# מבני נתונים

## 234218

# תרגיל רטוב 1

**מגישים: רוני אנגלנדר - 312168354,**

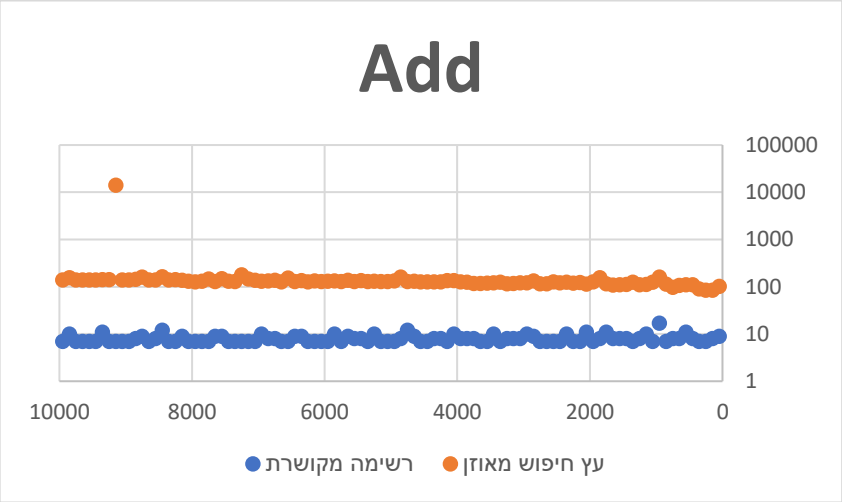
**נדב אורזך - 311549455**

**תאריך הגשה: 16.12.2018**

# חלק ראשון – טבלאות וגרפים

## ADD FUNCTION

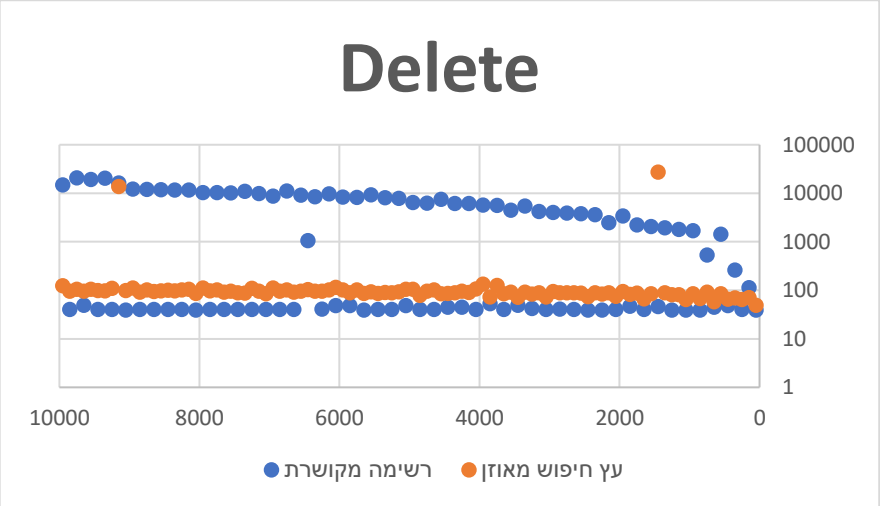
| גודל הקלט | רשימה מקושרת | עץ חיפוש מאוזן |
|---|---|---|
| 50 | 9 | 102 |
| 150 | 8 | 85 |
| 250 | 7 | 85 |
| 350 | 7 | 90 |
| 450 | 8 | 111 |
| 550 | 11 | 111 |
| 650 | 8 | 108 |
| 750 | 8 | 98 |
| 850 | 7 | 115 |
| 950 | 17 | 160 |
| 1050 | 7 | 125 |
| 1150 | 10 | 112 |
| 1250 | 8 | 110 |
| 1350 | 7 | 126 |
| 1450 | 8 | 112 |
| 1550 | 8 | 110 |
| 1650 | 8 | 109 |
| 1750 | 11 | 117 |
| 1850 | 8 | 157 |
| 1950 | 7 | 127 |
| 2050 | 11 | 114 |
| 2150 | 7 | 124 |
| 2250 | 7 | 120 |
| 2350 | 10 | 125 |
| 2450 | 7 | 121 |
| 2550 | 7 | 128 |
| 2650 | 7 | 117 |
| 2750 | 7 | 117 |
| 2850 | 9 | 133 |
| 2950 | 10 | 122 |
| 3050 | 8 | 122 |
| 3150 | 8 | 118 |
| 3250 | 8 | 117 |
| 3350 | 7 | 126 |
| 3450 | 10 | 121 |
| 3550 | 7 | 119 |
| 3650 | 7 | 118 |
| 3750 | 8 | 118 |
| 3850 | 8 | 126 |
| 3950 | 8 | 127 |
| 4050 | 10 | 136 |
| 4150 | 7 | 136 |
| 4250 | 8 | 128 |
| 4350 | 8 | 128 |
| 4450 | 7 | 128 |
| 4550 | 7 | 128 |
| 4650 | 9 | 132 |
| 4750 | 12 | 129 |



