

2016

String Compression

ספר פרויקט

מגישים:

רעות מזרחי: 305160079

עדי פרלוב: 200477842

מנחות:

מיקה עמית, ליאת רוזנברג

מועד הגשה: 23/09/16

תוכן עניינים

1. מבוא	שגיאה! הסימניה אינה מוגדרת.
2. מסמכי הגדרת פרויקט	שגיאה! הסימניה אינה מוגדרת.
3. אפיון הפרויקט	שגיאה! הסימניה אינה מוגדרת.
3.1 סביבה ושפת פיתוח	5
3.2 דרישות ומשימות לפרויקט	5
3.3 תכנית עבודה	5
4. תיאור תהליכים מרכזיים ואלגוריתמים	שגיאה! הסימניה אינה מוגדרת.
4.1 אלגוריתם LZ77 עם טעויות	6
4.2 Forecasting	6
4.3 דיאגרמת תהליכים מרכזיים	7
5. אלגוריתם LZ77 כאב טיפוס	שגיאה! הסימניה אינה מוגדרת.
6. צילומי מסך עיקריים של המערכת	שגיאה! הסימניה אינה מוגדרת.
7. קשיים טכניים ודרכי התמודדות	שגיאה! הסימניה אינה מוגדרת.
7.1 שיפור הדיוק בתהליך בחירת bit כטעות	10
7.2 Dynamic Convert Integer to binary	10
8. מסמכי בדיקות	שגיאה! הסימניה אינה מוגדרת.
8.1 Application Sanity test	11
8.2 Exception Handling	12
8.3 LZ77 vs LZ77 with mistakes	13
9. סיכום דוחות חודשיים	שגיאה! הסימניה אינה מוגדרת.
9.1 סיכום דוחות חודשיים ומפגשים עם המנחים	15
10. סיכום הפרויקט ומסקנות	שגיאה! הסימניה אינה מוגדרת.
11. כתיבת ספרות ומקורות נוספים	שגיאה! הסימניה אינה מוגדרת.

הרקע והמוטיבציה העיקרית לכיווץ מחרוזות, וכיווץ מחרוזות בינאריות בפרט, הוא זיכרון RAM:

זיכרון גישה אקראית (RAM ראשי תיבות של Random Access Memory) הוא שם כללי למספר רב של סוגי זיכרון מחשב, המתאפיינים כולם ביכולת המעבד לגשת ישירות לכל תא בזיכרון לפי כתובתו, לכתוב בו ולקרוא ממנו. ההתייחסות הנפוצה ל**זיכרון מחשב** היא למעשה ההתייחסות לזיכרון הגישה האקראית הראשי שלו.

בעידן של היום השימוש במחשבים נרחב מאוד, ועולה הצורך לשמור מידע רב על גבי מחשבים/שרתים, מכיוון שזיכרון המחשב הוא **סופי** – כמות המידע שניתן לשמור על המחשב מוגבלת, וכאן עולה הצורך לכוון מחרוזות.

מכיוון שכל Byte זיכרון מורכב מ-8 Bits, כלומר סדרה של 0'ים ו-1'ים, ולכן אזור רצוף בזיכרון הוא סידרה בינארית ארוכה מאוד.

ולכן על ידי תוכנות לכיווץ מחרוזות ניתן להציג סידרה של bits בזיכרון, כסדרה קצרה יותר, שיהיה ניתן להחזירה למחרוזת המקורית. (Uncompressing).

במסגרת המחקר שערכות גב' מיקה עמית וגב' ליאת רונזנברג, אנו מניחים כי במחרוזת נתונה יש "טעויות" (ביטים הפוכים) – ולכן, ע"פ הנחה זו, האלגוריתם צריך למצוא ביטים אלה כך שמספר הטעויות יהיה מינימלי וכיווץ המחרוזת יהיה אופטימלי.

האלגוריתם מתבסס על האלגוריתם של LEMPEL ZIV 77 לכיווץ מחרוזות, כמו אלגוריתמים רבים נוספים שמתבססים על אלגוריתם זה והביאו לשיפור ביחס הכיווץ.

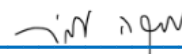
2. מסמכי הגדרת פרויקט

3.1 דף הגדרת פרויקט

שם סטודנט א: עדי פרלוב	תז: 200477842
שם סטודנט ב: רעות מזרחי	תז: 305160079
שם הפרויקט: כיווץ מחרוזות	
מנחה: מיקה עמית	ליאת רוזנברג
טלפון: 0525978226	נייד: 0545686012
	Email:

חתימת

המנחה:



