דו"ח מיני פרוייקט – אנליזה של מידע

שמות המגישים:

רן עזרא, ת.ז 302779657

נדב חנמ, ת.ז 313260648

שאלה 1

בגרסת המימוש השנייה, בחרנו לבדוק עבור כל פיקסל נתון לשכלל גם את הפיקסלים שבסביבתו.

לדוג', עבור פיקסל 324, נרצה לחשב בנוסף לערכו גם את ערכם של הפיקסלים הבאים:

295	296	297
323	324	325
351	352	353

לצורך כך, נדרשנו לחשב את העמודה והשורה שמס' פיקסל נתון מייצג, ולחשב את מספרי הפיקסלים של העמודות והשורות הסמוכים.

כמובן שנדרשנו לחישוב ספציפי עבור מקרים שונים בהם פיקסל נתון הוא בשולי הדף.

לדוג' עבור פיקסל 5, קיימים 5 תאים סמוכים בלבד, שכן לא קיימת שורה מעליו:

4	5	6
32	33	34

אל מול רעיון מרכזי זה, ניסינו מגוון וריאציות שונות, ע"מ לקבל את הפתרון האפקטיבי ביותר.

במהלך גיבוש הרעיון, ניסינו לבחון את השפעת התנאים הבאים על אחוז השגיאה:

(1) <u>שיטת החישוב:</u>

א. <u>Average</u> – בשיטה זו, מתבצע חישוב ממוצע הערכים של סביבת פיקסל נתון ובדיקה האם הערך שמתקבל גדול ממס' קבוע כלשהו – "MinAvg"

ב. <u>Counter</u> - בשיטה זו, מגדירים מס' מינימום "MinCount", כך שהתנאי מתקיים עבור כל פיקסל שבסביבתו יש לכל הפחות "MinCount" פיקסלים הגדולים מ – 128.

ההבדל בין השיטות מתבטא במשקל שמקבלים ערכים "גבוליים" של פיקסלים נתונים. כלומר, פיקסלים שערכם קרוב יחסית לערך 128 משפיעים באופן שונה בכל אחת מהשיטות.

לדוג', נוכל להתבונן בסביבת הפיקסל הבאה:

120	**120**	**120**
120	**255**	**120**
120	**120**	**120**

לפי שיטה א', עבור MinAvg = 128,

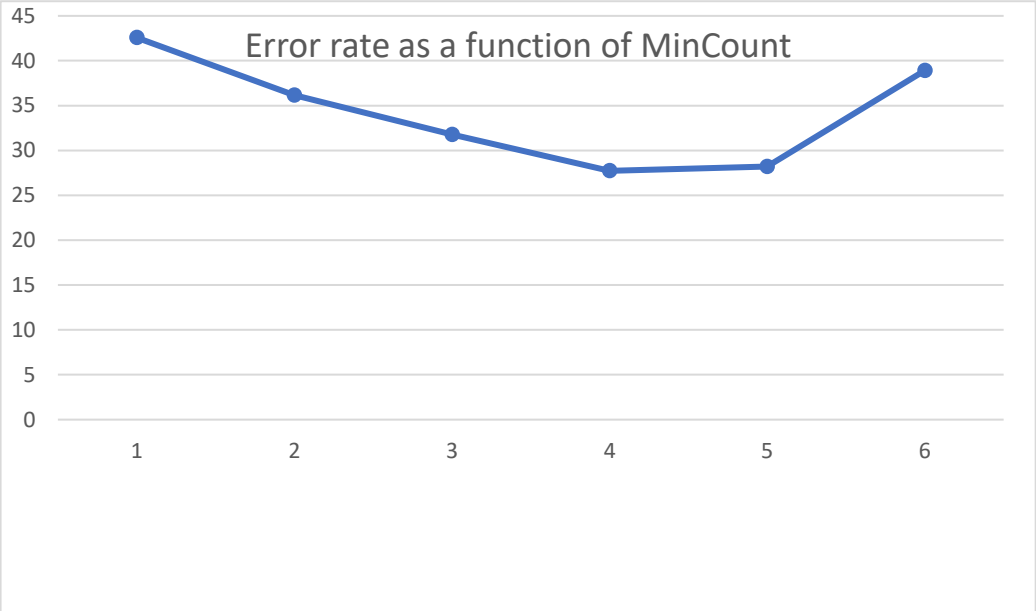
הערך שיתקבל עבור ממוצע סביבת הפיקסל הנתון יהיה $\frac{(8*120)+255}{9} = 135$.

כלומר, **סביבת הפיקסל תעמוד בשיטה א'.**

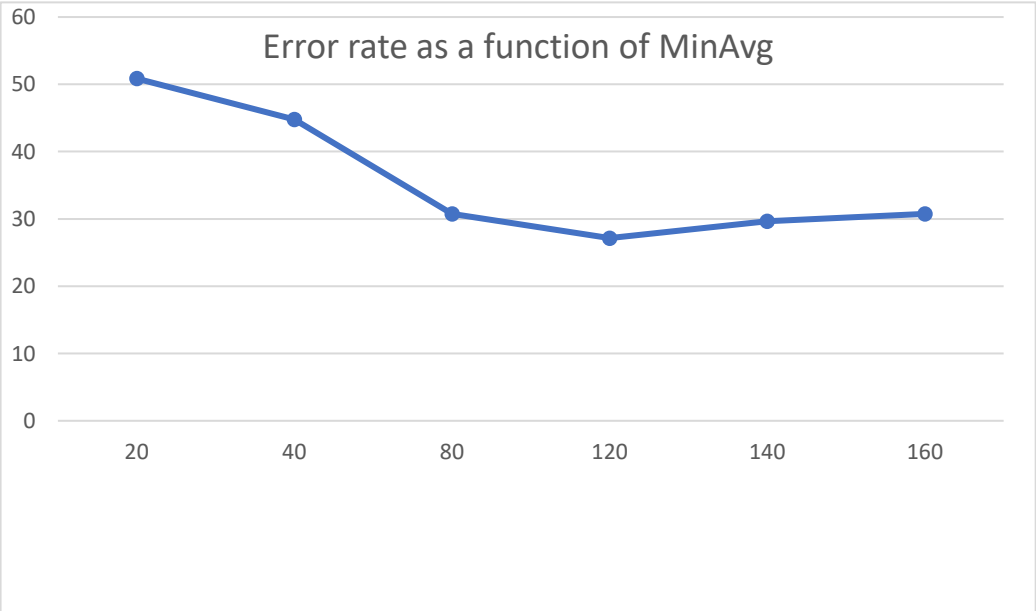
לעומת זאת, לפי שיטה ב', עבור ערך MinCount = 4,

כמות הפיקסלים בסביבת הפיקסל הנתון שערכם 128 ומעלה היא 1, ולכן סביבת הפיקסל **לא תעמוד בשיטה ב'.**

הנחנו כי ערך MinCount האידיאלי יהיה כמחצית מכמות הפיקסלים בסביבתו של כל פיקסל. למעט בשולי התמונה, בסביבתו של כל פיקסל נמצאים 8 פיקסלים, ולכן הערכנו כי ערך MinCount האופטימלי יהיה 4, אך בחנו גם ערכים אופציונליים אחרים, המוצגים בגרף הבא, כאשר P=10, L=10 וגודל המדגם הוא 10,000.