| | | |
|---|---|---|
| 161 | 8 | 4850 |
| 133 | 7 | 4950 |
| 129 | 7 | 5050 |
| 129 | 7 | 5150 |
| 132 | 10 | 5250 |
| 130 | 7 | 5350 |
| 136 | 8 | 5450 |
| 129 | 8 | 5550 |
| 137 | 9 | 5650 |
| 130 | 7 | 5750 |
| 134 | 10 | 5850 |
| 132 | 7 | 5950 |
| 130 | 7 | 6050 |
| 133 | 7 | 6150 |
| 128 | 7 | 6250 |
| 136 | 9 | 6350 |
| 130 | 9 | 6450 |
| 154 | 7 | 6550 |
| 128 | 7 | 6650 |
| 137 | 8 | 6750 |
| 133 | 8 | 6850 |
| 132 | 10 | 6950 |
| 137 | 7 | 7050 |
| 146 | 7 | 7150 |
| 181 | 7 | 7250 |
| 129 | 7 | 7350 |
| 131 | 7 | 7450 |
| 150 | 9 | 7550 |
| 129 | 9 | 7650 |
| 147 | 7 | 7750 |
| 131 | 7 | 7850 |
| 128 | 7 | 7950 |
| 132 | 7 | 8050 |
| 138 | 9 | 8150 |
| 142 | 7 | 8250 |
| 139 | 7 | 8350 |
| 164 | 12 | 8450 |
| 139 | 8 | 8550 |
| 141 | 7 | 8650 |
| 161 | 9 | 8750 |
| 145 | 8 | 8850 |
| 139 | 7 | 8950 |
| 139 | 7 | 9050 |
| 14155 | 7 | 9150 |
| 142 | 7 | 9250 |
| 142 | 11 | 9350 |
| 139 | 7 | 9450 |
| 141 | 7 | 9550 |
| 139 | 7 | 9650 |
| 140 | 7 | 9750 |
| 157 | 10 | 9850 |
| 140 | 7 | 9950 |

# DELETE FUNCTION

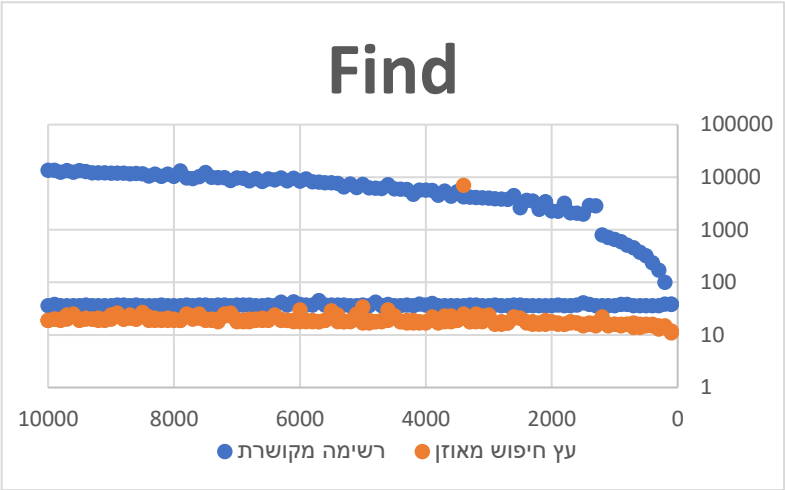
| גודל הקלט | רשימה מקושרת | עץ חיפוש מאוזן |
|---|---|---|
| 50 | 39 | 49 |
| 150 | 113 | 71 |
| 250 | 40 | 64 |
| 350 | 262 | 69 |
| 450 | 48 | 65 |
| 550 | 1436 | 85 |
| 650 | 45 | 58 |
| 750 | 530 | 92 |
| 850 | 39 | 67 |
| 950 | 1680 | 84 |
| 1050 | 39 | 64 |
| 1150 | 1795 | 81 |
| 1250 | 39 | 81 |
| 1350 | 1927 | 89 |
| 1450 | 46 | 27099 |
| 1550 | 2066 | 85 |
| 1650 | 40 | 66 |
| 1750 | 2216 | 87 |
| 1850 | 47 | 82 |
| 1950 | 3379 | 94 |
| 2050 | 40 | 75 |
| 2150 | 2461 | 89 |
| 2250 | 39 | 83 |
| 2350 | 3628 | 89 |
| 2450 | 39 | 74 |
| 2550 | 3793 | 87 |
| 2650 | 40 | 89 |
| 2750 | 3912 | 88 |
| 2850 | 41 | 89 |
| 2950 | 4008 | 94 |
| 3050 | 40 | 73 |
| 3150 | 4203 | 89 |
| 3250 | 42 | 83 |
| 3350 | 5441 | 91 |
| 3450 | 49 | 72 |
| 3550 | 4456 | 92 |
| 3650 | 40 | 84 |
| 3750 | 5628 | 126 |
| 3850 | 53 | 73 |
| 3950 | 5734 | 134 |
| 4050 | 40 | 106 |
| 4150 | 6174 | 90 |
| 4250 | 45 | 96 |
| 4350 | 6163 | 89 |
| 4450 | 45 | 86 |
| 4550 | 7435 | 85 |
| 4650 | 40 | 102 |
| 4750 | 6271 | 95 |