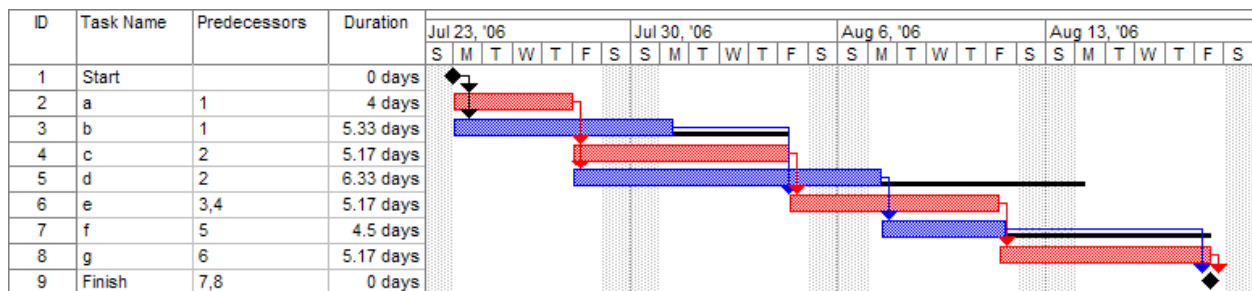
מטרת הפרויקט:	תכנות אלגוריתם לכיווץ מחרוזות, חלקם בעזרת הספרות, וחלקם בעזרת הסטודנטים.
דרישות קדם:	תכנות מונחה עצמים, אלגוריתמים, מבני נתונים.
שלבי ביצוע תוכנית עבודה:	<ol style="list-style-type: none"> 1. קריאת מאמרים והבנתם. 2. תכנות אלגוריתמים ידועים. 3. הצגת תוצאות. 4. סיעור מוחות. 5. תכנות אלגוריתם חדש לכיווץ מחרוזות עם טעויות.
שלבים עיקריים ומטרות ראשיות:	הבנת אלגוריתם LZ77 על מנת להשתמש בו לכיווץ מחרוזות עם טעויות.
שלבי בונסים	<ol style="list-style-type: none"> 1. הצגת הוכחה – אופטימליות הכיווץ האלגוריתם LZ77. 2. תכנות האלגוריתם לכיווץ מחרוזות ע"פ HUFFMAN
סביבה נדרשת	Visual Studio (C++)
קו סיום משוער	יוני 2016

1. A Universal Algorithm for Sequential Data Compression
2. Huffman Encoding and Data Compression
3. Lempel-Ziv , Peter Shor, MIT
4. Introduction to algorithm , Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest

ספרות ומאמרים

2.2 תכנון לוח זמנים

1. משימות לביצוע
2. סדר משימות
3. זמן לכול משימה
4. תלויות
5. אילוצים ונתיב קריטי



<u>משך זמן:</u>	<u>משימה:</u>
חודשיים	1. קריאת מאמרים והבנתם + הצגת הוכחה
חודש	2. תכנות האלגוריתם + פגישת תוצאות
חודש	3. הופמן – קריאת מאמר + תכנות האלגוריתם.
חודשיים	4. סיעור מוחות על אלגוריתם LZ77 לכיוון עם טעויות + תכנות אלגוריתם.

* המשימות תלויות אחת בשנייה לפי הסדר הנתון.

3. אפיון הפרויקט

3.1 סביבה ושפת פיתוח

את המימוש לאלגוריתם כתבנו בשפה C++.

Visual studio 2015 – IDE

Windows Form Application – Application.

3.1 דרישות לפרויקט

- ✓ קריאה והבנה של מאמר *A Universal Algorithm for Sequential Data Compression*
- ✓ הוכחת אופטימליות קידוד האלגוריתם LZ77 והצגה פרונטלית של ההוכחה.
- ✓ מימוש האלגוריתם LZ77. – טעינת המחרוזת מקובץ TXT, שמירת ניתוח המידע Words בקובץ TXT בפורמט הדרוש, כולל שמירה של תת המחרוזת המקושרת לכל מילה, ration compression, new size of compressed string.
- ✓ קריאה והבנה של המאמר *Huffman Encoding and Data Compression*.
- ✓ מימוש תכנית המחשבת את מספר הביטים הדרושים בזיכרון על מנת לייצג את המחרוזת המכוסה ע"פ האלגוריתם של Huffman Encoding – טעינת המחרוזת מקובץ TXT, ניתוח המידע ע"פ אלגוריתם הופמן לכיוון, הדפסת מספר הביטים למסך.
- ✓ מימוש האלגוריתם LZ77 עם טעיות – יכולת קונפיגורציה ליתר עבור k (מספר הטעיות עבור כל תת מחרוזת ארוכה ביותר), דירוג טעות ע"י נוסחה.
- ✓ שיפור ביכולת הדירוג על ידי forecasting.
- ✓ הצגת אפליקציה בגרסה סופית העונה על כל הדרישות.

