבוכיות $O(1)$). אם ה-Hash Table רבע מלא (כלומר אם יש כמות תמונות שהיא רבע מהגודל הכולל של הטבלה), **נקטין** את גודלו פי 2 (כפי שלמדנו), נעדכן את פונקציית ה-Hash בהתאם, ונעדכן את כל התמונות שהיו בטבלה הקודמת לטבלה המתאימה לפונקציית ה-Hash החדשה (בדיוק כפי שלמדנו בתרגול – מערך דינמי של Hash Table). כלומר, נוציא את כל התמונות מהטבלה הקודמת, נגדיר פונקציית Hash חדשה שמתאימה לגודל המערך החדש, ונכניס את כל התמונות מחדש לטבלה תוך שימוש בפונקציית ה-Hash החדשה. הוצאה והכנסה מחדש למערך נכללת בסיבוכיות משוערכת $O(1)$ למבנה ה-Hash Table (כפי שלמדנו בתרגול), ולכן אינה הורסת את הסיבוכיות של מתודה זאת. ללא מחיקת העצים ומבנה הנתונים $Union - Find$, סיבוכיות הזמן המשוערכת בממוצע על הקלט למחיקת תמונה היא $O(1)$, כפי שלמדנו בתרגול. לכן ,סיבוכיות הזמן המשוערכת בממוצע על הקלט של מתודה זו תהיה $O(k + m)$, כנדרש.
- $SetLabelScore(void * DS, int\ imageID, int\ pixel, int\ label, int\ score)$: נבדוק את תקינות הפרמטרים שהתקבלו, ונחזיר שגיאות מתאימות במידה ואחד מהן לא תקין בסיבוכיות זמן $O(1)$. נחפש את התמונה בטבלת ה-Hash בסיבוכיות זמן $O(1)$ בממוצע על הקלט, ובמידה והיא לא

נמצאת נחזיר שגיאה. אחרת, ניגש למבנה הנתונים $Union - Find$ הנמצא בתמונה, ששומר את קבוצות הסופר פיקסלים השייכים לתמונה. נמצא את הקבוצה אליה $pixel$ שייך בסיבוכיות משוערכת $log * (k)$ (למדנו בהרצאות). לאחר מכן, ניגש לעץ AVL הנמצא בסופר פיקסל שמצאנו (שהוא ראש הקבוצה) ונבצע הכנסה\עדכון של $score$ למפתח $label$ המתאים בעץ בסיבוכיות זמן $O(\log(m))$ כאשר m הוא כמות התיוגים הקיימים בעץ (הכנסה לעץ AVL בגודל m מתבצעת בסיבוכיות $O(\log(m))$). לאחר ההכנסה של הצומת החדש לעץ, ייתכן כי הרסנו את מבנה הדרגות שלו (כלומר ייתכן כעת כי בשורש לא תהיה הדרגה הגבוהה ביותר – ערך הניקוד הגבוהה ביותר בעץ – כי אולי הצומת שהכנסנו הוא בעל הניקוד הגבוהה ביותר). לכן, נעבור מסוף מסלול החיפוש של הצומת שהכנסנו עד לשורש העץ, ולכל צומת v במסלול נעדכן את הדרגה להיות המקסימלית מבין הדרגה של תת העץ השמאלי, תת העץ הימני, לבין הניקוד של צומת v עצמו. בדרך זו נשמור על מבנה הדרגות כי רק הצמתים במסלול ההכנסה יושפעו מהניקוד של הצומת החדש שהכנסנו. מעבר חוזר על מסלול ההכנסה מהתחתית אל השורש, וביצוע מספר סופי של השוואות והשמות מתבצעים ב-$O(\log(m))$.
לכן, סיבוכיות הזמן המשוערכת בממוצע על הקלט היא

$$O(log * (k) + \log(m))$$

כנדרש.