| | | |
|---|---|---|
| 79 | 40 | 4850 |
| 104 | 6446 | 4950 |
| 104 | 48 | 5050 |
| 93 | 7860 | 5150 |
| 88 | 40 | 5250 |
| 90 | 8019 | 5350 |
| 86 | 40 | 5450 |
| 93 | 9186 | 5550 |
| 86 | 39 | 5650 |
| 102 | 8143 | 5750 |
| 89 | 48 | 5850 |
| 101 | 8315 | 5950 |
| 115 | 48 | 6050 |
| 102 | 9727 | 6150 |
| 95 | 41 | 6250 |
| 96 | 8444 | 6350 |
| 103 | 1051 | 6450 |
| 95 | 9052 | 6550 |
| 91 | 40 | 6650 |
| 102 | 11188 | 6750 |
| 96 | 40 | 6850 |
| 111 | 8733 | 6950 |
| 85 | 40 | 7050 |
| 95 | 9836 | 7150 |
| 110 | 40 | 7250 |
| 87 | 10952 | 7350 |
| 88 | 40 | 7450 |
| 95 | 10100 | 7550 |
| 91 | 40 | 7650 |
| 102 | 10237 | 7750 |
| 97 | 40 | 7850 |
| 111 | 10362 | 7950 |
| 86 | 39 | 8050 |
| 104 | 11555 | 8150 |
| 101 | 40 | 8250 |
| 97 | 11629 | 8350 |
| 100 | 40 | 8450 |
| 97 | 11844 | 8550 |
| 94 | 40 | 8650 |
| 102 | 12043 | 8750 |
| 92 | 40 | 8850 |
| 111 | 12228 | 8950 |
| 99 | 39 | 9050 |
| 13655 | 16308 | 9150 |
| 110 | 40 | 9250 |
| 97 | 20427 | 9350 |
| 98 | 40 | 9450 |
| 105 | 19103 | 9550 |
| 95 | 49 | 9650 |
| 105 | 20743 | 9750 |
| 96 | 40 | 9850 |
| 124 | 14874 | 9950 |

# FIND FUNCTION

| גודל הקלט | רשימה מקושרת | עץ חיפוש מאוזן |
|---|---|---|
| 100 | 38 | 12 |
| 100 | 39 | 11 |
| 200 | 101 | 15 |
| 200 | 39 | 14 |
| 300 | 172 | 13 |
| 300 | 36 | 15 |
| 400 | 235 | 16 |
| 400 | 36 | 15 |
| 500 | 321 | 15 |
| 500 | 36 | 16 |
| 600 | 375 | 14 |
| 600 | 36 | 16 |
| 700 | 455 | 14 |
| 700 | 36 | 17 |
| 800 | 513 | 16 |
| 800 | 38 | 16 |
| 900 | 594 | 16 |
| 900 | 38 | 15 |
| 1000 | 655 | 16 |
| 1000 | 36 | 16 |
| 1100 | 717 | 15 |
| 1100 | 36 | 16 |
| 1200 | 801 | 18 |
| 1200 | 36 | 22 |
| 1300 | 2856 | 15 |
| 1300 | 36 | 17 |
| 1400 | 2926 | 16 |
| 1400 | 38 | 17 |
| 1500 | 1991 | 15 |
| 1500 | 41 | 16 |
| 1600 | 2064 | 17 |
| 1600 | 37 | 17 |
| 1700 | 2114 | 18 |
| 1700 | 36 | 17 |
| 1800 | 3198 | 16 |
| 1800 | 36 | 16 |
| 1900 | 2249 | 16 |
| 1900 | 37 | 17 |
| 2000 | 2306 | 17 |
| 2000 | 36 | 18 |
| 2100 | 3428 | 16 |
| 2100 | 36 | 19 |
| 2200 | 2442 | 16 |
| 2200 | 36 | 17 |
| 2300 | 3522 | 16 |
| 2300 | 36 | 18 |
| 2400 | 3630 | 17 |