3.2 תכנית העבודה ומשימות לביצוע

- משימה 1 - קריאה והבנה של מאמר *A Universal Algorithm for Sequential Data Compression*
- משימה 2 – הוכחה אופטימליות הקידוד של האלגוריתם LZ77.
- משימה 3 – מימוש האלגוריתם LZ77 בגרסתו המקורית.
- משימה 4 – קריאה והבנת המאמר *Huffman Encoding and Data Compression*.
- משימה 5 – מימוש תוכנית המחשבת את מספר הביטים הדרושים בזיכרון לאחר הפעלת האלגוריתם Huffman Encoding.
- משימה 6 – מימוש אלגוריתם LZ77 עם טעויות.
- משימה 7 – הכנסת יכולת forecasting על מנת לשפר את אופטימליות הכיוון.

4. תיאור תהליכים מרכזיים ואלגוריתמים

4.1 אלגוריתם LZ77 עם טעויות:

האלגוריתם LZ77 עם טעויות המבוסס על LZ77 Algorithm, נועד על מנת למצוא את מספר הביטים המינימלי ואת האינדקס שלהם במחרוזת המקורית, כך שאותם ביטים יוחלפו בהופכי שלהם.

תיאור האלגוריתם:

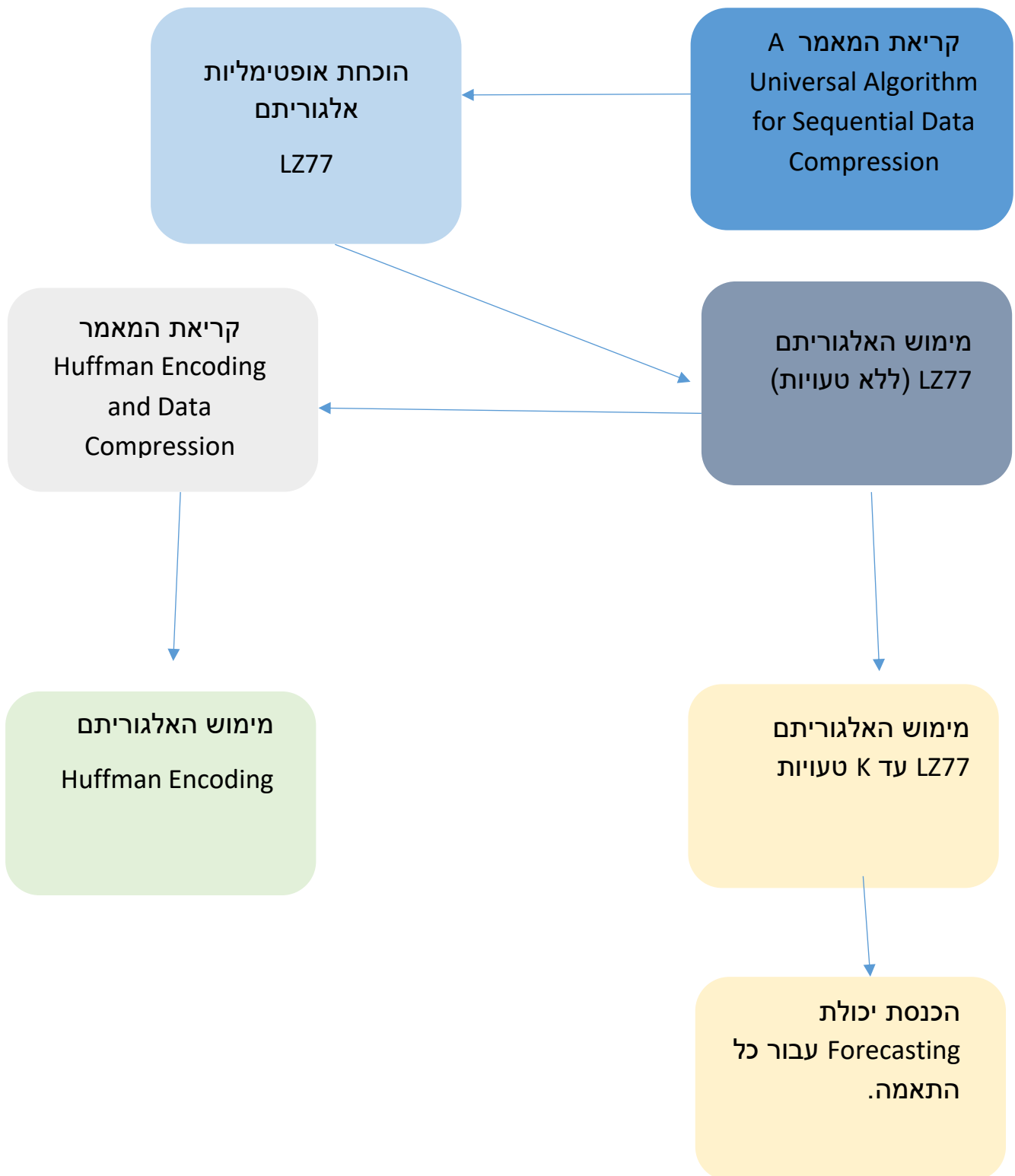
1. עבור כל איטרציה מצא את המחרוזת הארוכה ביותר עד כה.
2. החלף את ה- next char בביט ההופכי ודרג את הטעות באופן הבא:
 $(current\ length\ of\ longest\ match/i) + forecasting\ number$
כאשר i מייצג את מספר הטעות עבור מחרוזת זו.
3. חזור על סעיף 2 למשך k פעמים.
4. בחר את הטעות עם הציון הגבוה ביותר כאשר ה- priority של טעות נמוכה גבוה יותר.
5. החלף את הביטים שנמצא עבורם "טעות" במחרוזת המקורית.
6. המשך את ריצת האלגוריתם לאיטרציה הבאה.

