תרגיל בית רטוב 1

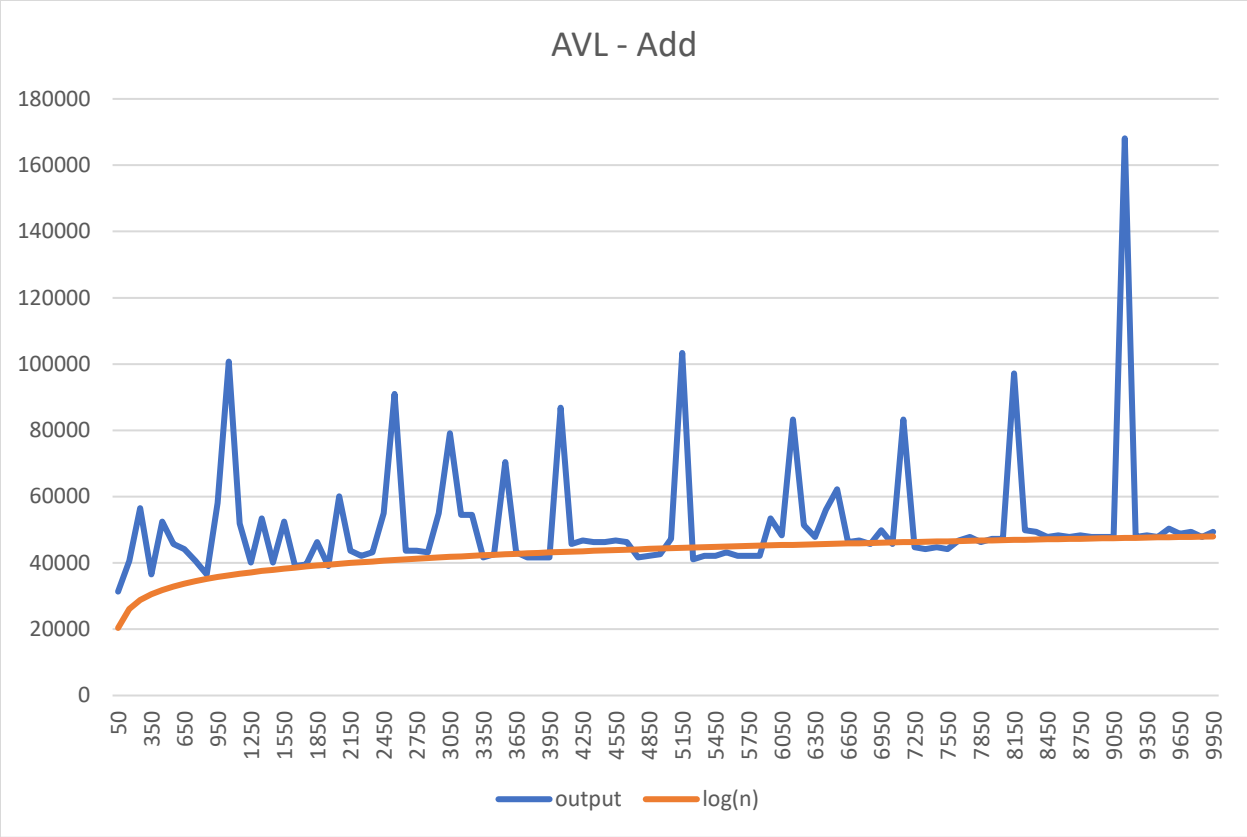
חלק יבש 1

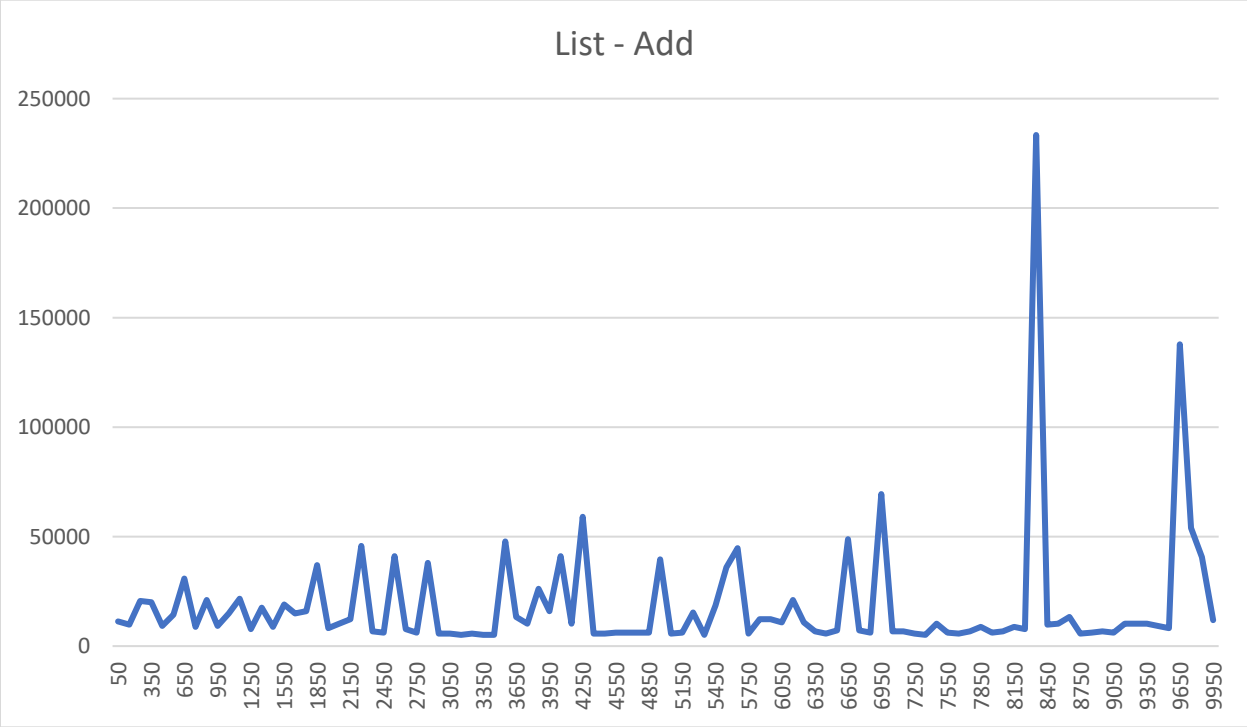
ADD:

n	רשימה מקושרת	עץ חיפוש מאוזן
50	11309	31358
150	9767	40611
250	20562	56546
350	20049	36498
450	9253	52434
550	14393	45751
650	30843	44208
750	8739	40610
850	21076	36498
950	9253	58088
1050	14908	100755
1150	21590	51919
1250	7710	40096
1350	17478	53462
1450	8739	40097
1550	19020	52434
1650	14907	39069
1750	15936	39582
1850	37012	46265
1950	8225	39068
2050	10281	60145
2150	12337	43694
2250	45751	42153
2350	6683	43181
2450	6169	55004
2550	41124	90988
2650	7711	43695
2750	6168	43695
2850	38040	43181
2950	5655	55005
3050	5655	79165
3150	5141	54490
3250	5655	54490
3350	5140	41638
3450	5140	42667
3550	47807	70426

43181	13365	3650
41639	10281	3750
41639	26217	3850
41638	15936	3950
86876	41124	4050
45751	10281	4150
46779	59117	4250
46265	5654	4350
46265	5655	4450
46779	6168	4550
46265	6168	4650
41639	6169	4750
42153	6168	4850
42666	39583	4950
47293	5655	5050
103325	6169	5150
41124	15422	5250
42153	5140	5350
42153	18506	5450
43181	35983	5550
42152	44723	5650
42153	5655	5750
42152	12337	5850
53462	12338	5950
48321	10795	6050
83278	21076	6150
51405	10795	6250
47807	6683	6350
56033	5654	6450
62201	7197	6550
46265	48835	6650
46780	7197	6750
45751	6168	6850
49863	69398	6950
45751	6683	7050
83277	6682	7150
44723	5654	7250
44209	5140	7350
44723	10281	7450
44209	6169	7550
46779	5655	7650
47808	6683	7750
46266	8739	7850

47293	6169	7950
47293	6682	8050
97156	8739	8150
49864	7711	8250
49349	233381	8350
47807	9768	8450
48321	10281	8550
47807	13366	8650
48321	5654	8750
47807	6169	8850
47807	6683	8950
47808	6169	9050
168097	10281	9150
47807	10281	9250
48321	10281	9350
47807	9253	9450
50377	8225	9550
48835	137767	9650
49350	53976	9750
47808	40611	9850
49349	11824	9950


AVL - Add
180000
160000
140000
120000
100000
80000
60000
40000
20000
0
50
350
650
950
1250
1550
1850
2150
2450
2750
3050
3350
3650
3950
4250
4550
4850
5150
5450
5750
6050
6350
6650
6950
7250
7550
7850
8150
8450
8750
9050
9350
9650
9950
output
log(n)


List - Add
250000
200000
150000
100000
50000
0
50
350
650
950
1250
1550
1850
2150
2450
2750
3050
3350
3650
3950
4250
4550
4850
5150
5450
5750
6050
6350
6650
6950
7250
7550
7850
8150
8450
8750
9050
9350
9650
9950

:FIND

n	רשימה מקושרת	עץ חיפוש מאוזן
100	15936	5141
100	16450	5140
200	61686	5654
200	17478	5140
300	104353	7197
300	22618	6169
400	99727	6683
400	17478	7197
500	128000	21076
500	16964	7196
600	162441	6683
600	16964	6683
700	228755	5655
700	16450	6683
800	317173	6682
800	24161	6683
900	452369	6169
900	50378	5654
1000	447229	6168
1000	17992	6168
1100	432835	5655
1100	17478	6682
1200	486811	5654
1200	22618	6682
1300	552610	6169
1300	15935	11309
1400	1144288	7711
1400	20562	11309
1500	673413	6169
1500	16964	6683
1600	703743	6682
1600	15936	7197
1700	715565	7711
1700	16964	6683
1800	720706	7196
1800	16964	6169
1900	1121670	7197
1900	34442	6682
2000	766457	6683
2000	15936	6683