| | | |
|---|---|---|
| 18 | 36 | 2400 |
| 21 | 2649 | 2500 |
| 21 | 37 | 2500 |
| 21 | 4482 | 2600 |
| 22 | 37 | 2600 |
| 17 | 3787 | 2700 |
| 17 | 36 | 2700 |
| 16 | 3848 | 2800 |
| 18 | 36 | 2800 |
| 16 | 3918 | 2900 |
| 17 | 37 | 2900 |
| 22 | 3993 | 3000 |
| 24 | 36 | 3000 |
| 23 | 4050 | 3100 |
| 18 | 36 | 3100 |
| 18 | 4137 | 3200 |
| 25 | 37 | 3200 |
| 18 | 4188 | 3300 |
| 18 | 37 | 3300 |
| 7044 | 4254 | 3400 |
| 25 | 36 | 3400 |
| 19 | 5321 | 3500 |
| 19 | 36 | 3500 |
| 23 | 4387 | 3600 |
| 18 | 36 | 3600 |
| 23 | 5453 | 3700 |
| 18 | 36 | 3700 |
| 17 | 4539 | 3800 |
| 18 | 36 | 3800 |
| 22 | 5588 | 3900 |
| 21 | 40 | 3900 |
| 17 | 5667 | 4000 |
| 18 | 37 | 4000 |
| 17 | 5733 | 4100 |
| 18 | 39 | 4100 |
| 17 | 4779 | 4200 |
| 19 | 36 | 4200 |
| 17 | 5855 | 4300 |
| 19 | 37 | 4300 |
| 18 | 5929 | 4400 |
| 18 | 36 | 4400 |
| 23 | 6001 | 4500 |
| 23 | 36 | 4500 |
| 30 | 7204 | 4600 |
| 19 | 38 | 4600 |
| 18 | 6126 | 4700 |
| 19 | 36 | 4700 |
| 18 | 6216 | 4800 |
| 18 | 42 | 4800 |
| 17 | 6260 | 4900 |
| 19 | 36 | 4900 |
| 17 | 7342 | 5000 |
| 34 | 37 | 5000 |

| | | |
|---|---|---|
| 24 | 6394 | 5100 |
| 25 | 36 | 5100 |
| 18 | 7469 | 5200 |
| 19 | 36 | 5200 |
| 18 | 6536 | 5300 |
| 18 | 37 | 5300 |
| 18 | 7624 | 5400 |
| 24 | 37 | 5400 |
| 29 | 7856 | 5500 |
| 24 | 37 | 5500 |
| 19 | 7856 | 5600 |
| 19 | 37 | 5600 |
| 18 | 8087 | 5700 |
| 18 | 45 | 5700 |
| 18 | 8210 | 5800 |
| 19 | 37 | 5800 |
| 18 | 9319 | 5900 |
| 19 | 37 | 5900 |
| 18 | 8424 | 6000 |
| 30 | 38 | 6000 |
| 18 | 9573 | 6100 |
| 19 | 43 | 6100 |
| 19 | 8539 | 6200 |
| 19 | 37 | 6200 |
| 20 | 9738 | 6300 |
| 19 | 42 | 6300 |
| 24 | 8984 | 6400 |
| 23 | 36 | 6400 |
| 20 | 9347 | 6500 |
| 19 | 37 | 6500 |
| 21 | 8298 | 6600 |
| 19 | 36 | 6600 |
| 20 | 9475 | 6700 |
| 19 | 36 | 6700 |
| 20 | 8532 | 6800 |
| 18 | 37 | 6800 |
| 19 | 9622 | 6900 |
| 18 | 37 | 6900 |
| 20 | 9672 | 7000 |
| 18 | 37 | 7000 |
| 23 | 8732 | 7100 |
| 26 | 36 | 7100 |
| 25 | 9817 | 7200 |
| 23 | 37 | 7200 |
| 20 | 9872 | 7300 |
| 18 | 37 | 7300 |
| 20 | 9937 | 7400 |
| 19 | 36 | 7400 |
| 20 | 12327 | 7500 |
| 19 | 37 | 7500 |
| 25 | 10269 | 7600 |
| 21 | 37 | 7600 |
| 20 | 9506 | 7700 |