4.2 Forecasting:

האלגוריתם מחזיר את מספר הפעמים שהמחרוזת הארוכה ביותר באיטרציה הנוכחית מופיעה בהמשך ה- buffer.

למשל עבור המחרוזת הבאה: 0111 והתבנית הבאה: 11, הפונקציה תחזיר 2, מכיוון ש- 11 מופיע 0111 וגם 0111.
פעולה זו משפרת את יכולת הדיוק של האלגוריתם לקבוע האם לקחת את הביט הנוכחי כטעות או עדיף להשאירו בלי שינוי.

4.3 דיאגרמת תהליכים מרכזיים:



5. אלגוריתם LZ77 כאב טיפוס

אלגוריתם למפל-זיו הוא אלגוריתם לדחיסת נתונים שפותח על ידי אברהם למפל ויעקב זיו מהטכניון.

ישנם מספר אלגוריתמים שונים שפותחו על בסיסו של האלגוריתם אשר שיפרו את ביצועיו. הדחיסה היא מסוג "דחיסה משמרת מידע" כלומר המידע הדחוס משוחזר במלואו ללא נזקים.

האלגוריתם:

האלגוריתם LZ77 הוצג ע"י למפל-זיו בשנת 1977, האלגוריתם משיג דחיסה על ידי החלפה של מופעים חוזרים של מידע, במצביע לעותק יחיד של אותה פיסת מידע, במופע הראשון שלו בקלט הרצפים הלא דחוס. הרעיון בבסיס הקידוד הוא כי כל מילה בקידוד היא המילה הארוכה ביותר שנראתה עד לאותה נקודת זמן, בתוספת של אות אחת

תהליך האלגוריתם נעשה ע"י סמן (cursor) אשר מתקדם על המחזורות ובכל שלב נשמרים 3 נתונים :

(1) מיקום ההתאמה הארוכה ביותר שנראתה לפני ה- cursor.

(2) אורך ההתאמה הארוכה ביותר.

(3) התו הבא במחזורות שלא נראה עדיין כחלק מהמחזורות הארוכה ביותר.

בכל שלב ה- cursor מתקדם על המחזורות באורך המחזורות הארוכה ביותר + 1.

המחזורות המכווצות מורכבת מ- Words אשר כל מילה מהצורה הבאה:
<offset, length, next char>.

Offset מציין את המיקום ההתחלתי של תת מחזורות הארוכה ביותר שנמצאה בשלב הנוכחי.

Length מציין את אורך תת המחזורות הארוכה ביותר שנמצאה בשלב הנוכחי.

next char מציין את התו הבא שטרם נראה.

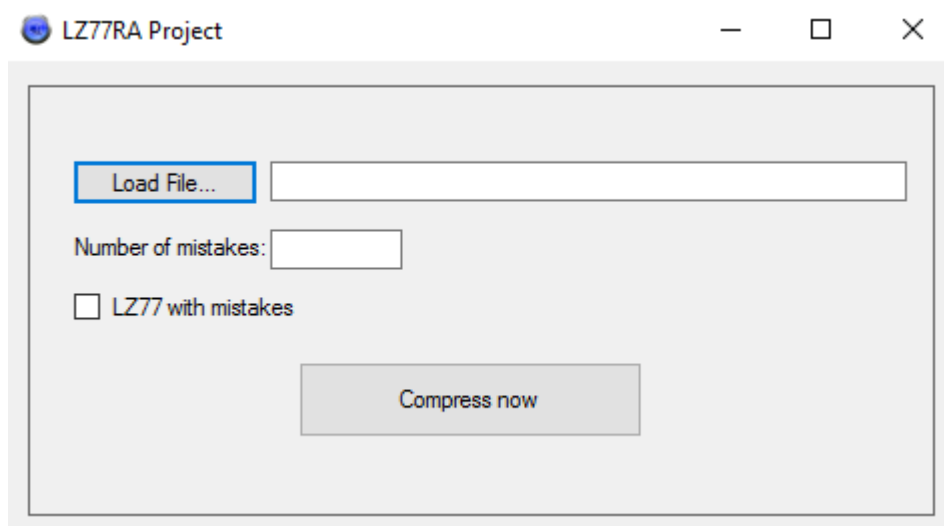
מכיוון שמספר X בבסיס דצימלי ניתן לייצוג ב- $\log X$ ביטים, אזי שאורך המחזורות המכווצות יהיה

$(\log X * 2 + 1) * \text{Number of words}$, כלומר את Offset ו- Length נייצג כמספר בינארי שחסום ב- \log של אורך המחזורות המקורית.