6683	795244	2100
26217	15935	2100
9253	1232192	2200
7710	20048	2200
6683	1003437	2300
7197	19020	2300
7711	2078842	2400
7197	17992	2400
6683	1051244	2500
6682	16450	2500
6683	1002923	2600
7711	15422	2600
6169	1021943	2700
6683	15936	2700
6169	1066152	2800
7711	15936	2800
9253	1198264	2900
9253	15936	2900
9767	1149943	3000
8739	15936	3000
11824	1199807	3100
7711	17478	3100
10281	1217799	3200
7711	15936	3200
8225	1250698	3300
7197	15936	3300
7711	1326264	3400
7711	16450	3400
7711	1585348	3500
9253	15936	3500
8739	1383325	3600
7197	15935	3600
10795	1409027	3700
7710	15422	3700
8225	1445525	3800
7711	15936	3800
7711	1480481	3900
7197	15421	3900
6682	1680449	4000
7197	15422	4000
6169	1563244	4100
7711	15935	4100
8225	1592545	4200

8225	15936	4200
7711	1631099	4300
9254	15421	4300
5655	1670167	4400
6683	15936	4400
6168	1823870	4500
6682	15422	4500
7711	1747790	4600
7196	15936	4600
6683	1928737	4700
7197	15935	4700
7197	1892240	4800
7196	15936	4800
7197	2014071	4900
7710	15936	4900
6682	1938505	5000
7711	15422	5000
6683	1987340	5100
7711	28273	5100
6682	2062906	5200
8739	15422	5200
6683	2293204	5300
7197	15421	5300
6683	2024352	5400
7197	14908	5400
6169	2363115	5500
9253	15936	5500
7197	2136416	5600
7197	14908	5600
7196	2484946	5700
7711	15936	5700
7197	2971242	5800
6683	17992	5800
6169	4013748	5900
25703	26731	5900
6168	3191773	6000
6683	18506	6000
6682	2943998	6100
6683	17478	6100
7711	3278648	6200
7197	16450	6200
7711	4234278	6300
7197	23646	6300

7711	3227242	6400
7196	15936	6400
7711	3085363	6500
7711	22619	6500
7711	5866919	6600
7197	16450	6600
7711	3508431	6700
6682	20562	6700
8225	5023354	6800
9253	17992	6800
8739	3572688	6900
7711	21076	6900
8225	6725392	7000
8739	21077	7000
8739	3633860	7100
8739	20048	7100
8225	5994405	7200
8739	23132	7200
8225	3632833	7300
7711	77622	7300
8225	4092912	7400
8225	16449	7400
8225	3927386	7500
12338	29815	7500
8225	4056929	7600
8225	15936	7600
8225	3595306	7700
8225	15935	7700
8225	4118101	7800
8225	15936	7800
8225	3732559	7900
8739	16450	7900
8739	4226567	8000
8225	15936	8000
8225	3820463	8100
7711	15936	8100
7711	3963884	8200
9253	25189	8200
8225	3981362	8300
8739	15935	8300
8739	4369474	8400
9253	16450	8400
8225	4526261	8500

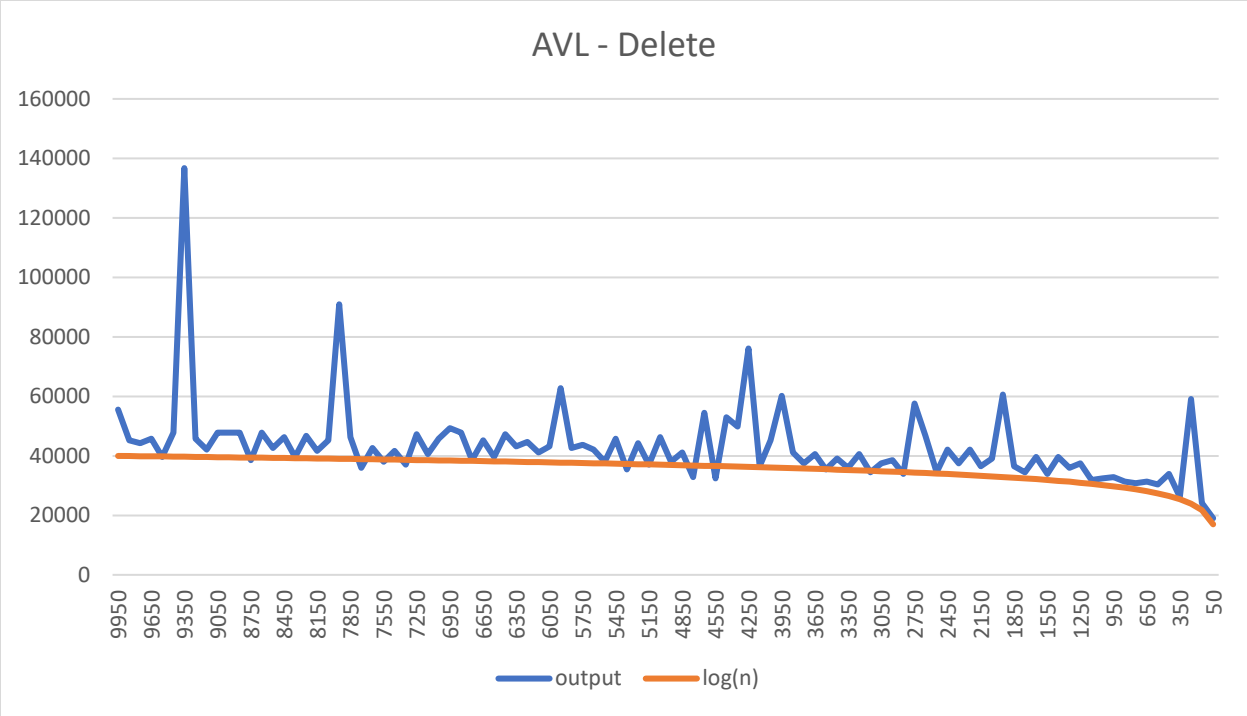
9253	15936	8500
8225	4475370	8600
8739	15936	8600
8225	4734968	8700
8739	17478	8700
8225	5064478	8800
8739	19534	8800
8225	5248510	8900
8739	21076	8900
8225	5139530	9000
8739	16450	9000
8225	7945761	9100
15936	25703	9100
10795	5349779	9200
10281	21076	9200
9253	5254165	9300
11310	18506	9300
12337	5689056	9400
9253	17992	9400
7711	5135932	9500
8225	17991	9500
7711	8478837	9600
8225	16449	9600
9253	6620525	9700
8739	17992	9700
8225	6905826	9800
8739	15935	9800
8225	6752638	9900
8225	15936	9900
8225	5080928	10000
9253	15936	10000