בחנו גם את שיטת AVG עבור ערכי MinAvg שונים, המוצגים בגרף הבא, כאשר P=10, L=10 וגודל המדגם הוא 10,000.

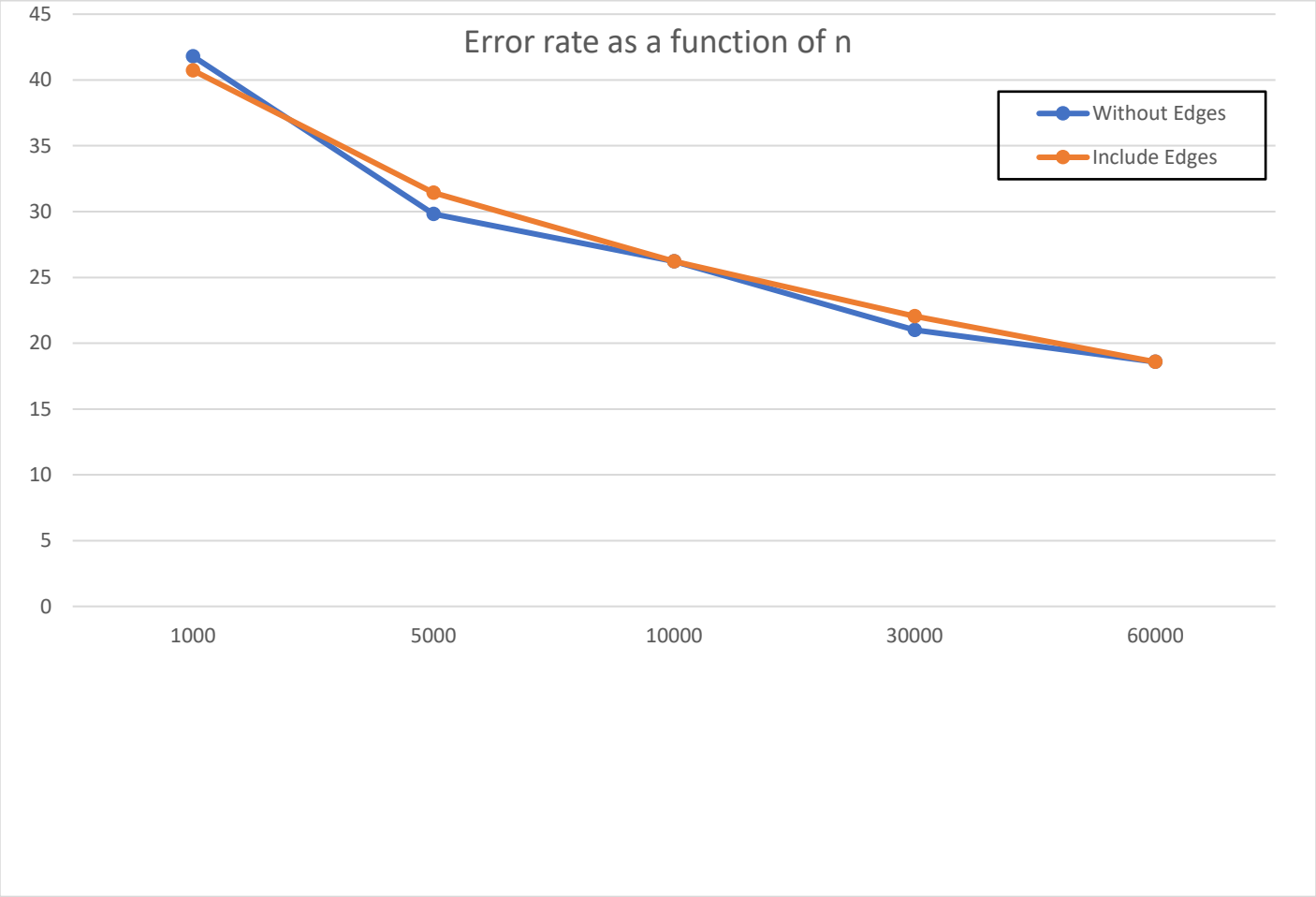


(2) **<u>סיווג הפיקסלים השונים בהקשר ההשפעה שלהם על התוצאה:</u>**

פיקסלים שמייצגים מידע ב"קצוות התמונה" למשל, משמעותיים הרבה פחות אל מול פיקסלים שנמצאים במרכז התמונה.

ניסינו לבחון 2 גישות – דילוג על חישוב "סביבת הפיקסלים" בקצוות התמונה וחישוב ערך הפיקסלים בלבד, אל מול חישוב סביבת הפיקסלים החלקית בקצוות התמונה.

בחרנו לעבוד לפי שיטת Counter, L=10, P=10 וגודל מדגם משתנה.



ראינו כי קיים הבדל מינורי בין השיטות, כאשר העדיפה מביניהן היא השיטה שלא כוללת את בדיקת סביבתם של הפיקסלים בקצוות התמונה.

(3) מציאת מכנה משותף בין רשומות עם תוויות זהות:

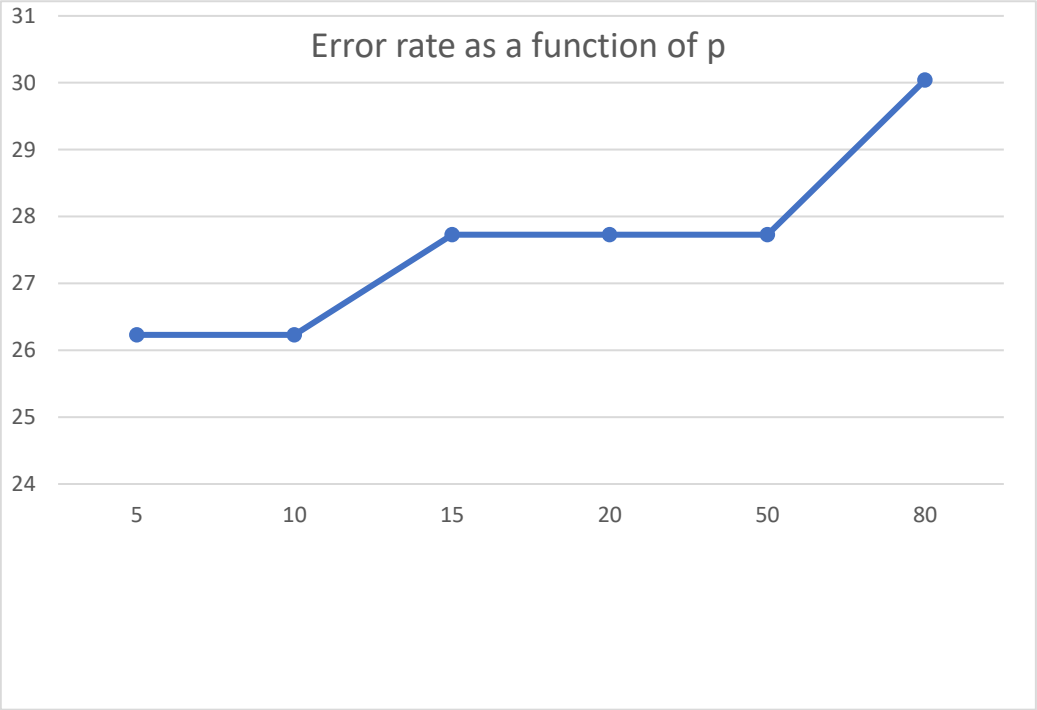
ניסינו לאפיין האם קיים מכנה משותף כלשהו המייחד כל אחת מהספרות, ולהטמיע את התובנות שלנו כתנאים. לאחר ניסיונות שונים ומגוונים, הבנו כי ללא הנחות נוספות על הקלט (מיקום הספרה ביחס לדף שעליו היא מופיעה למשל, או גודל הספרה ביחס לדף שעליו נכתבה), יהיה לנו קשה עד בלתי אפשרי ליצור עץ החלטה שיהיה רלוונטי לכל קלט.

(4) שינוי גודלו של P:

בחנו את תפקודם של ערכי P שונים.

בחרנו לבחון את השפעתו של P לפי שיטת Counter, L=10 וגודל מדגם של 10,000.

קיבלנו את הגרף הבא:

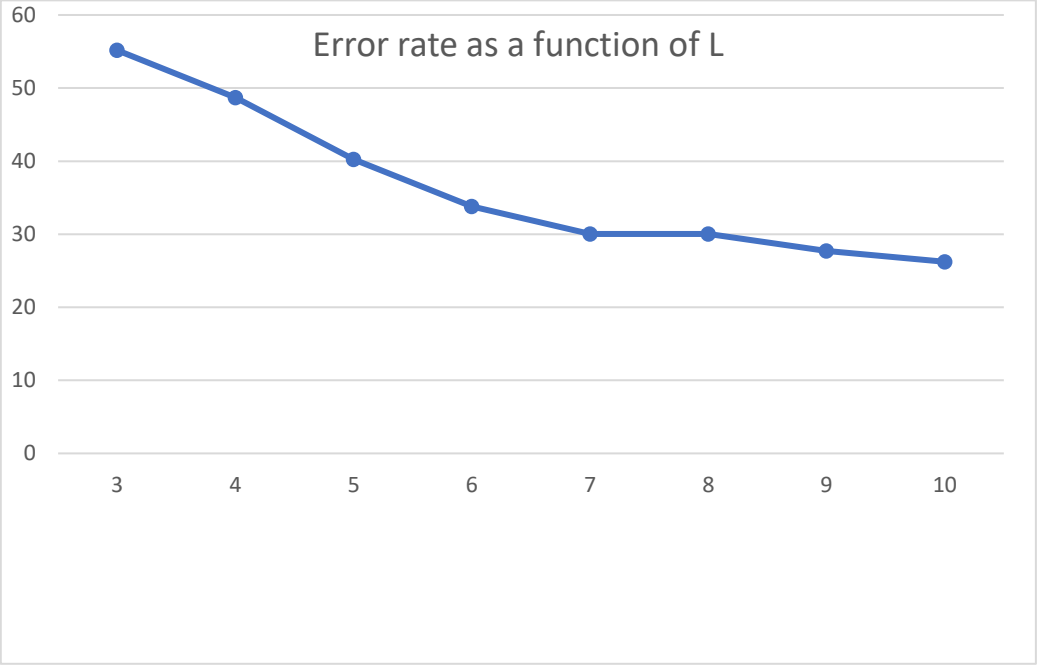


(5) **<u>שינוי גודלו של L</u>**:

בחנו את תפקודם של ערכי L שונים.

בחרנו לבחון את השפעתו של L לפי שיטת Counter, P=10 וגודל מדגם של 10,000.

קיבלנו את הגרף הבא:

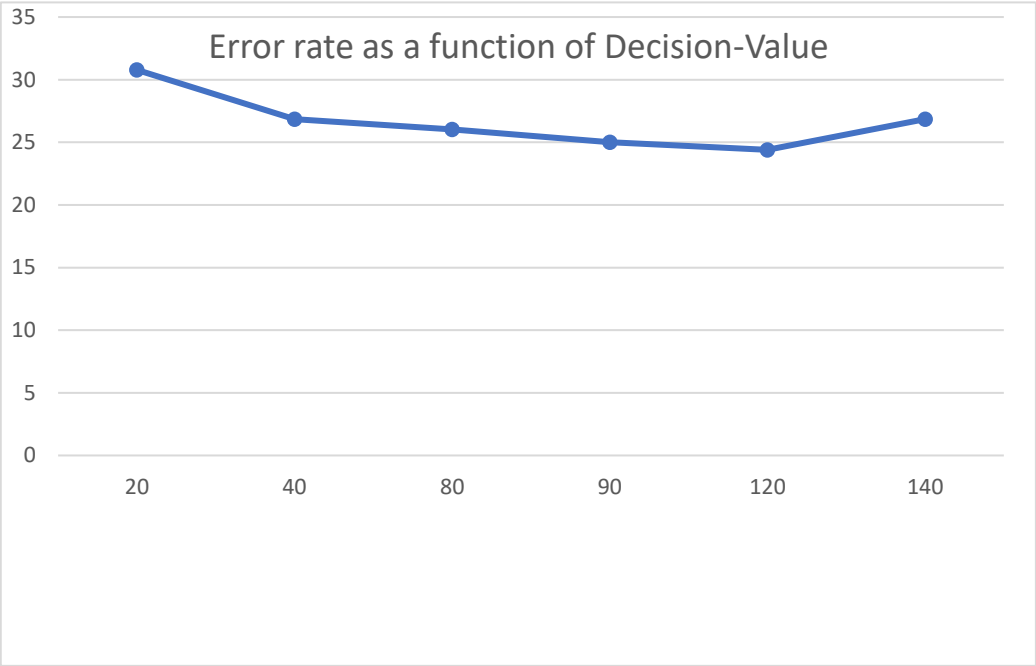


(6) **<u>שינוי הערך "128" :</u>**

ניסינו לבחון את התנהגות האלגוריתם עבור ערכים שונים, בעיקר נמוכים יותר מ – 128.