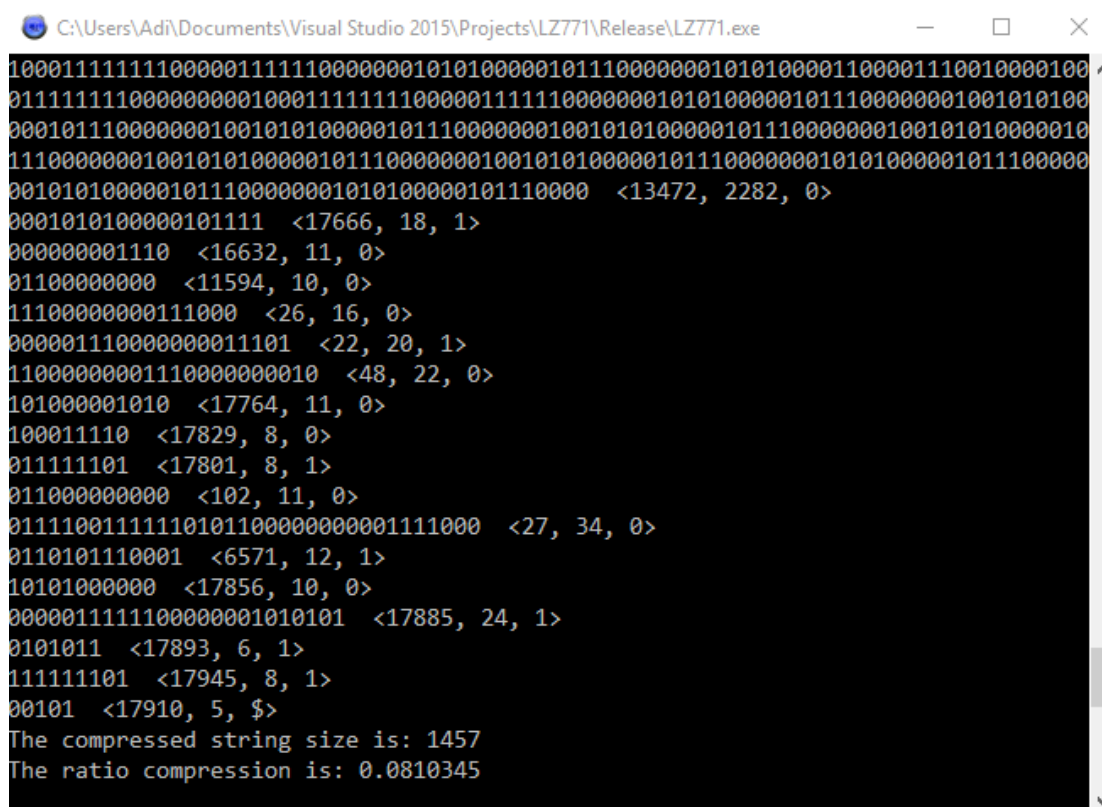
האלגוריתם LZ77 משמש כבסיס לאלגוריתם LZ77 עם טעויות בכך שעבור כל איטרציה LZ77 פועל רגיל, על buffer שעבר שינוי עם הביטים שנבחרו כ-"טעויות".

6. צילומי מסך עיקריים של המערכת

6.1 UI – The main screen of the application:



6.2 Info window – This screen should print the all words & related subs strings from the compression process.



7. קשיים טכניים ודרכי התמודדות

7.1 שיפור הדיוק בתהליך בחירת bit כ- "טעות":

כחלק מהאלגוריתם LZ77 עם טעויות, אנו מדרגים כל טעות לפי הנוסחה $\text{longest match} / i$ כאשר i מייצג את מספר הטעות הנוכחית. ועל מנת לשפר את דיוק במתן הדירוג עבור כל טעות על מנת לקבל מספר מינימלי של טעויות וכיוון אופטימלי - הוספנו את היכולת forecast אשר משקללת לתוך החישוב את מספר ההופעות של המחרוזת הארוכה ביותר הנוכחית בהמשך ה- `buffer`.

כלומר, הנוסחה מהצורה $\text{longest match} / i + \text{forecasting number}$ - על ידי הוספת יכולת זו, מחרוזת אשר מופיעה מספר פעמים רב בהמשך המחרוזת מקבלת עדיפות מכיוון שהיא תופיע כחלק ממחרוזת ארוכה ביותר עתידית. וכך למעשה, אנו משפרים את יכולת האלגוריתם לזהות ביט כ- "טעות" לקבלת כיוון אופטימלי.