| | | |
|---|---|---|
| 23 | 36 | 7700 |
| 25 | 9768 | 7800 |
| 24 | 37 | 7800 |
| 19 | 13289 | 7900 |
| 19 | 36 | 7900 |
| 20 | 10349 | 8000 |
| 19 | 36 | 8000 |
| 21 | 11411 | 8100 |
| 19 | 36 | 8100 |
| 20 | 10474 | 8200 |
| 19 | 37 | 8200 |
| 19 | 11542 | 8300 |
| 21 | 36 | 8300 |
| 19 | 10605 | 8400 |
| 22 | 36 | 8400 |
| 25 | 11686 | 8500 |
| 27 | 36 | 8500 |
| 20 | 11849 | 8600 |
| 20 | 37 | 8600 |
| 24 | 11694 | 8700 |
| 21 | 36 | 8700 |
| 20 | 11928 | 8800 |
| 20 | 37 | 8800 |
| 26 | 11984 | 8900 |
| 23 | 37 | 8900 |
| 24 | 12010 | 9000 |
| 20 | 36 | 9000 |
| 19 | 12082 | 9100 |
| 20 | 36 | 9100 |
| 19 | 12149 | 9200 |
| 21 | 36 | 9200 |
| 20 | 12204 | 9300 |
| 20 | 36 | 9300 |
| 20 | 12842 | 9400 |
| 21 | 37 | 9400 |
| 19 | 13348 | 9500 |
| 20 | 36 | 9500 |
| 23 | 12410 | 9600 |
| 25 | 36 | 9600 |
| 20 | 13488 | 9700 |
| 24 | 36 | 9700 |
| 20 | 12538 | 9800 |
| 19 | 36 | 9800 |
| 20 | 13616 | 9900 |
| 20 | 38 | 9900 |
| 19 | 13581 | 10000 |
| 19 | 36 | 10000 |

