

# חלק ראשון

Add:

עץ	רשימה	n	עץ	n רשימה	עץ	רשימה	n	
67165	5535	9050	64098	5510	4550	109112	25377	50
63471	8241	9150	64429	5604	4650	128429	19931	150
67251	5546	9250	63841	6878	4750	117629	18853	250
62689	6819	9350	64645	6464	4850	49059	18345	350
96642	5628	9450	65189	5671	4950	47070	18543	450
63882	6597	9550	64296	5597	5050	52602	24726	550
64017	6497	9650	65458	5797	5150	51428	34275	650
64493	5658	9750	63652	6945	5250	52500	21438	750
62682	5574	9850	65028	5557	5350	54133	19099	850
64523	6940	9950	64958	5574	5450	54182	18525	950
			65095	5611	5550	56001	18431	1050
			63830	5595	5650	54971	27363	1150
			64799	5685	5750	57045	27295	1250
			64508	6929	5850	55306	18657	1350
			63632	5581	5950	57612	19178	1450
			63608	6589	6050	56159	18251	1550
			64547	5564	6150	57787	18380	1650
			79624	5596	6250	58027	23291	1750
			61948	5596	6350	57787	31527	1850
			60809	6826	6450	58520	5575	1950
			62738	5582	6550	58746	5564	2050
			60683	5562	6650	62010	5533	2150
			62126	6756	6750	61016	5630	2250
			60228	5540	6850	61166	7198	2350
			64179	5495	6950	60879	6721	2450
			60608	6890	7050	63430	5660	2550
			64240	5542	7150	63193	5514	2650
			63023	6607	7250	61869	5630	2750
			62546	5624	7350	62377	5564	2850
			64948	5751	7450	62699	6940	2950
			61419	5583	7550	76652	5621	3050
			63177	6872	7650	61857	5551	3150
			61914	5692	7750	61703	5571	3250
			76184	5524	7850	62036	5522	3350
			78749	5699	7950	73940	5513	3450
			79617	5584	8050	60292	6890	3550
			64521	6927	8150	60139	6598	3650
			62239	5604	8250	60513	5635	3750
			62766	5565	8350	58827	5608	3850
			60889	6644	8450	60907	6906	3950
			63580	5611	8550	58434	5540	4050
			62614	5600	8650	65476	6976	4150
		74719	6950	8750	64198	5560	4250	
		65540	5631	8850	63048	5565	4350	
		69965	5534	8950	64562	5669	4450	

Find:

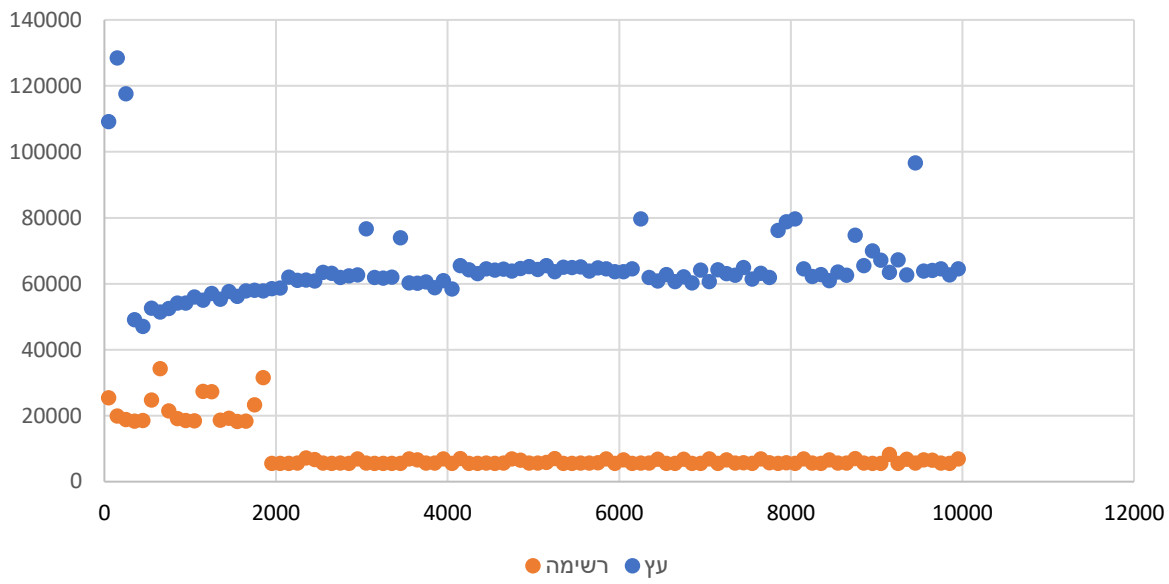
עץ	רשימה	n	עץ	רשימה	n	עץ	רשימה	n
9374	1831687	4600	11032	24768	2300	23887	85194	100
12658	19607	4600	8897	1070629	2400	23471	83696	100
9357	1837848	4700	10853	24818	2400	25636	145169	200
11891	19634	4700	8611	1111976	2500	23482	26067	200
9595	1871586	4800	10301	24833	2500	25172	118829	300
11161	19621	4800	8618	1162207	2600	26004	26059	300
9545	1884287	4900	10328	24701	2600	27962	164600	400
12069	19776	4900	8669	1233976	2700	25972	25878	400
9400	1968889	5000	10006	24797	2700	27190	210716	500
10764	19594	5000	8691	1277381	2800	25313	25871	500
9493	1899349	5100	10320	24682	2800	24837	272802	600
12336	18930	5100	8735	1313547	2900	26900	24815	600
9727	1929060	5200	10168	24697	2900	23055	304554	700
11887	19005	5200	8553	1406632	3000	27962	24746	700
9510	1964308	5300	10249	24797	3000	30068	345069	800
12336	19016	5300	11361	1409040	3100	28725	24690	800
9685	2002167	5400	10082	24810	3100	27938	389368	900
11455	18932	5400	11288	1517945	3200	26016	24707	900
9463	2046854	5500	10180	23688	3200	9393	433881	1000
12649	18985	5500	10787	1448699	3300	9176	24742	1000
9376	2090273	5600	10351	23637	3300	9209	479556	1100
12142	18981	5600	10433	1540610	3400	10211	24688	1100
9270	2141998	5700	10493	22771	3400	9020	524535	1200
11438	18358	5700	10619	1528528	3500	10669	24703	1200
9057	2118748	5800	9840	22864	3500	8638	570194	1300
11946	18977	5800	10158	1515629	3600	10223	24763	1300
9285	2178149	5900	10426	21084	3600	8638	615461	1400
11010	17780	5900	9893	1552719	3700	9966	26399	1400
9187	2113259	6000	10096	21867	3700	8284	708001	1500
12034	17844	6000	9838	1543769	3800	9639	25176	1500
9202	2148422	6100	10240	20394	3800	10655	738898	1600
11524	17810	6100	9619	1612768	3900	9618	25113	1600
15155	2138769	6200	10338	20345	3900	10769	753007	1700
14678	16762	6200	9657	2019082	4000	9827	24849	1700
12579	2106723	6300	10191	20740	4000	10050	822257	1800
12121	16741	6300	9506	1713192	4100	9518	24798	1800
11781	2145871	6400	10672	20367	4100	9790	895080	1900
10537	16883	6400	9300	1702352	4200	9690	25045	1900
11453	2180689	6500	13082	20330	4200	9589	893421	2000
12525	16302	6500	9431	1758202	4300	9518	24727	2000
11211	2200679	6600	11743	20298	4300	9221	931295	2100
11598	16274	6600	9364	1791102	4400	10914	24743	2100
11149	2234319	6700	12998	19742	4400	8919	976650	2200
12329	18214	6700	9451	1790979	4500	10732	24742	2200
11111	2259953	6800	11640	19614	4500	8796	1020842	2300

עץ	רשימה	n	עץ	רשימה	n
9664	2697112	9100	11435	16518	6800
12626	14229	9100	10977	2271956	6900
9660	2736649	9200	15859	15938	6900
13969	14250	9200	10757	2237941	7000
10100	2763663	9300	11284	16015	7000
14593	14263	9300	10622	2331551	7100
13796	2801450	9400	12272	16405	7100
16421	14250	9400	10478	2387595	7200
10047	2835543	9500	12485	16318	7200
12835	14342	9500	10498	2373261	7300
9899	2868636	9600	10559	15403	7300
12101	14233	9600	10460	2266454	7400
10583	2893774	9700	12155	14629	7400
13432	14189	9700	10276	2292155	7500
9920	2970470	9800	11099	14575	7500
13016	14274	9800	10326	2278528	7600
9583	2959761	9900	12216	14568	7600
13371	14317	9900	10041	2316319	7700
9728	3031929	10000	11311	14225	7700
13092	14300	10000	9661	2334845	7800
			11542	14310	7800
			9888	2360419	7900
			11031	14302	7900
			9856	2384089	8000
			10655	14602	8000
			9742	2482297	8100
			11722	14593	8100
			9708	2414118	8200
			12347	14282	8200
			9787	2432799	8300
			13725	14344	8300
			9547	2468806	8400
			13097	14204	8400
			9713	2533331	8500
			13663	14209	8500
			9505	2534437	8600
			12872	14208	8600
			9614	2558272	8700
			13530	14285	8700
			9704	2639605	8800
			13571	14267	8800
			10198	2636280	8900
			12812	14267	8900
			9683	2663784	9000
			13241	14212	9000