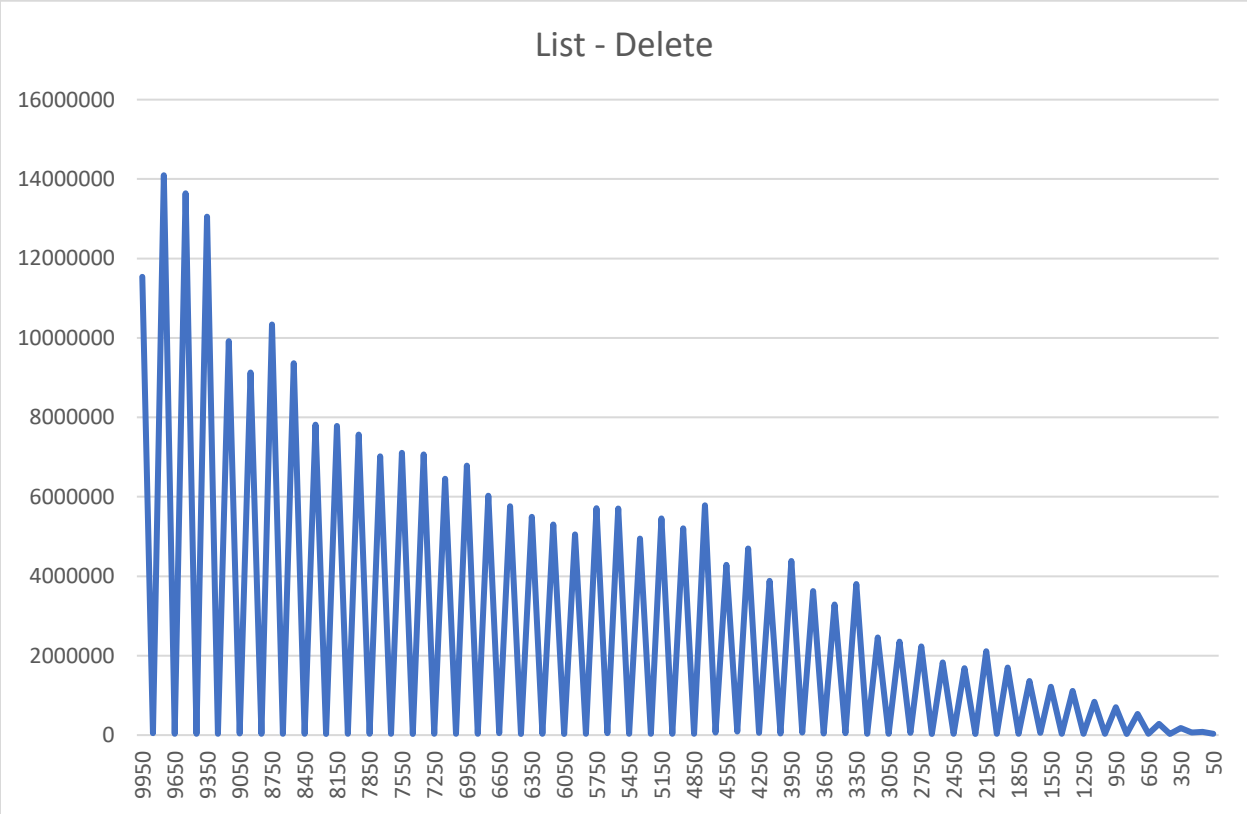
DELETE:

n	רשימה מקושרת	עץ חיפוש מאוזן
9950	11540554	55518
9850	50891	45237
9750	14095411	44209
9650	35470	45751
9550	13642527	39582
9450	34442	47808
9350	13049307	136739
9250	33414	45751
9150	9920764	42152
9050	37526	47807
8950	9128090	47807
8850	32386	47807
8750	10335606	38555
8650	32385	47807
8550	9366612	42666
8450	29816	46265
8350	7817761	39583
8250	29301	46779
8150	7784347	41638
8050	30329	45237
7950	7564846	90988
7850	29302	46265
7750	7014806	35984
7650	29815	42667
7550	7103738	38040
7450	29301	41638
7350	7069296	37012
7250	29815	47293
7150	6454999	40611
7050	29815	45752
6950	6782967	49349
6850	29815	47808
6750	6029875	38554
6650	45751	45237
6550	5757426	39582
6450	29301	47294
6350	5498856	43181
6250	33413	44723
6150	5299402	41125
6050	29301	43181