- $ResetLabelScore(void * DS, int\ imageID, int\ pixel, int\ label)$: נבדוק את תקינות הפרמטרים שהתקבלו, ונחזיר שגיאות מתאימות במידה ואחד מהן לא תקין בסיבוכיות זמן $O(1)$. נחפש את התמונה בטבלת ה-Hash בסיבוכיות זמן $O(1)$ בממוצע על הקלט, ובמידה והיא לא נמצאת נחזיר שגיאה. אחרת, ניגש למבנה הנתונים $Union - Find$ הנמצא בתמונה, ששומר את קבוצות הסופר פיקסלים השייכים לתמונה. נמצא את הקבוצה אליה $pixel$ שייך בסיבוכיות משוערכת $O(log * (k))$ (למדנו בהרצאות). לאחר מכן, ניגש לעץ AVL הנמצא בסופר פיקסל שמצאנו (שהוא ראש הקבוצה) ונבצע הוצאה של המפתח (התיוג) $label$ המתאים בעץ בסיבוכיות זמן $O(\log(m))$ כאשר m הוא כמות התיוגים הקיימים בעץ (הוצאה מעץ AVL בגודל m מתבצעת בסיבוכיות $O(\log(m))$). לאחר ההוצאה של הצומת מהעץ, ייתכן כי הרסנו את מבנה הדרגות שלו (כלומר ייתכן כעת כי בשורש לא תהיה הדרגה הגבוהה ביותר – ערך הניקוד הגבוהה ביותר בעץ – כי אולי הצומת שהוצאנו הוא בעל הניקוד הגבוהה ביותר). לכן, נעבור מסוף מסלול החיפוש של הצומת שהוצאנו עד לשורש העץ, ולכל צומת v במסלול נעדכן את הדרגה להיות המקסימלית מבין הדרגה של תת העץ השמאלי, תת העץ הימני, לבין הניקוד של צומת v עצמו. בדרך זו נשמור על מבנה הדרגות כי רק הצמתים במסלול ההוצאה יושפעו מהניקוד של הצומת החדש שהכנסנו. מעבר חוזר על מסלול ההוצאה מהתחתית אל השורש, וביצוע מספר סופי של השוואות והשמות מתבצעים ב-$O(\log(m))$.
לכן, סיבוכיות הזמן המשוערכת בממוצע על הקלט היא

$$O(log * (k) + \log(m))$$

כנדרש.

- $GetHighestScoreLabel(void * DS, int\ imageID, int\ pixel, int *\ label)$: נבדוק את תקינות הפרמטרים שהתקבלו, ונחזיר שגיאות מתאימות במידה ואחד מהן לא תקין בסיבוכיות זמן $O(1)$. נחפש את התמונה בטבלת ה-Hash בסיבוכיות זמן $O(1)$ בממוצע על הקלט, ובמידה והיא לא נמצאת נחזיר שגיאה. אחרת, ניגש למבנה הנתונים $Union - Find$ הנמצא בתמונה, ששומר את קבוצות הסופר פיקסלים השייכים לתמונה. נמצא את הקבוצה (הסופר פיקסל) אליה $pixel$ שייך בסיבוכיות משוערכת
$O(log * (k))$ (למדנו בהרצאות). ניגש לעץ ה-AVL המצא בסופר פיקסל הזה, שהוא גם עץ דרגות,

ונשלוף מהשורש את הניקוד הגבוהה ביותר בעץ בסיבוכיות זמן $O(1)$. לבסוף, נחזיר למשתמש את הערך. לכן סיבוכיות הזמן המשוערכת בממוצע על הקלט היא $O(log*(k))$.

- $MergeSuperPixels(void*DS, int\ imageID, int\ pixel1, int\ pixel2)$: נבדוק את תקינות הפרמטרים שהתקבלו, ונחזיר שגיאות מתאימות במידה ואחד מהן לא תקין בסיבוכיות זמן $O(1)$.
נחפש את התמונה בטבלת ה-Hash בסיבוכיות זמן $O(1)$ בממוצע על הקלט, ובמידה והיא לא נמצאת נחזיר שגיאה. אחרת, ניגש למבנה הנתונים $Union-Find$ הנמצא בתמונה, ששומר את קבוצות הסופר פיקסלים השייכים לתמונה. נמצא את 2 הקבוצות (הסופר פיקסלים) אליהן שייכים $pixel1, pixel2$ בסיבוכיות משוערכת $O(log*(k))$
(למדנו בהרצאות). אם שני הפיקסלים שייכים לאותו הסופר פיקסל, נחזיר שגיאה. אחרת, נשלוף מ-2 הסופר פיקסלים שמצאנו את עצי הדרגות שלהם (נסמן את העצים ב-$k_{1,2}$. נעתיק את כל אחד מהעצים למערך באלגוריתם $InOrder$ בסיבוכיות $O(m)$ עבור כל עץ, כאשר m הוא גובה כל עץ.
נקבל 2 מערכים ממוינים בסדר עולה, כאשר כל אחד מהם שומר את כל התיוגים והניקודים של כל אחד מהעצים $k_{1,2}$. נאחד את 2 המערכים למערך ממוין אחד (בסיבוכיות $O(m_1+m_2)$ כאשר m_1+m_2 הוא סכום גדלי העצים), ואם ניתקל בשני תיוגים זהים, נסכום את הניקוד שלהם כנדרש. נבנה עץ כמעט שלם, שהוא בפרט עץ AVL, בעזרת המערך המאוחד והממוין שיצרנו. על מנת לעשות זאת, נשתמש באלגוריתם שלמדנו בתרגול על עצים בינאריים, שקופיות 9-11 (אלגוריתם לאיחוד עצים בינאריים). לפי מה שלמדנו, סיבוכיות הזמן של האיחוד היא