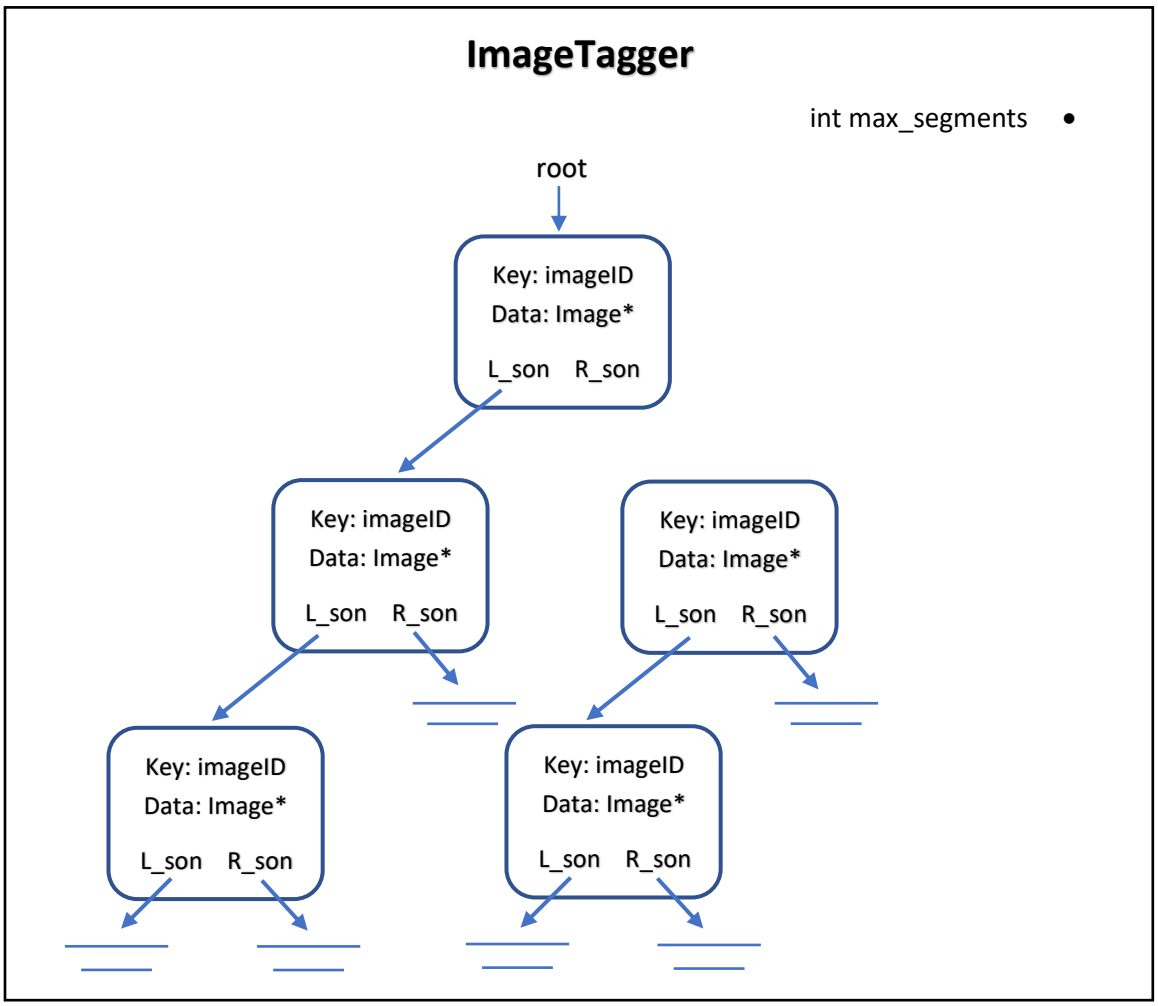
# חלק שני – תיאור מבני הנתונים

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

**ImageTagger**

מבנה נתונים ImageTagger מכיל בתוכו:

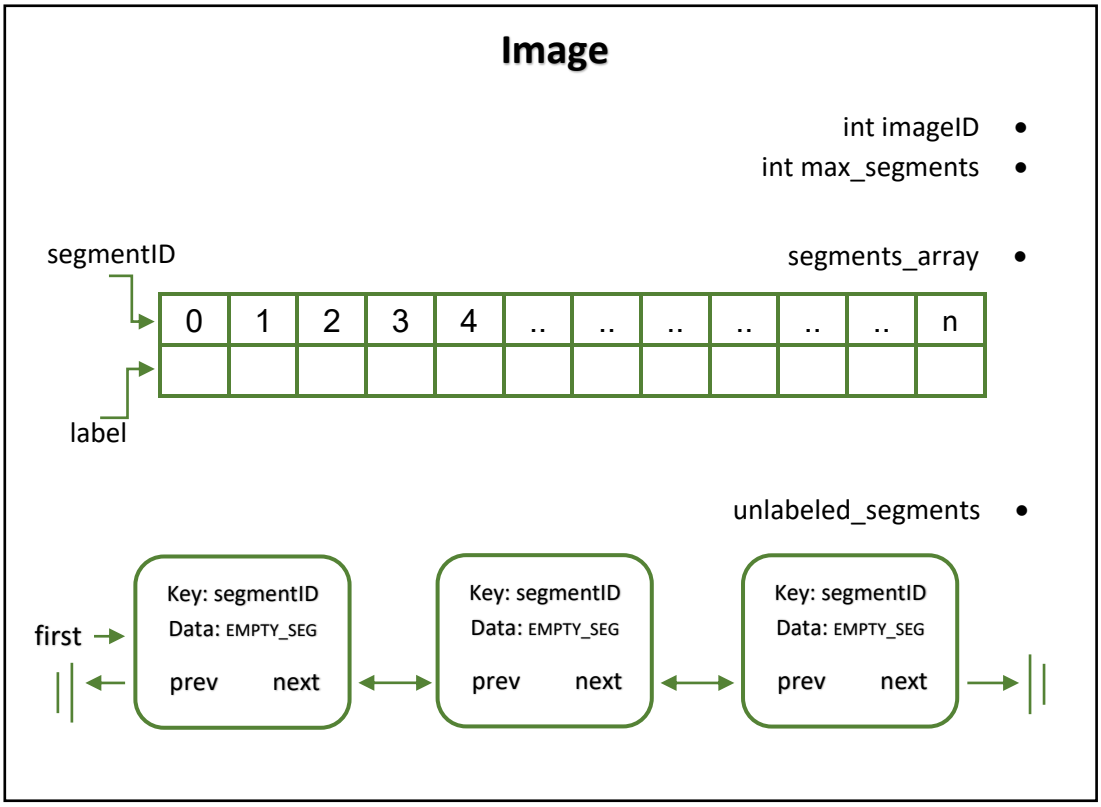
- max_segments – מספר מקסימלי של סגמנטים אפשריים לכל תמונה
- מילון - הממומש ע"י עץ AVL (Map_Tree), אשר כל צומת בעץ מכילה מפתח שהוא ImageID (int), ומידע מסוג מצביע ל- Image - מבנה נתונים שיורחב בהמשך.