בחרנו לבחון את השפעתו של ערך זה לפי שיטת Counter, L=10, P=10 וגודל מדגם של 10,000.

קיבלנו את הגרף הבא :



אל מול התוצאות שהניבו שיטת Average ושיטת Counter , בחרנו להשתמש בסופו של דבר בשיטת Counter, עם הפרמטרים הבאים:

MinCount = 4 ושימור הערך 128.

למרות העובדה שהפרמטרים L ו P השפיעו על אחוז השגיאה, הם מתקבלים כקלט לאלגוריתם ע"י המשתמש ולכן לא מוגדרים כחלק מהאלגוריתם.

שאלה 2

במהלך כתיבת הפרויקט, נתקלנו באתגרים שונים. אלו האתגרים העיקריים:

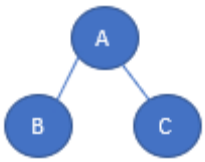
בחירת מבני נתונים אידיאליים:

(1) **פלט אלגוריתם בניית העץ** –

התלבטנו רבות כיצד לייצא את העץ שנוצר ע"י אלגוריתם הלמידה לאלגוריתם החיזוי. ניסינו לעשות זאת באמצעות ייצוא אובייקט של עץ, אך התקשינו לממש זאת באמצעות ספריות קיימות, והבנו שהאובייקט כולו שומר מידע רב שלא רלוונטי לאלגוריתם הפענוח.

לכן, בחרנו ליצור ערך String בעלות זיכרון מינימאלית, המייצג את ערכי הקודקודים בעץ בלבד ללא שאר הערכים שמבנה נתונים זה שומר, וליצור פונקציה באלגוריתם הפענוח שמפענחת מחרוזת זו לעץ בינארי.

לדוג, את העץ הבא



נייצג ע"י המחרוזת (A,(B),(C))

(2) **שמירת מבנה נתונים יחיד עבור כלל התמונות ופניה אליו לפי אינדקס** –

האלגוריתם מקבל תחילה מספר רב של תמונות (60,000) ולכל תמונה, מערך בן 784 תאים המייצגים את הפיקסלים שלו.

במהלך בניית האלגוריתם שמנו לב שתמונות רבות מועתקות פעמים רבות, והדבר עולה בזמן ריצה וזיכרון שניתן לחסוך. על מנת לפתור בעיה זו העברנו את כל מבני הנתונים להחזיק אינדקס של תמונה בלבד ולא את התמונה עצמה, על מנת לפנות לערך התמונה נפנה למאגר הנתונים הראשי במיקום האינדקס. יש לציין שמאגר התמונות הוא יחיד וכל שאר המאגרים מחזיקים int.

(3) ייצוג Node :

בחרנו לייצג Node ע״י השדות הבאים :

(א) <u>מאגר אינדקסי התמונות הרלוונטיות –</u> לכל Node יש מאגר אינדקסים המייצג את האינדקסים של כלל התמונות המגיעות אליו במצב העץ הנוכחי. באופן זה, בעת פעולת הרחבה של העץ, נבדקים התנאים רק על תמונות אלו ולא על כלל התמונות במאגר.

(ב) <u>תנאים רלוונטיים –</u> ע״מ לחסוך כפילויות של תנאים, כל Node מחזיק אך ורק מאגר של תנאים הרלוונטיים אליו, ללא תנאים שכבר השתמשנו בהם בדרך אליו (כך נחסכת בדיקה כפולה של תנאי).

(ג) <u>IG –</u> כל Node מתחזק את ערך הIG האופטימאלי עבורו, שחושב פעם אחת לכל Node ומחזיק בנוסף את החלוקה לבנים לפי התנאי שהניב ערך IG זה.

<u>זמן ריצה :</u>

לאור העובדה שהאלגוריתם נדרש לעבד כמות גדולה יחסית של מידע (כ – 60,000 רשומות המכילות כל אחת 784 אינדקסים), כל פעולה שבתכנות רגיל לא מקבלת ״טיפול מיוחד״ בהקשר זמני ריצה כמו מעבר על כלל הפיקסלים של תמונה נתונה, הופכת למשמעותית.

הבנו שהמפתח לקיצור זמן הריצה הינו ע״י ביצוע Trade-Off אל מול משאבי זיכרון.

לכן, בחרנו להשתמש בכמות גדולה יחסית של זיכרון ע״מ להרוויח זמני ריצה יעילים יותר בפעולות שונות.

היישום של עיקרון זה בא לידי ביטוי, למשל, בהגדרת המחלקה ״Node״ – בשונה מעץ בינארי רגיל, הוספנו למחלקה שדות נוספים שישמרו מידע שכבר חושב בעבר, למשל רשימת האינדקסים של רשומות שכבר הגיעו לאותו Node ספציפי.

בנוסף, מימשנו את הפתרונות הבאים:

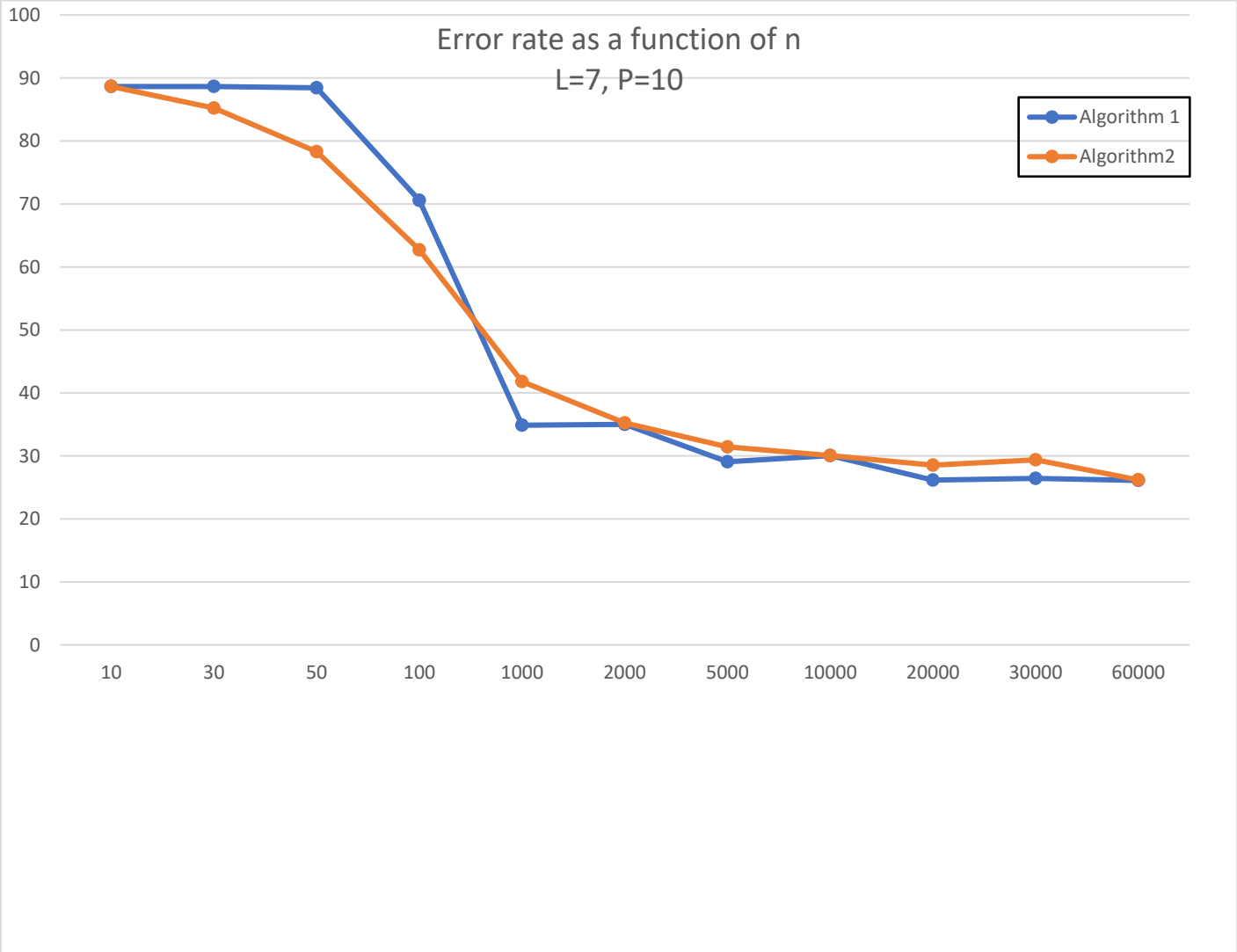
(1) **מערך דו – מימדי השומר את תוצאות התנאים** – בהינתן קבוצת תנאים, נחשב מבעוד מועד בפעם אחת, עבור כל תמונה את הפעלת קבוצת התנאים על כל פיקסל שבה ואת התוצאה נשמור במערך דו – מימדי גלובאלי.
בכל פעם שנרצה לחשב IG נעבור על כל התנאים עבור כל התמונות לקבלת התנאי המקסימאלי, נוכל לגשת ב – O(1) למאגר גלובאלי זה ולגלות את תוצאת תנאי ספציפי על פיקסל ספציפי.
במהלך ביצוע בדיקות, הוספת עדכון זה צמצמה בכמעט חצי את זמן הריצה.

(2) **שימוש בערמת מקסימום עבור ערכי IG** – בכל איטראציה של בניית העץ, אנו נדרשים לבחור להרחיב את העלה בעל ה-IG המקסימאלי. ע״מ לעשות פעולה זו ביעילות, בכל איטראציה אנו מחזיקים ערמת מקסימום עבור כל העלים הממוינת לפי ערכי ה - IG שלהם. בצורה זו, שליפת ה – Node בעל ערך ה – IG המקסימאלי מתבצעת ב – O(1).

שאלה 3

סעיף א'

עבור כל ערך n, נדגמו 3 הרצות של כל אלג' ונבחר ערך השגיאה הממוצע.



סעיף ב'

באופן כללי, ניתן לראות בצורה מובהקת ב – 2 האלגוריתמים, כי קיים יחס הפוך בין גודל המדגם לבין אחוז השגיאה.

כלומר, ככל ש –n גדול יותר, גודל השגיאה קטן.

בדיוק כמו שהיינו מניחים באופן אינטואיטיבי, ככל שעץ החיזוי נבנה ע"ב כמות נרחבת יותר של "תמונות אימון", איכות התוצר משתפרת.

ההבדלים הניכרים לעין הם שעבור n = 500, 2 האלגוריתמים נפגשים ב – Error rate דומה. עבור ערכים קטנים מ – 500, האלגוריתם הראשון מדויק פחות, ואילו עבור ערכים גדולים מ -500, האלגוריתם הראשון מדויק יותר.

סעיף ג'

ניתן לראות שעבור קלטים גדולים, האלגוריתם בגרסתו השנייה מדויק פחות מהאלגוריתם בגרסתו הראשונה. לאור העובדה שההפרשים בין אחוזי השגיאה הם מינוריים בין 2 הגרסאות, להנחתנו, סביבת הפיקסל אינה מעידה בצורה משמעותית על ערכו של פיקסל מסוים.

<u>סעיף ד'</u>

<u>גירסה ראשונה :</u> גודל המדגם = 10,000, P =10%, L =7

	0	1	2	3	4	5	6	7	8	9
0	791	0	25	11	5	47	59	21	9	12
1	0	1057	10	2	1	9	9	11	36	0
2	0	25	618	19	24	17	78	38	145	34
3	20	16	43	598	5	185	20	29	32	62
4	5	5	13	35	701	12	61	30	14	106
5	39	5	24	225	36	452	51	19	14	27
6	27	10	51	8	50	28	748	0	23	13
7	5	11	49	3	29	18	8	813	19	73
8	27	23	38	29	44	29	167	25	538	54
9	8	8	8	45	47	51	18	124	21	679

- ניתן לראות כי כמות ניכרת מהתמונות שערכן 3 פוענחו להיות הערך 5 (185 תמונות) ומנגד, המון תמונות שערכן 5 פוענחו להיות הערך 3 (225 תמונות). מה שמצביע על כך שספרות אלו דומות מבחינת האלגוריתם הראשון.

(תוצאה דומה יצאה גם בבדיקה על האלגוריתם השני).

- הערך 4 סווג להיות הערך 9 פעמים רבות אך מנגד, הערך 9 לא סווג להיות הערך 4 כמות פעמים ניכרת

(תוצאה דומה יצאה גם בבדיקה של האלגוריתם השני).