62715	5055739	5950
42667	33928	5850
43695	5715273	5750
42153	46265	5650
38040	5705505	5550
45751	35470	5450
35470	4946246	5350
44209	35470	5250
37012	5458759	5150
46265	35470	5050
38040	5209441	4950
41124	32385	4850
32899	5787240	4750
54490	61687	4650
32385	4287226	4550
52948	85847	4450
49864	4696928	4350
76080	60145	4250
36498	3883178	4150
45237	41124	4050
60144	4380270	3950
41125	63229	3850
37526	3630262	3750
40610	38041	3650
35469	3287901	3550
39069	51920	3450
35984	3804527	3350
40611	29815	3250
34441	2460271	3150
37526	32386	3050
38554	2355404	2950
33928	58603	2850
57574	2229974	2750
46779	29301	2650
34441	1828497	2550
42153	29816	2450
37526	1686617	2350
42153	29301	2250
36498	2113798	2150
39069	31357	2050
60658	1697413	1950
36498	32385	1850
34442	1365333	1750

39583	56032	1650
33928	1214200	1550
39583	31871	1450
35984	1116015	1350
37526	29815	1250
31871	834827	1150
32386	29815	1050
32899	698088	950
31357	29301	850
30843	533076	750
31357	29815	650
30329	285301	550
33928	29301	450
26217	176321	350
59117	66827	250
24160	78651	150
19020	29815	50


AVL - Delete
160000
140000
120000
100000
80000
60000
40000
20000
0
9950
9650
9350
9050
8750
8450
8150
7850
7550
7250
6950
6650
6350
6050
5750
5450
5150
4850
4550
4250
3950
3650
3350
3050
2750
2450
2150
1850
1550
1250
950
650
350
50
output
log(n)

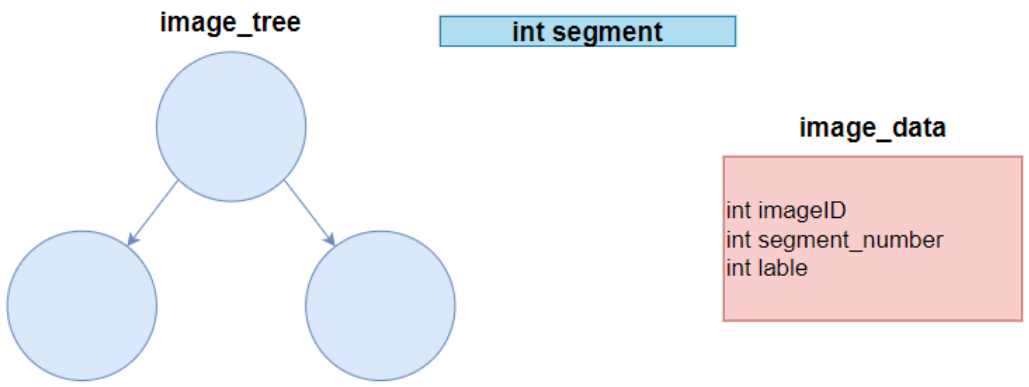

List - Delete
16000000
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
9950
9650
9350
9050
8750
8450
8150
7850
7550
7250
6950
6650
6350
6050
5750
5450
5150
4850
4550
4250
3950
3650
3350
3050
2750
2450
2150
1850
1550
1250
950
650
350
50