**Image**

מבנה נתונים Image מכיל בתוכו:

- imageID – מזהה התמונה
- max_segments – מספר מקסימלי של סגמנטים אפשריים לתמונה
- segments_array – מערך של מספרים באורך max_segments, אשר במקום ה i מציג את התיוג של האזור (segment) ה i בתמונה. אם האזור לא מתויג, התא במערך מכיל את הערך EMPTY_SEG
- unlabeled_segments – רשימה מקושרת (Node_list) המכילה את כל האזורים הלא מתויגים בתמונה. המפתח מכיל את הsegmentID, המידע מכיל את הערך EMPTY_SEG.



**מימוש מבנה הנתונים:**

***void * Init(int segments):*** הפעולה יוצרת מבנה ImageTagger חדש, אשר מגדיר max_segments לפי הערך שנשלח לפונקציה, ויוצר מילון מסוג עץ מאוזן ריק. יצירת עץ מאוזן ריק היא פעולה בסיבוכיות זמן של O(1) ומספר הפעולות קבוע ולכן:

**סיבוכיות זמן:** O(1)

***StatusType AddImage(void *DS, int imageID):*** הפעולה מוסיפה תמונה חדשה לעץ. כפי שראינו בהרצאה הוספת צומת לעץ היא פעולה המתבצעת בסיבוכיות זמן של O(logk) כאשר k=גובה העץ.

ביצירת התמונה יוצרים:

1. מערך חדש בגודל האזורים בתמונה (segments), ומאתחלים את כל הערכים בו להיות EMPTY_SEG. משמע עוברים על מערך בגודל n.
2. רשימה מקושרת המכילה את כל האזורים שאינם מתויגים, ומכניסים בה את כל n האזורים (כי כולם אינם מתויגים בעת יצירת התמונה).

שתי פעולות אלה המתבצעות בסיבוכיות זמן של O(n) כל אחת כאשר n=segments.

שאר הפעולות ביצירת תמונה הן בסיבוכיות זמן של O(1) – אתחול הערכים imageID ו-max_segments. מכאן נקבל:

**סיבוכיות זמן:** O(logk +n)

***StatusType DeleteImage(void *DS, int imageID):*** הפעולה מסירה תמונה קיימת מהעץ.

נמצא את התמונה הרצויה בעץ, פעולה זאת מתבצעת בסיבוכיות זמן של O(logk) (כפי שלמדנו בהרצאה), כאשר k=גובה העץ.

לאחר מכן נמחק את התמונה ונסיר אותה מהעץ. במחיקת התמונה נשחרר את המערך המציג את התיוגים של כל אזור, ואת הרשימה המקושרות המציגה את האזורים הלא מתוייגים. המערך בגודל n והרשימה במקס' בגודל n , כאשר n=segments. לכן מחיקת המערך והרשימה מתבצע בסיבוכיות זמן של O(n). מכאן נקבל:

**סיבוכיות זמן:** O(logk +n)

***StatusType AddLabel(void *DS, int imageID, int segmentID, int label):*** הפעולה מוסיפה תיוג label לאזור segmentID בתמונה imageID.

נמצא את התמונה הרצויה בעץ, פעולה זאת מתבצעת בסיבוכיות זמן של O(logk) (כפי שלמדנו בהרצאה), כאשר k=גובה העץ.

נעדכן את המערך המציג תיוגים של כל אזור, כיוון שיש לנו את המיקום הנחוץ במערך (segmentID), הפעולה מתבצעת בסיבוכיות זמן של O(1).

אח"כ נסיר את האזור מהרשימה המציגה את האזורים הלא מתויגים. נמצא את האזור ברשימה, פעולה זו מתבצעת בסיבוכיות זמן של O(n), ונסיר אותה מהרשימה (סיבוכיות זמן של O(1). מכאן נקבל:

**<u>סיבוכיות זמן:</u>** O(logk +n)

***StatusType GetLabel(void *DS, int imageID, int segmentID, int *label):*** הפעולה מחזירה את התיוג של האזור המבוקש בתמונה המבוקשת.