- הערך 8 סווג להיות הערך 6 פעמים רבות אך מנגד, הערך 6 לא סווג להיות הערך 8 כמות פעמים ניכרת.
- הערך 2 סווג להיות הערך 8 פעמים רבות אך מנגד, הערך 8 לא סווג להיות הערך 2 כמות פעמים ניכרת.

<u>גירסה שנייה:</u> גודל המדגם = 10,000, P =10%, L = 7

	0	1	2	3	4	5	6	7	8	9
0	820	0	10	23	16	27	31	11	7	35
1	0	1037	33	18	0	8	5	10	24	0
2	30	20	711	34	18	25	62	58	52	22
3	22	5	20	714	4	130	9	20	65	21
4	7	3	19	34	673	37	45	13	20	131
5	42	5	21	113	36	523	50	12	42	48
6	48	9	40	7	37	31	708	16	53	9
7	1	6	42	26	14	12	4	843	12	68
8	9	17	57	47	38	65	54	21	592	74
9	10	5	9	48	61	47	14	32	25	758

- ניתן לראות כי כמות ניכרת מהתמונות שערכן 3 פוענחו להיות הערך 5 (185 תמונות) ומנגד, המון תמונות שערכן 5 פוענחו להיות הערך 3 (225 תמונות). מה שמצביע על כך שספרות אלו דומות מבחינת האלגוריתם השני.
- הערך 4 סווג להיות הערך 9 פעמים רבות אך מנגד, הערך 9 לא סווג להיות הערך 4 כמות פעמים ניכרת.

מעיון בתוצאותיהם של 2 האלגוריתמים, ניתן להבחין כי התקבלו ע"י האלגוריתם השני ערכים נמוכים יותר עבור מרבית המספרים.

הבלבול המשמעותי והדו – כיווני בין הספרות 3 ו – 5 עדיין נשמר, אך ב"עוצמה נמוכה יותר", כמו גם הבלבול הניכר בסיווג הערך 4 לערך 9.

אך מנגד, כמות הפעמים שהערך 8 סווג להיות הערך 6 והערך 2 סווג להיות הערך 8 ירדה משמעותית.

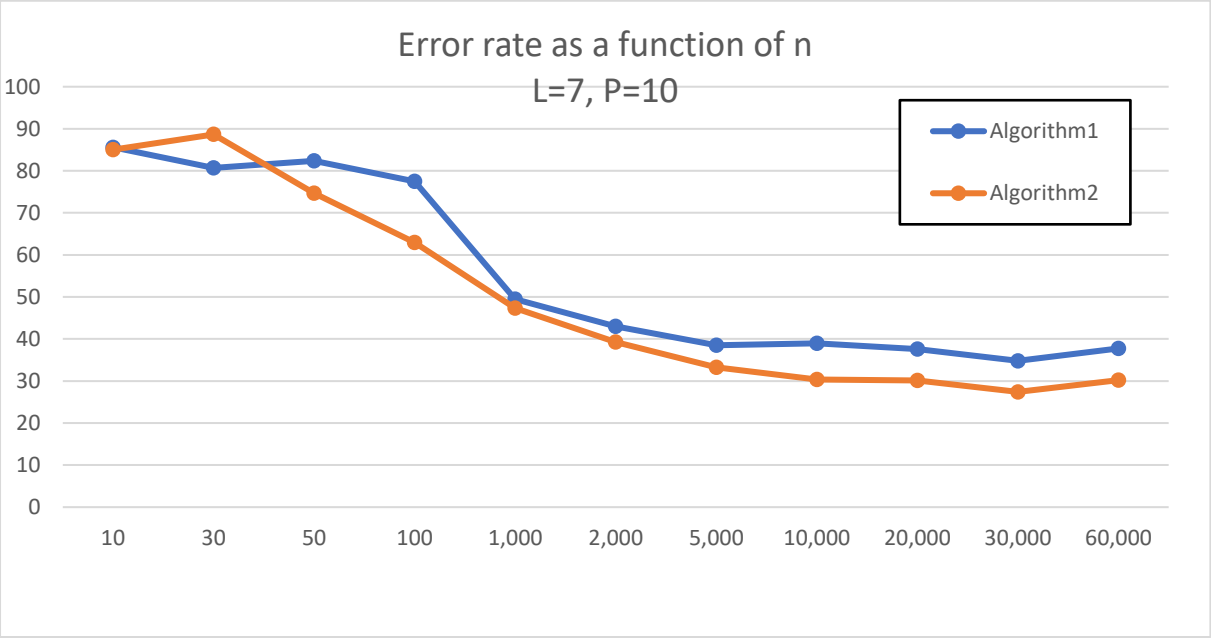
מכאן, ניתן להסיק שבדיקה בסביבת הפיקסל משפיעה באופן ניכר על פיזור השגיאה בין ערכים שונים ולכן האלגוריתם השני עדיף מבחינה זו.

שאלה 4

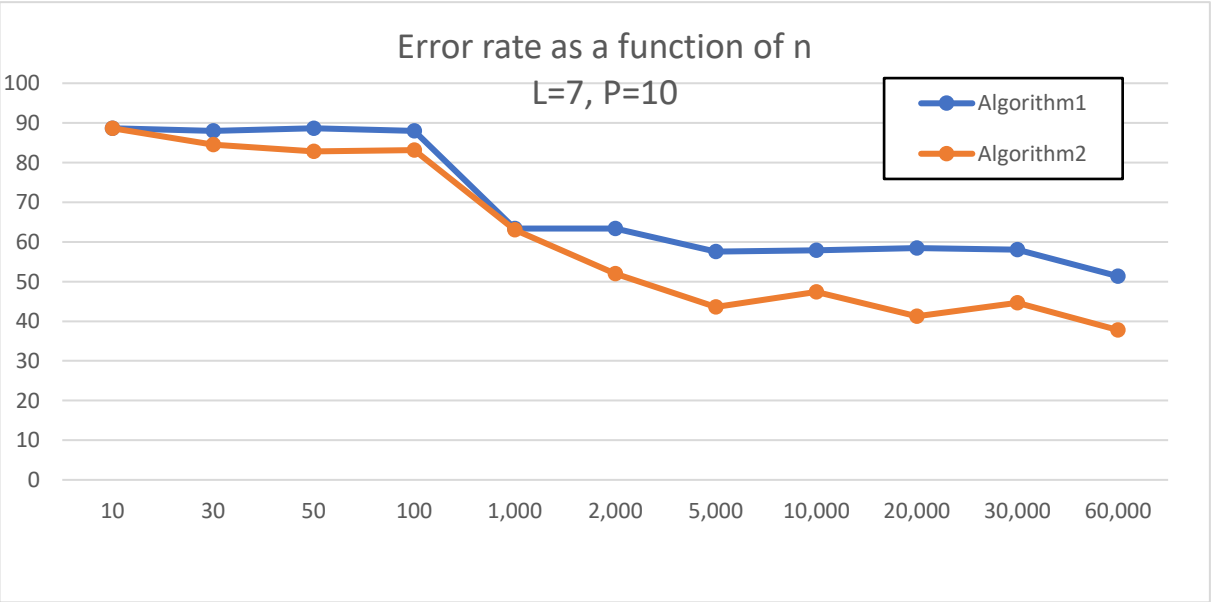
סעיף א'

עבור כל ערך n, נדגמו 3 הרצות של כל אלג' ונבחר ערך השגיאה הממוצע.