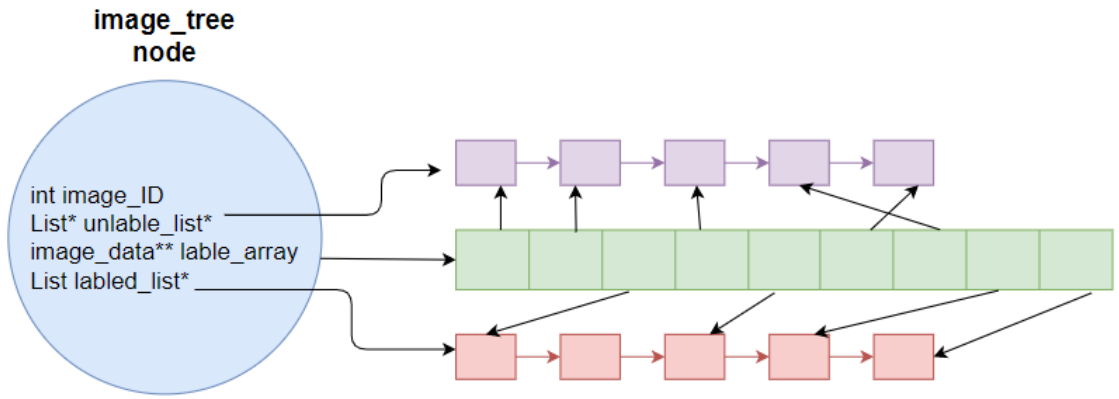
תרגיל בית רטוב 1

חלק יבש 2

תיאור מבנה הנתונים

נשמור את הsegment במשתנה במבנה. וניצור עץ חיפוש בינארי שהמפתחות שלו יהיו הimageID. בכל צומת בעץ נשמור את הimageID, מערך בגודל segment, ושתי רשימות מקושרות, אחת לכל האזורים הלא מתויגים והשנייה לאזורים המתוייגים. הרשימות הן רשימות של image_data, משתנה ששומר את מזהה התמונה, הסגמנט והתיוג של התמונה בסגמנט הנ"ל.





---

void * Init(int segments):

1. ניצור עץ ריק שישמור את הנתונים של התמונות ונשמור אליו מצביע
2. במקרה ויש כישלון ביצירת העץ נחזיר NULL
3. נשמור את הsegment בתוך משתנה שנקרא לו segment

סיבוכיות מקום O(1) , סיבוכיות זמן O(1)

StatusType AddImage(void *DS, int imageID):

1. אם DS הוא NULL נחזיר INVALID_INPUT
2. אם IMAGEID הוא קטן או שווה ל0, נחזיר INPUT_INVALID
3. נבדוק בעץ התמונות אם קיימת צומת בעלת מפתח imageID
   - אם כן, נחזיר FAILURE ונסיים.
   - אם לא, נמשיך
4. נוסיף לעץ התמונות צומת בעלת מפתח imageID
5. נאתחל את השדה image_ID של הצומת הזו להיות imageID
6. ניצור מערך lables בגודל segment , מערך מסוג מצביעים ל- image_data , אליו יצביע השדה lable_array בצומת imageID
7. ניצור רשימה חדשה , אליה יצביע השדה unlable_list
8. נעבור על מערך הlable, ועבור כל אינדקס במערך נבצע את הפעולות הבאות:
   - נוסיף לרשימה unlable_list איבר חדש,
     i. בו נכתוב בשדה imageID את הimageID שלנו,
     ii. בשדה segment_number את האינדקס.
     iii. בשדה lable נשמור -1
   - נשמור במערך ה lable במקום של האינדקס הנוכחי מצביע לאיבר החדש שהוספנו עכשיו לרשימה
   - כך אתחלנו את המערך כולו להצביע לרשימת ה"לא מתוייגים" , ועדכנו את רשימת ה"לא מתויגים" להיות רשימה שמכילה את כל הסגמנטים.
9. במידה ויש בעיה בהקצאת הזיכרון בכל אחד מהשלבים נחזיר ALLOCATION_ERROR
10. נחזיר SUCCESS

<u>סיבוכיות:</u>

חיפוש צומת בעלת מפתח imageID-סיבוכיות זמן O(log(k))

הוספת צומת בעלת מפתח imageID – סיבוכיות זמן O(log(k))

יצירת מערך בגודל n - סיבוכיות מקום n

מעבר על המערך lable ועבור כל איבר במערך נבצע מספר פעולות קבועות (הוספת איבר לרשימה ושמירת מצביע במקום הi במערך לאיבר זה) -סיבוכיות זמן O(n)

סה"כ סיבוכיות זמן O(log(k)+n) כנדרש

StatusType DeleteImage(void *DS, int imageID):

1. אם DS הוא NULL נחזיר INVALID_INPUT
2. אם imageID הוא קטן או שווה ל0 נחזיר INVALID_INPUT
3. //
   - אם לא, נחזיר FAILURE
   - אם כן, נמשיך.
4. ניגש למערך הlabel ונעבור על כולו, כל עוד האינקס i קטן מגודל המערך ועבור כל i נבצע את הפעולות הבאות:
   - ניגש לאיבר image_data אליו המערך במקום ה-i מצביע.
   - נבדוק האם השדה lable שווה ל-1.
   - אם כן, נסיר את האיבר עליו אנו מצביעים מהרשימה unlable_list ונמשיך ל i+1

- אם לא, נסיר את האיבר עליו אנו מצביעים מהרשימה labled_list ונמשיך ל i+1
- נשחרר את הפוינטר

5. נשחרר את הזיכרון של המערך אליו מצביע lable_array ואת הזיכרון של הפוינטר המצביע אליו
6. נשחרר את הזיכרון של הרשימה המקושרת unlable_list (נמחק אותה) וכמובן שגם את הזיכרון של המצביע עליה
7. נשחרר את הזיכרון של הרשימה המקושרת labled_list (נמחק אותה) וגם נשחרר את הזיכרון של המצביע עליה
8. נמחק את הצומת imageID מהעץ
9. אם בשלב כלשהו נתקלנו בבעיה בהקצאת זיכרון נחזיר ALLOCATION_ERROR
10. נחזיר SUCCESS

<u>סיבוכיות:</u>

נבדוק האם קיים מפתח imageID- סיבוכיות זמן O(log(k))
מעבר על כל המערך וביצוע מספר קבוע של פעולות קבועות עבור כל תא במערך, סיבוכיות הזמן היא O(n)
כאשר n הוא גודל המערך (מספר התיוגים האפשריים)
מחיקת הצומת מהעץ היא גם בסיבוכיות של O(log(k)),
ולכן הסיבוכיות של הפונקציה הזו היא O(log(k)+n) כנדרש.