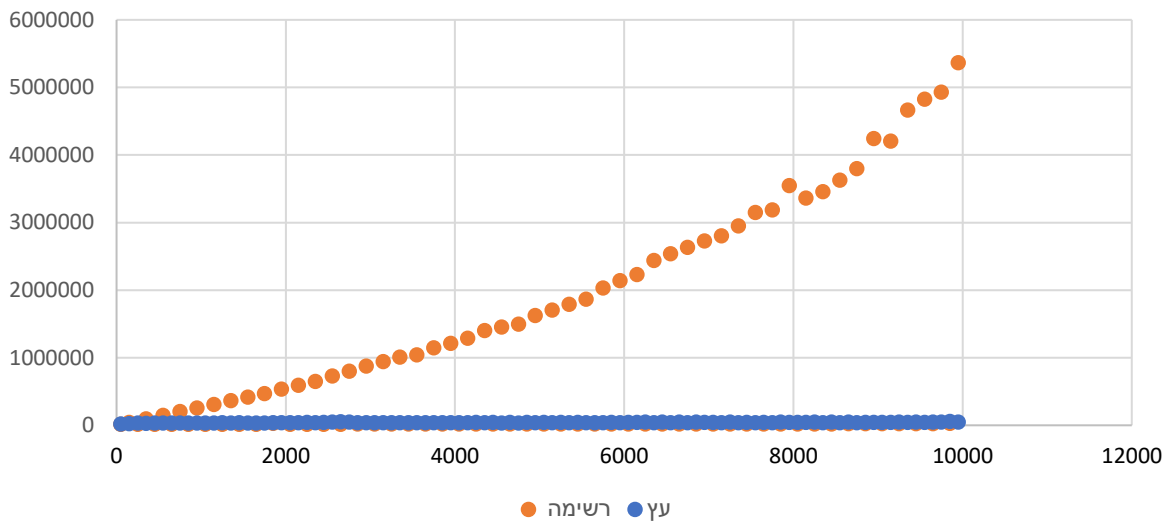
Delete:

עץ	רשימה	n	עץ	רשימה	n	עץ	רשימה	n
44275	22671	9050	34181	1452378	4550	21052	16014	50
41653	4204834	9150	42238	17568	4650	26226	41959	150
45496	24732	9250	35318	1499042	4750	31378	15692	250
40862	4664138	9350	41347	17717	4850	29414	93980	350
48783	25879	9450	37591	1622260	4950	32145	15699	450
42714	4826002	9550	40804	18118	5050	32128	149306	550
47763	26113	9650	36406	1705806	5150	33061	15777	650
47672	4929020	9750	42444	17964	5250	33335	202612	750
55791	27504	9850	36872	1788189	5350	33402	15713	850
46319	5366416	9950	43865	18020	5450	35105	256391	950
			37165	1865101	5550	34022	15762	1050
			40260	18110	5650	33462	309605	1150
			38087	2033231	5750	37454	15688	1250
			43511	19146	5850	34959	362574	1350
			40141	2140416	5950	35944	15697	1450
			43130	19233	6050	34755	416181	1550
			41393	2231388	6150	35462	15701	1650
			46494	19781	6250	35473	469530	1750
			40125	2438630	6350	37434	28773	1850
			47571	20435	6450	37553	536457	1950
			38903	2540882	6550	37274	16083	2050
			46234	20511	6650	36192	591205	2150
			39101	2635594	6750	40993	16092	2250
			46738	20705	6850	37669	647808	2350
			41327	2728364	6950	42706	16111	2450
			44294	20436	7050	46589	728404	2550
			37435	2806165	7150	50154	16765	2650
			46416	20590	7250	46488	798913	2750
			36629	2950874	7350	36469	17032	2850
			44483	21385	7450	37856	875964	2950
			38751	3147792	7550	37242	17482	3050
			42711	21201	7650	35903	940874	3150
			38198	3185188	7750	40237	17629	3250
			45345	21178	7850	36967	1010918	3350
			41220	3548207	7950	38466	17037	3450
			43433	21119	8050	36827	1041509	3550
			41911	3364859	8150	38222	17182	3650
			45734	21192	8250	37323	1148524	3750
			40331	3456752	8350	38821	17507	3850
			47177	21299	8450	38546	1214464	3950
			40362	3628264	8550	38849	17524	4050
			45813	22004	8650	40034	1288821	4150
			38602	3798074	8750	44001	17708	4250
			44376	22100	8850	36501	1399721	4350
			40733	4242304	8950	43320	17635	4450