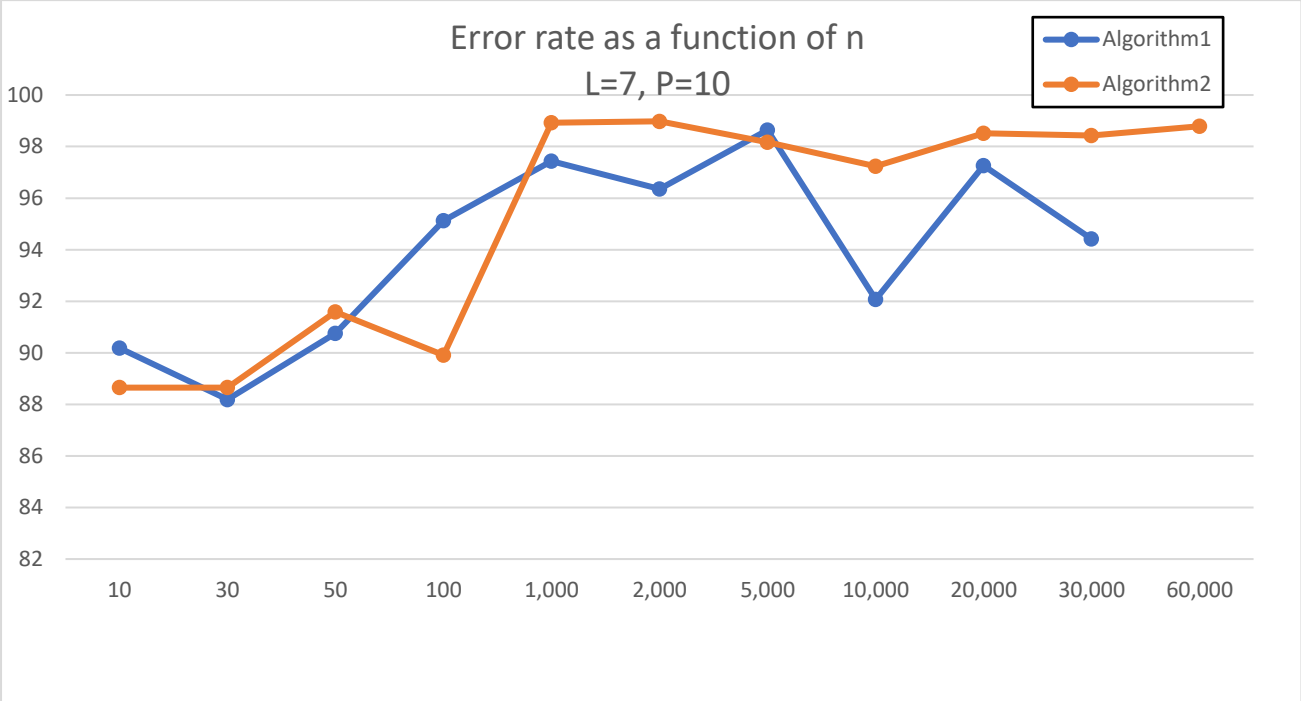
$\alpha = 0.1$:



$\alpha = 0.3$:



$\alpha = 0.6$:



סעיף ב'

ניתן לראות כי עבור ערכי α נמוכים יחסית, אלגוריתם 2 בעל אחוז שגיאה נמוך יותר מאלגוריתם 1.

בנוסף לכך, ב – 2 האלגוריתמים עבור ערך α נמוך יחסית, ניתן לראות כי אחוז השגיאה יורד ככל שגודל המדגם גדל.

מנגד, עבור ערך α גבוה (0.6), ניתן לראות כי אחוז השגיאה עולה באופן דרמטי ב – 2 האלגוריתמים, ונשאר גבוה גם כאשר גודל המדגם גבוה.

סעיף ג'

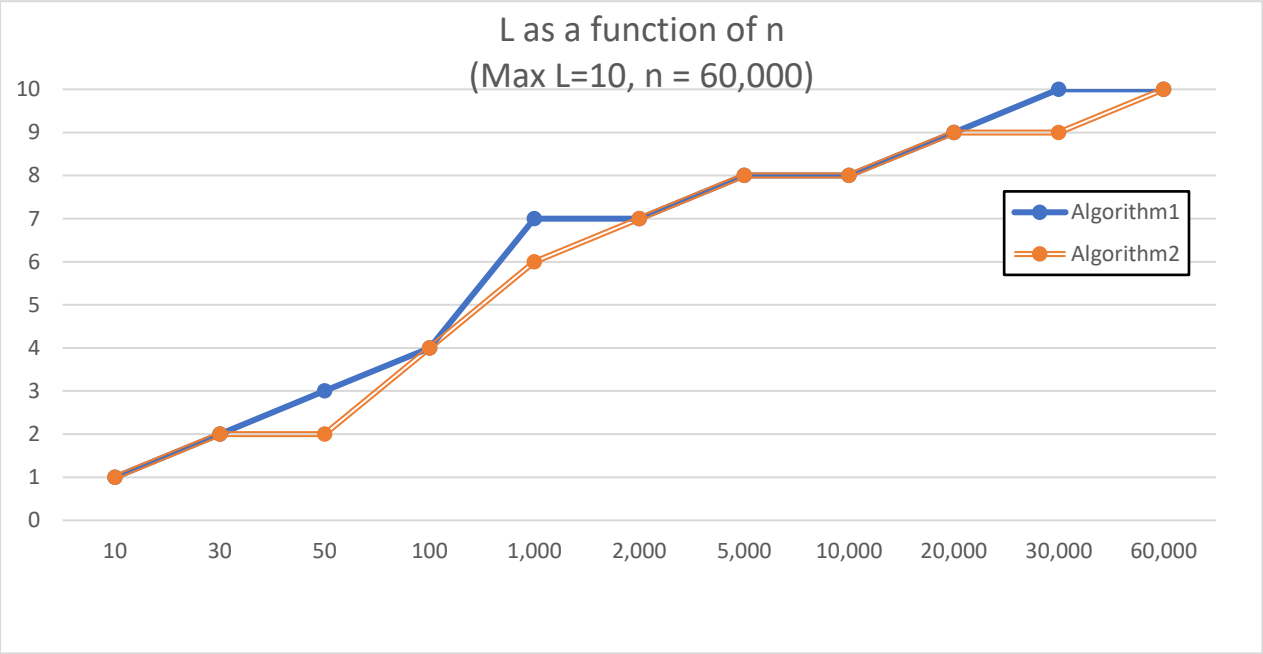
להנחתנו, ההבדל באחוז השגיאה בין אלגוריתם 1 ואלגוריתם 2 עבור רמות רעש נמוכות יחסית נובע מכך שהבדיקה בסביבתו של כל פיקסל נתון "מנטרלת" את הרעש בכך שהיא דוגמת עבור כל פיקסל את הנקודות הצמודות אליו, שהסתברותית (כאשר מקדם הרעש נמוך יחסית) לא השתנו ובכך האפקט של הרעש קטן.