<u>StatusType AddLabel(void *DS, int imageID, int segmentID, int label):</u>

1. אם DS הוא NULL או segmentID >=segments או segmentID<0 נחזיר INVALID_INPUT
2. אם imageID הוא קטן או שווה ל0 , או lable קטן או שווה ל0 נחזיר INVALID_INPUT
3. נבדוק האם imageID הוא מפתח של צומת שקיימת בעץ.
   - אם לא, נחזיר FAILURE
   - אם כן, נשמור אליה מצביע ונמשיך
4. ניגש אל הצומת המייצגת את התמונה בעץ, וניגש למערך lable_array , למקום segmentID, וניגש ל image_data אליו הוא מצביע.
   - אם הlable שונה מ 1- נחזיר FALIURE .
   - אם הוא שווה ל 1- נמשיך
5. נמחק את האיבר image_data מהרשימה unlable_list
6. נוסיף איבר image_data חדש לרשימה labled_list , בוא נכתוב בשדה imageID את הimageID, בsegment_number את הsegmentID, וב lable את הlable
7. נשמור במערך lable_array באינדקס segment מצביע לאיבר שהוספנו בlabled_list
8. אם בשלב כלשהו נתקלנו בבעיה בהקצאת זיכרון נחזיר ALLOCATION_ERROR
9. נחזיר SUCCESS

סיבוכיות:
חיפוש הצומת imageID בעץ היא בסיבוכיות של O(log(k)) , גישה לאיבר במערך , מחיקה והוספה של איברים ברשימות הן פעולות אותם אנו מבצעים מספר קבוע של פעמים, ולהן סיבוכיות של O(1), לכן סיבוכיות הזמן של פונקציה זו היא O(log(k)) שפרט קטנה מ O(log(k)+n) ולכן הסיבוכיות היא כנדרש.

StatusType GetLabel(void *DS, int imageID, int segmentID, int *label):

1. אם DS הוא NULL או segmentID >=segments או segmentID<0 נחזיר INVALID_INPUT
2. אם imageID הוא קטן או שווה ל0 , או lable הוא NULL אז נחזיר INVALID_INPUT
3. נבדוק האם imageID הוא מפתח של צומת שקיימת בעץ.
   - אם לא, נחזיר FAILURE
   - אם כן, נשמור אליה מצביע ונמשיך
4. ניגש אל הצומת המייצגת את התמונה בעץ, וניגש למערך lable_array , למקום segmentID, וניגש ל image_data אליו הוא מצביע.
   - אם ה lable שווה ל1- נחזיר FAILURE
   - נשמור את lable בתור המצביע lable שקיבלנו
5. אם בשלב כלשהו נתקלנו בבעיה בהקצאת זיכרון נחזיר ALLOCATION_ERROR
6. נחזיר SUCCESS

סיבוכיות:

נמצא את התמונה בעץ בסיבוכיות של O(log(k)), ניגש למערך בO(1), וממנו לimage_data ב O(1), ושם נבדוק את הlable בO(1) , ולכן הסיבוכיות של הפונקציה הזו היא O(log(k)) כנדרש.

StatusType DeleteLabel(void *DS, int imageID, int segmentID):

1. אם DS הוא NULL או segmentID >=segments או segmentID<0 נחזיר INVALID_INPUT
2. אם imageID הוא קטן או שווה ל 0 נחזיר INVALID_INPUT
3. נבדוק האם imageID הוא מפתח של צומת שקיימת בעץ.
   - אם לא, נחזיר FAILURE
   - אם כן, נשמור אליה מצביע ונמשיך
4. ניגש אל הצומת המייצגת את התמונה בעץ, וניגש למערך lable_array , למקום segmentID, וניגש ל image_data אליו הוא מצביע.
   - אם ה lable שווה ל 1- , נחזיר FAILURE
   - אחרת , נמשיך
5. נמחק מהרשימה lables_list את האיבר image_data אליו אנו מצביעים.
6. נוסיף לרשימה unlable_list איבר imgae_data חדש בו נאתחל את השדות להיות – imageID ל imageID segment_number לsegmentID , וlable להיות 1- .
7. אם בשלב כלשהו נתקלנו בבעיה בהקצאת זיכרון נחזיר ALLOCATION_ERROR
8. נחזיר SUCCESS

סיבוכיות:

חיפוש הצומת בעץ היא בסיבוכיות של O(log(k)), גישות לרשימות השונות ומחיקת האיברים בהן הן בסיבוכיות של O(1), לכן אנו מבצעים מספר קבוע של פעולות קבועות ועוד פעולה בסיבוכיות של O(log(k)) , ולכן הסיבוכיות של הפונקציה היא O(log(k)) כנדרש.