$$O(m_1+m_2)$$

כאשר m_1+m_2 הוא סכום גדלי העצים שאיחדנו. לכן, סיבוכיות הזמן המשוערכת בממוצע על הקלט היא

$$O(log*(k)+m)$$

כאשר $m=m_1+m_2$, סכום גדלי העצים שתייגנו, שהוא גם מספר התיוגים בשני הסופר פיקסלים.

$Quit(void^{**}DS)$:

ראשית נבדוק האם DS הוא מצביע $NULL$, ואם כן נחזור מהפונקציה.

אחרת, נחלץ מ־DS בגישה ישירה את המבנה ונשחרר את הזיכרון המוקצה לו באופן הבא:

נעבור על טבלת הערבול של התמונות, ועבור כל תמונה נשחרר את מבנה ה־$UnionFind$ בתוכה:

נעבור על מערך ה־$Data$ שמכיל מצביעים לאובייקטי הסופר־פיקסל, ועבור כל אובייקט סופר פיקסל נשחרר את העץ שבתוכו:

נעבור על העץ ב־$PostOrder$ ונמחק רקורסיבית כל צומת.

בסך הכול, מחיקת עץ AVL בשיטה שתיארנו היא $O(m_i)$, כאשר m_i הוא מספר התיוגים בעץ ה־i, שכן קיימת התאמה חח"ע בין התיוגים של כל סופר־פיקסל למספר הצמתים בעץ שהוא מכיל. בפרט $m_i < m$ לכל i, כלומר מספר התיוגים בכל עץ קטן ממספר התיוגים הכולל.

לכן פעולת מחיקת סופר־פיקסל והעץ שלו מתבצעת ב־$O(m)$.

פעולה זו מתבצעת עבור כל אובייקט $Data$ ב־$UnionFind$, ובנוסף סיבוכיות הריסת ה־$UnionFind$ וה־$Image$ שלו היא סיבוכיות המעבר על המערכים שבאמצעותם הוא ממומש, כלומר $O(k)$ כאשר k מספר הפיקסלים (כל פיקסל מתאים לתא במערך).

במקרה הגרוע ביותר, כל פיקסל יהיה סופר־פיקסל משל עצמו, ולכן עבור כל פיקסל נצטרך להרוס עץ, כלומר פעולת הריסת תמונה וכל הזיכרון שבתוכה היא $O(k \cdot m)$.

את פעולה זו אנו מבצעים עבור כל תמונה בטבלת הערבול, כלומר אנו עוברים על כל התאים שבה ועל כל הרשימות בכל התאים, ומכיוון שהטבלה דינאמית וגודלה הוא תמיד $O(n)$, נקבל שהריסת המבנה הכוללת היא בסיבוכיות $O(n \cdot k \cdot m)$ כאשר n מספר התמונות, כדרוש.

סיבוכיות מקום:

סיבוכיות המקום של עץ AVL היא כמספר הצמתים שלו, ובמקרה שלנו זהו מספר התיוגים, שבפרט קטן מ־m מספר התיוגים בכל המערכת.

סיבוכיות המקום של מבנה $UnionFind$ במימוש שראינו בהרצאה היא ליניארית במספר המפתחות (האיברים), ובמקרה שלנו זהו מספר הפיקסלים בכל תמונה k, אך בנוסף במקרה הגרוע כל פיקסל הוא סופר פיקסל שמכיל עץ AVL, ולכן במקרה הגרוע סיבוכיות המקום של מבנה $UnionFind$ היא $O(k \cdot m)$.

סיבוכיות המקום של טבלת ערבול דינאמית כפי שראינו בתרגול היא ליניארית במספר האיברים, ובמקרה שלנו זהו מספר התמונות n, אך בנוסף כל תמונה מכילה בתוכה $UnionFind$ של עצים, ולכן במקרה הגרוע סיבוכיות המקום של המבנה כולו היא $O(n \cdot k \cdot m)$ כנדרש.