7.2 Dynamic convert Integer to binary

על מנת לייצר את המחרוזת המקודדת, האלגוריתם מייצר מכל `word` את הערך הבינארי הרצוי, נזכיר שכל `word` הוא מהצורה `<offset, longest match size, next char>`.

מכיוון שאורך ה- `buffer` ידוע רק בזמן ריצה אנו צריכים לבצע המרה של `offset` ו- `longest match size` לערכם בבסיס 2, בייצוג של $\log X$ ביטים (X הוא אורך ה- `buffer`), אז צריך לבצע המרה דינאמית מבסיס דצימלי לבינארי בגודל $\log X$ ביטים.

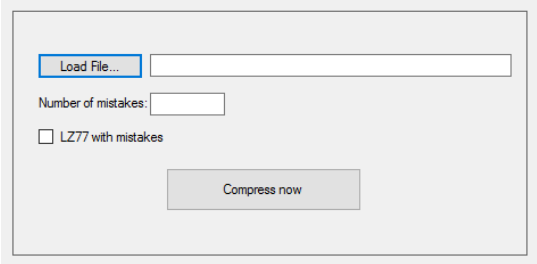
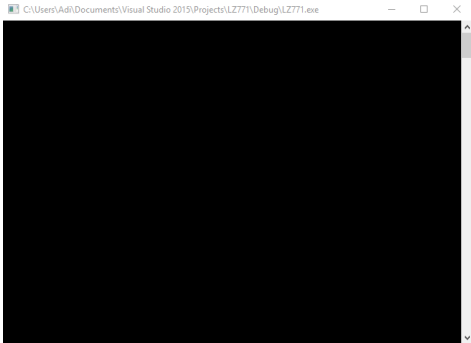
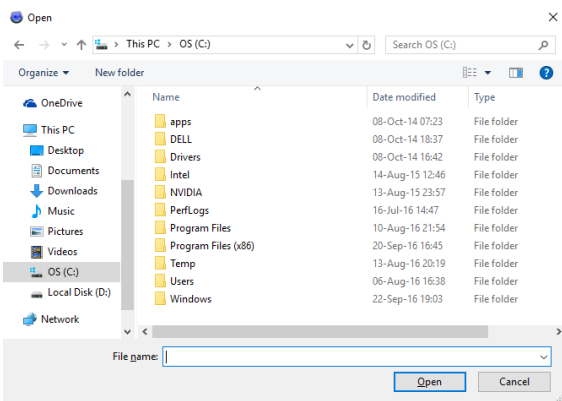
פתרון שנשקל ונפסל: הכנסת `boost library` המכיל את ה- `dynamic_bitset` class, פתרון זה נפסל משום שספריה זו כבדה מאוד, ומכילה המון יכולות שלא תורמות לנו.

הדרך בה התמודדנו עם הקושי הוא מימוש של `dynamic convert from integer to binary`, ע"י כך שאנו ממירים את המספר לבינארי ע"י אלגוריתם מוכר, `module 2` והכפלה באינדקס המתאים בכפולות של 10. לאחר מכן חישבנו את מספר הביטים שקיבלנו. ו- "ריפדנו" באפסים לקבלת מספר באורך $\log X$ ביטים.

8. מסמכי בדיקות:

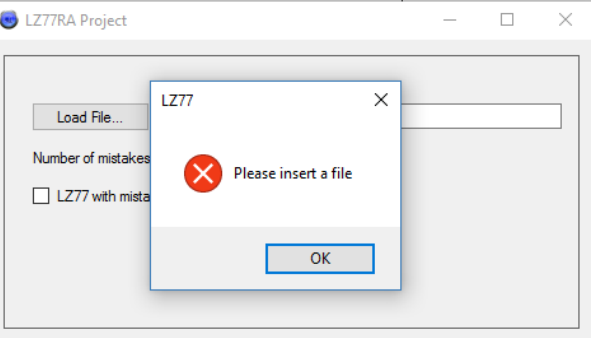
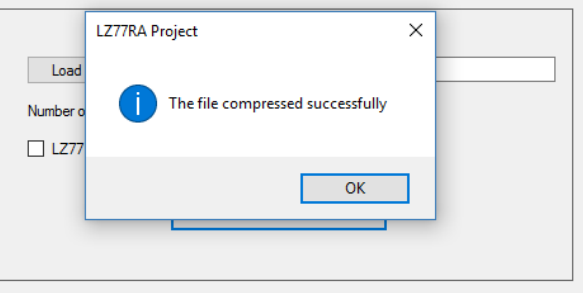
8.1 מטרת הבדיקה: Application Sanity test .