נמצא את התמונה הרצויה בעץ, פעולה זאת מתבצעת בסיבוכיות זמן של O(logk) (כפי שלמדנו בהרצאה), כאשר k=גובה העץ.

נמצא את התיוג של האזור הרצוי ע"י גישה לתא הsegmentID במערך המציג את התיוגים של כל אזור, ונשים במצביע לlabel את הערך שמצאנו. הפעולות הנ"ל מתבצעות בסיבוכיות זמן O(1). מכאן נקבל:

**<u>סיבוכיות זמן:</u>** O(logk)

***StatusType DeleteLabel(void *DS, int imageID, int segmentID):*** הפעולה מוחקת את התיוג של האזור המבוקש בתמונה המבוקשת.

נמצא את התמונה הרצויה בעץ, פעולה זאת מתבצעת בסיבוכיות זמן של O(logk) (כפי שלמדנו בהרצאה), כאשר k=גובה העץ.

נעדכן את התיוג של האזור להיות EMPTY_SEG במערך המציג את התיוגים של כל אזור. אנו יודעים לאיזה מקום במערך לגשת, ולכן פעולה זו מתבצעת בסיבוכיות זמן של O(1).

נוסיף את האזור שזה עתה מחקנו לו את התיוג לרשימה המציגה את האזורים הלא מתויגים. הוספת איבר לרשימה מכניסה אותו להתחלה ולכן מתבצעת במספר סופי של פעולות, כלומר בסיבוכיות זמן של O(1). מכאן נקבל:

**<u>סיבוכיות זמן:</u>** O(logk)

***StatusType GetAllUnLabeledSegments(void *DS, int imageID, int **segments, int* numOfSegments):*** הפעולה מחזירה מערך של כל האזורים הלא מתויגים בתמונה.

נמצא את התמונה הרצויה בעץ, פעולה זאת מתבצעת בסיבוכיות זמן של O(logk) (כפי שלמדנו בהרצאה), כאשר k=גובה העץ.

נעבור על כל הצמתים ברשימה המציגה את האזורים הלא מתויגים בתמונה, וכל צומת נכניס למערך הרצוי. פעולה זו מתבצעת בסיבוכיות זמן O(s), כאשר s= מספר אזורים לא מתויגים (אורך הרשימה). מכאן נקבל:

**<u>סיבוכיות זמן:</u>** O(logk +s)

***StatusType GetAllSegmentsByLabel(void *DS, int label, int **images, int **segments, int *numOfSegments):*** פעולה זו מחזירה את כל האזורים בתמונות בעץ המתויגות בתיוג label.

נעבור על כל הצמתים בעץ בשיטת inorder, מעבר זה מתבצע בסיבוכיות זמן של O(k), כאשר k= גודל העץ (כמות התמונות). סריקה בשיטה זאת תיצור לנו את מערך התמונות ממוין כבר (לפי תכונות עץ בינארי ממוין) ולכן לא יהיה צורך במיון נוסף.

בכל צומת שנעבור, נסרוק את המערך המציג את האזורים המתויגים, ואם נמצא אזור עם התיוג המבוקש נכניס את התמונה למערך התמונות, ובאותו במקום במערך הסגמנטים נכניס את האזור שמצאנו. הסריקה של המערך מתבצעת בסיבוכיות זמן של O(n), כאשר segments=n (גודל מערך האזורים). עדכון מערכי הimages וsegments זה בכמות פעולות סופית ולכן סיבוכיות של O(1).

לכל תמונה בעץ סורקים מערך בגודל n, ולכן נקבל:

**<u>סיבוכיות זמן:</u>** O(n*k)

***void Quit(void **DS):*** פעולה זו משחררת את המבנה.

ראשית אנו עוברים על כל הצמתים בעץ ומשחררים להם את הdata (כלומר משחררים את הImage). בשחרור הImage אנו מוחקים את המערך המציג את התיוגים של האזורים, ואת הרשימה של האזורים הלא מתוייגים. שניהם באורך n ולכן השחרור של שניהם מתבצע בסיבוכיות זמן של O(n) כאשר n=segments.

המעבר על כל צמתי העץ לוקח סיבוכיות זמן של O(k) כאשר k= מספר התמונות במערכת. לכן נקבל:

**<u>סיבוכיות זמן:</u>** O(n*k)