מאידך, כאשר מקדם הרעש הוא גבוה מדי, גם סביבתו של הפיקסל מורעשת מספיק ולכן, איכות הדיוק של 2 האלגוריתמים נפגעת באופן דומה.

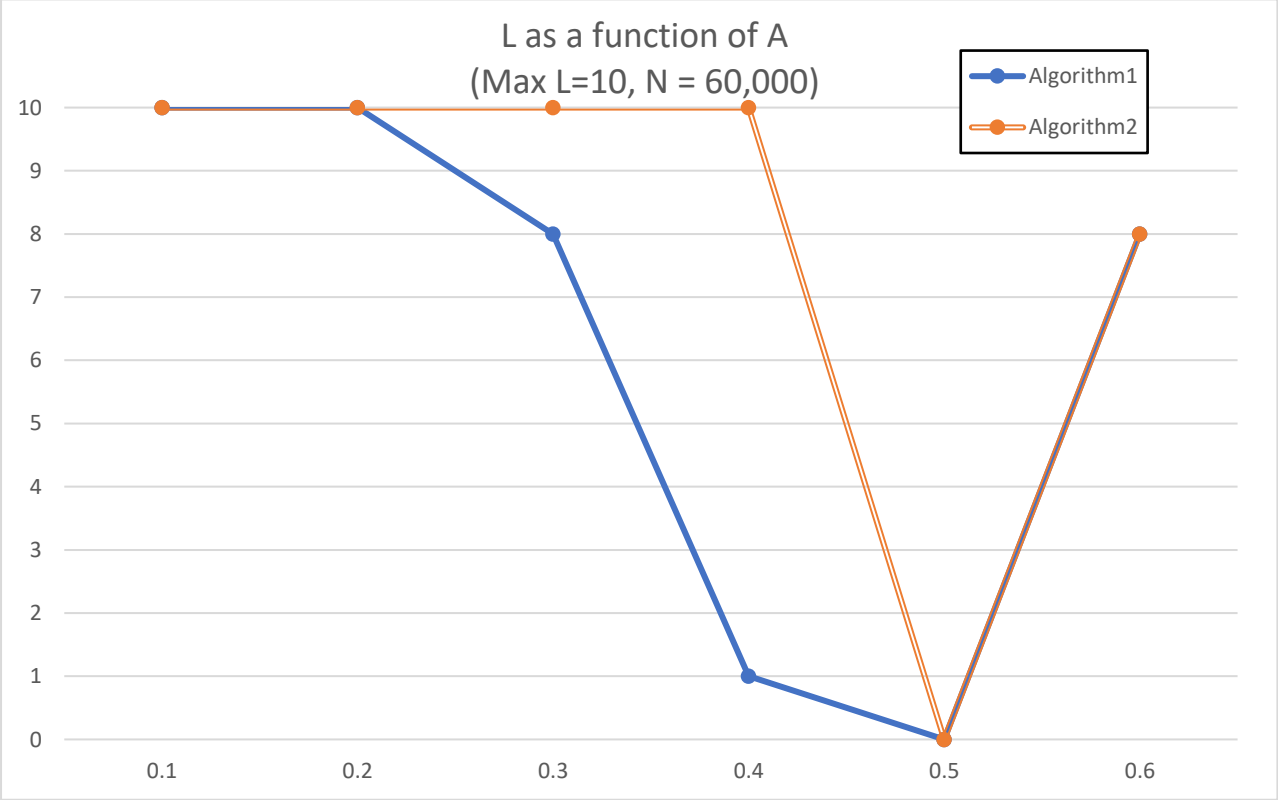
שאלה 5

סעיף א'

גודל העץ כתלות בגודל המדגם:



גודל העץ כתלות במקדם הרעש:



סעיף ב'

מהגרף המתאר את גודל העץ כתלות בגודל המדגם, ניתן לראות עבור 2 האלגוריתמים כי ככל שגודל המדגם גדול יותר, מתקבל עץ גדול יותר.

מהגרף המתאר את גודל העץ כתלות במקדם הרעש, ניתן לראות כי ככל שערכי α גדלים, גודל העץ קטן.

בנוסף, ניתן לראות מהגרף כי אלגוריתם 2 מייצר עץ גדול יותר גם עבור מקדמי רעש גבוהים יחסית, שבאלגוריתם 1 גרמו ליצירת עץ בגודל קטן.

סעיף ג'

להבנתנו, תוצאותיו של הגרף המתאר את גודל העץ כתלות בגודל המדגם הגיוניות, שכן עבור מס' דוגמאות גבוה יותר, מדגם האימון מגוון יותר, ויש לאלגוריתם יותר מידע לעבד ולנתב, לכן ככל שמס' צמתי ההחלטה שהאלגוריתם מייצר גדול יותר, כך המידע מעובד בצורה מדויקת יותר.

עבור גרף מקדם הרעש, התוצאות מתכתבות עם התוצאות שהתקבלו בסעיפים הקודמים, המצביעות על כך שהאלגוריתם בגרסתו השנייה פחות רגיש לרעשים (ניתן לראות זאת במסקנותיו של סעיף 3 בשאלה הקודמת). נזכור כי מדדנו את השפעת מקדם הרעש על גודל מדגם מקסימאלי ולכן, הגיוני שיתקבלו תוצאות השקולות לתוצאות המתקבלות עבור גודל מדגם גבוה יחסית ללא מקדם רעש כלל.