StatusType GetAllUnLabeledSegments(void *DS, int imageID, int **segments, int* numOfSegments):

1. אם DS הוא NULL , imageID<=0 , segments==NULL או numOfSegments==NULL נחזיר INVALID_INPUT
2. נבדוק האם imageID הוא מפתח של צומת שקיימת בעץ.
   - אם לא, נחזיר FAILURE
   - אם כן, נשמור אליה מצביע ונמשיך
3. ניגש לרשימה unlable_list, ונבדוק האם היא ריקה.
   - אם כן, נחזיר FAILURE
   - אחרת, נמשיך.
4. נשמור בnumOfSegments את מספר האיברים ברשימה unlable_list
5. ניצור מערך בגודל שקיבלנו בסעיף 4.
6. נעבור על הרשימה unlable_list ועבור כל איבר ברשימה, נשמור במערך segments את segment_numberה ששמור באיבר הספציפי ברשימה עליו אנו עוברים.
7. נעדכן את המשתנה segments שקיבלנו להצביע על המערך החדש והמעודכן שיצרנו.
8. אם בשלב כלשהו נתקלנו בבעיה בהקצאת זיכרון נחזיר ALLOCATION_ERROR
9. נחזיר SUCCESS

** אנחנו צריכים להקצות את המערך עם malloc ו free ...

סיבוכיות:

נחפש את התמונה בעץ בסיבוכיות של O(log(k)) נעבור על הרשימה שמכילה s איברים אשר הם האזורים הלא מתויגים בתמונה ובכל מעבר כזה נבצע מספר קבוע של פעולות קבועות. לכן סיבוכיות הזמן של הפונקציה הזו היא O(log(k)+s) כנדרש.

StatusType GetAllSegmentsByLabel(void *DS, int label, int **images, int **segments, int *numOfSegments):

1. אם DS==NULL או images==NULL או segmenst==NULL או numOfSegments==NULL , או ש lable<=0 , נחזיר INVALID_INPUT.
2. ניצור משתנה counter ונאתחל אותו ל0
3. נעבור על העץ image_tree באמצעות inorder, עבור כל צומת בעץ ניגש למערך lable_array ונבצע את הפעולות הבאות:
   - ❖ נעבור על המערך ועבור כל i במערך :
     (1) נבדוק האם הlable ב image_data אליו הוא מצביע שווה לlable
     - אם לא נמשיך
     - אם כן, נעלה את ה counter ב 1.
4. עכשיו שאנו יודעים מה הגודל הנדרש למערכים, נשמור בnumOfSegments את counter, וניצור שני מערכים images ו segments בגודל counter
5. נעבור שוב על העץ image_tree באמצעות inorder , ובכל צומת בעץ ניגש למערך lable_array, ונעבור עליו, ועבור כל אינדקס i נבצע את הפעולות הבאות:
   i. נבדוק האם ה lable ב image_data אליו הוא מצביע שווה ל lable
   ii. אם לא נמשיך

iii. אם כן ,נשמור את הimageID של הצומת הזו במערך images במקום counter ואת האינדקס i , שהוא הסגמנט, בערך segments במקום ה counter.

iv. נעלה את ה counter ב 1 ונמשיך.

6. לאחר שסיימנו לעבור על כל העץ ולמלא את המערכים, נעדכן את המצביעים שקיבלנו להצביע למערכים שלנו בהתאמה .
7. אם בשלב כלשהו נתקלנו בבעיה בהקצאת זיכרון נחזיר ALLOCATION_ERROR
8. נחזיר SUCCESS

סיבוכיות:

נעבור על העץ בסיבוכיות של n, ועבור כל צומת נעבור על כל הסגמנטים (k) ועבורכל אחד מהם נבצע מספר קבוע של פעולות קבועות. הסיבוכיות של פעולה זו היא O(n·k) אנו מבצעים פעולה זו פעמיים, ולכן סה"כ סיבוכיות הזמן של פונקציה זו היא O (n·k) כנדרש.

<u>void Quit(void **DS)</u> :

1. אם DS==NULL נחזיר INVALID_INPUT
2. נעבור על העץ image_tree ב inoreder , עבור כל צומת נבצע את הפעולות הבאות:
   (1) נעבור על המערך lable_array,
   (2) בכל אינדקס i במערך נבדוק האם lable בimage_data אליו הוא מצביע שווה ל-1
   (3) אם כן, נסיר את האיבר הזה מהרשימה unlable_list, ואז נשחרר את המצביע
   (4) אם לא , נסיר את האיבר מהרשימה lable_list , ואז נשחרר את המצביע
   (5) נמחק את הרשימה lable_list
   (6) נמחק את הרשימה unlable_list
   (7) נמחק את המערך
   (8) נמחק את כל המצביעים בצומת
   (9) נסיר את הצומת
3. אם בשלב כלשהו נתקלנו בבעיה בהקצאת זיכרון נחזיר ALLOCATION_ERROR
4. נחזיר SUCCESS

סיבוכיות:

נעבור על העץ ב O(n) , ועבור כל עץ נעבור על המערך באורך segments (k) ועבור כל איבר במערך נבצע מספר קבוע של פעולות, ובנוסף לכך נבצע עוד מספר קבוע של פעולות, ולכן סה"כ הסיבוכיות היא O (n·k) כנדרש.