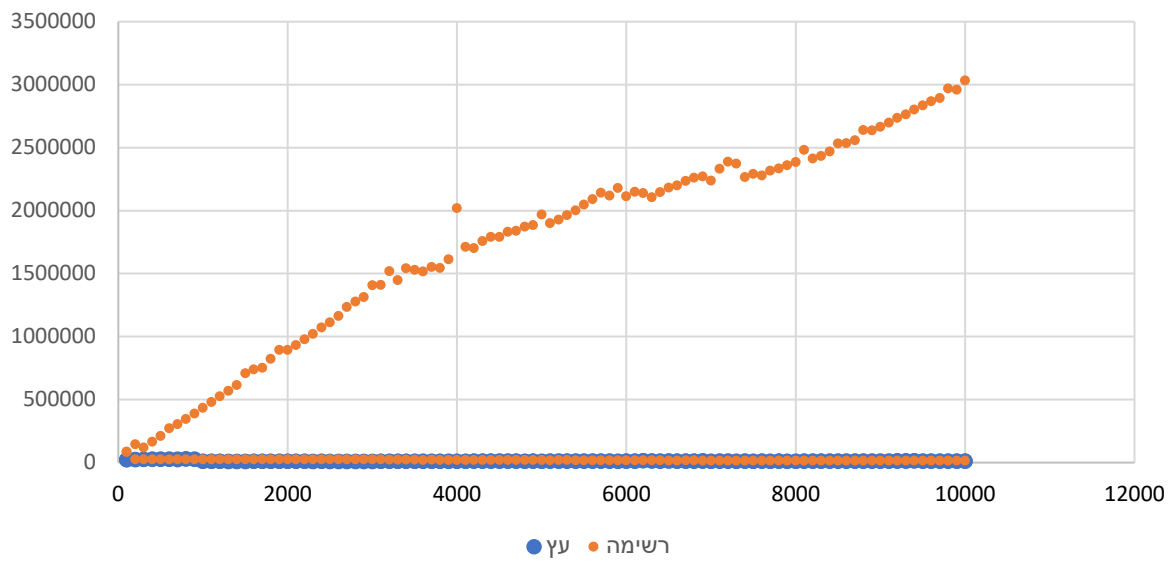
### Add



### Delete



# Find



## חלק שני:

### 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 זו 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.

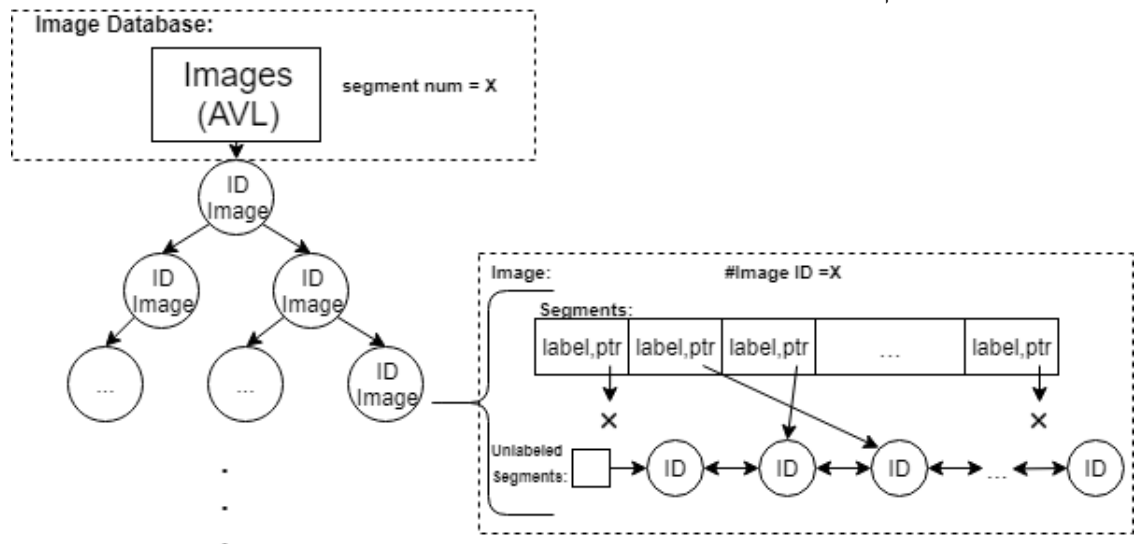
המערך באורך זהה למספר הסגמנטים, כאשר כל תא במערך תואם באינדקס שלו למספר הסגמנט שהוא מייצג.

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

- עבור סגמנט מתיוג: התא המתאים במערך יכול את הערך המספרי של התיוג ומצביע ל-NULL.

- עבור סגמנט חסר תיוג: התא המתאים במערך יכול 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 ישירה 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). ההעתיקה 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* והעתיקת 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, וסיוור *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  $O(k \cdot n) = O(k + k \cdot n + k \cdot n + k)$ .

: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.