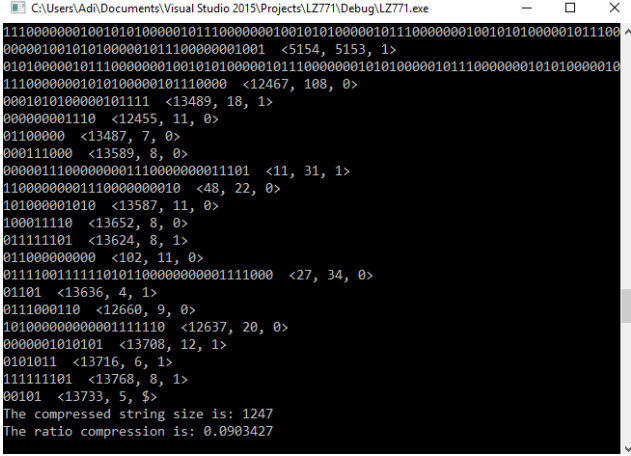
תהליך הבדיקה:

צעד	פעולה	תגובה צפויה
1.	הפעל את הקובץ LZ77RA Project.exe	<p>ה-UI נפתח ונראה כך:</p>  <p>נפתח Info Window:</p> 
2.	לחץ על הכפתור "Load File" ובחר קובץ.	<p>נפתח dialog עם רשימת קבצים לבחירה:</p>  <p>הנתיב המלא של הקובץ מופיע ב- text box הצמוד לכפתור "load file".</p>
3.	סמן את השדה "LZ7 with mistakes"	השדה "LZ7 with mistakes" מסומן:

		
		4

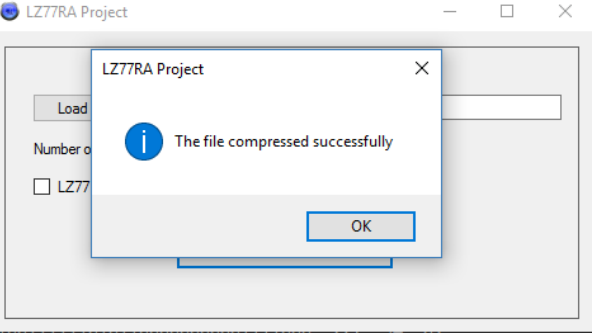
8.2 מטרת הבדיקה: Exception Handling

תגובה צפויה	פעולה	צעד
<p>מופיעה הודעת שגיאה הבאה:</p> 	<p>1. לחץ על 'Compress now' מבלי שנבחר קובץ לכיווץ</p>	
<p>נפתח חלון עם רשימת קבצים לבחירה</p>	<p>2. לחץ על הכפתור "Load File"</p>	
<p>הקובץ מופיע בשדה הצמוד לכפתור "Load File"</p>	<p>3. בחר קובץ עם סיומת txt כרצונך</p>	
<p>הודעה תוצג למסך שתהליך הכיווץ הסתיים בהצלחה.</p> 	<p>4. לחץ על הכפתור "Compress now"</p>	

בחלון Info window יודפסו כל ה- words בנוסף למחרוזות המתאימות:		
		
הקובץ ה- log נוצר ונמצא ב- path של האפליקציה.	פתח את הקובץ "LZ77_Result.txt"	5.
קובץ המכיל את המחרוזות המכווצות נוצר ונמצא ב- path של האפליקציה.	פתח את הקובץ "LZ77_Compressed.txt"	

8.3 מטרת הבדיקה: שיפור אופטימליות הכיווץ LZ77 עם טעויות לעומת LZ77.

צעד	פעולה	תגובה צפויה
1.	טען קובץ לכיווץ. (חזור על פעולה זו עבור קבצים בגודל 30000 bits, 40000 bits, 50000 bits).	הנתיב המלא של הקובץ מופיע ב- text box הצמוד לכפתור "load file".
	לחץ על הכפתור "Compressed now" כאשר ה- check box לא מסומן.	תהליך הכיווץ התבצע והתקבלה הודעה על סיום.
2.	שמור את הערך המופיע ב- LZ77_Result.txt string size is: xxxx.	/
3.	לחץ על הכפתור "Compressed now" כאשר ה- check box מסומן. ומספר הטעויות מקונפג ל-3.	תהליך הכיווץ התבצע והתקבלה הודעה על סיום.

		
/	4. שמור את הערך המופיע ב- The compressed string size is: yyyy בקובץ LZ77_Result.txt.	
	5. בדוק ש- $xxxx \leq \text{עניין}$	

9. סיכום דוחות חודשיים

9.1 סיכום דוחות חודשיים ומפגשים עם המנחים:

אינדקס	סיכום הפגישה	חודש
1	דיון על שלבי ריצת האלגוריתם LZ77 ועל היותו אלגוריתם greedy, דיברנו על חישוב גודל המחרוזת המכוצת, סיבוכיות, ומימוש האלגוריתם. המשך העמקת הידע בהוכחת אופטימליות הכיווץ של האלגוריתם LZ77 והצגת ההוכחה במפגש הבא.	דצמבר
2	הצגה של הוכחת אופטימליות הכיווץ של האלגוריתם LZ77, דנו ב- worst case, ובמקרה הרנדומלי שהוזכר קודם וראינו מדוע הכיווץ נעשה באופן אופטימלי בצורה אסימפטוטית. כמו כן, דיברנו על שיטות שונות למימוש האלגוריתם LZ77, היתרונות והחסרונות של כל שיטה, סיבוכיות מימוש האלגוריתם, וקיבלנו הנחיות ודרישות לפלט הרצוי של הגרסה הראשונה לאלגוריתם LZ77 ללא טעויות.	ינואר
3	הצגה בפני מיקה וליאת את ההתקדמות, ודנו בדרכים שונות למימוש כך שיתאים מבחינת סיבוכיות לאלגוריתם הסופי LZ77 עם טעויות, בנוסף התבקשנו לממש תכנית המקבל מחרוזת בינארית אקראית ומוציאה כפלט את אורך המחרוזת המכוצת (בביטים) ע"פ Huffman Encoding.	פברואר
4	הצגה של המימוש לאלגוריתם LZ77 (ללא טעויות), דנו בשיפור יחס הדחיסה הצפוי במחרוזות בינאריות ע"י שימוש באלגוריתם LZ77 עם טעויות, כמו כן, קיבלנו הנחיות למימוש האלגוריתם, ודנו ברעיון בו נוכל להחליט בצורה מדויקת יותר האם תו מסוים במילה בתוך מחרוזת בינארית צריך להיות מסומן כטעות, ע"י כך שנבחן את כמות הפעמים שבו המילה מופיעה בהמשך המחרוזת. הגענו למסקנה שבדיקה כזו אכן תביא ליחס כיווץ טוב יותר, ולכן נוסף פרמטר זה לשיקול האם לסמן תו כטעות. בנוסף, דנו בתוכנית אשר מחשבת את מספר הביטים לייצוג מחרוזת דחוסה ע"פ האלגוריתם של Huffman וקיבלנו הנחיות לכך.	מרץ
5	הצגה של את המימוש לאלגוריתם Huffman Encoding, ודנו על האופן בו מתבצע הכיווץ. בנוסף, הצגנו את המימוש של האלגוריתם LZ77 עם טעויות עד כה, ודנו על התוצאות הצפויות ביחס הכיווץ לאחר שינוי זה.	אפריל
6	הצגה של המימוש לחלקו הראשון של האלגוריתם LZ77 עם טעויות, ערכנו בדיקות על מחרוזות רנדומליות, ומחרוזות תבניות, וראינו את השיפור הצפוי ביחס הכיווץ. בנוסף, דיברנו על מימוש החלק השני ועל אופן חיפוש המחרוזת לקבלת תוצאות אופטימליות.	מאי
7	במהלך הפגישה עם מיקה וליאת הצגנו את המימוש לאלגוריתם LZ77 עם טעויות על כל חלקיו, הרצנו מספר דוגמאות יחד וניתחנו את אופן הכיווץ, את יעילותו ואת ההשפעה של כל חלק על ה- ratio compression. דנו על העדיפות בחיפוש מחרוזת bit by bit בהמשך ה- buffer. כמו כן, דנו על השיפור בביצועים של האלגוריתם LZ77 עם טעויות.	יוני
8	לא התקיימה פגישה חודש יולי	יולי
9	לא התקיימה פגישה חודש אוגוסט	אוגוסט

העבודה על הפרויקט הייתה מעניינת ומאתגרת מאוד, נחשפנו לתחום חדש מבחינתנו – כיווץ מחרוזות. לאחר שלדמנו על הנושא, ראינו שזה הוא תחום רחב מאוד שנעשו בו ניסיונות רבים למציאת אלגוריתמים לכיווץ אופטימלי, שיפור אלגוריתמים קיימים וכו'. ללא ספק זה הוא תחום חשוב במדעי המחשב מכיוון שבעיית הזיכרון הסופי לא תעלה מן העולם, והחיפוש המתמיד אחר פתרונות לקידוד מחרוזות ימשך.

מעבר לסיפוק שליצור פרויקט המתאים לצרכיו של הלקוח (במקרה שלנו מיקה וליאת), ותרם למחקר שהן עורכות במסגרת הדוקטורט, אנו יכולים להגיד בוודאות שאת הידע והכלים שצברנו במהלך הפרויקט אנו ניקח איתנו להמשך העבודה בתעשייה.

במהלך הפרויקט התמודדנו עם לא מעט קשיים, חלקם תיאורטיים וחלקם מעשיים, ובעזרת הליווי הצמוד שקיבלנו מגב' מיקה עמית וגב' ליאת רוזנברג צלחנו את כל הקשיים והגענו לתוצאה משביעת רצון, ועל כך רצינו להודות להן.

11.1 ספרים ומאמרים:

1. A Universal Algorithm for Sequential Data Compression
2. Huffman Encoding and Data Compression
3. Lempel-Ziv , Peter Shor, MIT
4. Introduction to algorithm , Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest.