**פרויקט סיום – דחיסת נתונים**

**מגישים:**

רון עמר – 318541745

פלג סויסה – 207226812

**נושא הפרויקט: LZW מורחב –** דחיסה עפ"י האלגוריתם LZW בתוספת פרמטר d לעומק חיפוש.

**הרעיון הכללי:**

קידוד על פי האלגורתים LZW את התו היחיד נקודד כמספר של קוד ההאסקי שלו ( עד 256) לרצף תווים נבנה מילון שבו לכל רצף יש מספר משלו - קוד (החל מ256) בכל איטרציה שהאלגוריתם ישתמש במילה שקיימת אצלו במילון על מנת לקודד, הוא יוסיף את אותה המילה בצירוף התו הבא למילון להמשך הדחיסה.

הדרך שאנו בחרנו בה כדי לממש את האלגוריתם היא שימוש במילון בצורת Hash-Map כדי לצוות STRING

ל Int המתאים לו.

את המספרים שמרנו בקובץ וכדי לדחוס עוד יצרנו לכל מספר שקיים זוג תווים משלו .

כדי לפענח השתמשנו במילונים שבנינו.

\*\*\*\*תפקידו של העומק לבדוק איזה צירוף תווים כדאי לאלגוריתם להשתמש לאחר מספר צעדים, בשונה מ-LZW הרגיל והידוע הלוקח את המחרוזת הארוכה ביותר שהוא מוצא.

\*\*\* קבצי Bmp דחסנו לקובץ של מספרים ומיד לאחר מכן פענחו . ( הדחיסה לא תשתמש בפונקציות numbersCompress, *restoreNumbers*

\*\*\*קבצי txt דחסנו בשיטת Lzw הפכנו את הקובץ מספרים שנוצר לקובץ דחוס יותר על ידי ציוות כל מספר לזוג chars

כדי לפענח היינו צריכים לחזור בחזרה על כל השלבים

**MAIN**

**פונקציית lzwEncoded:**

מקבלת את הקובץ (לאחר שהומר ל-byte) אותו נרצה לדחוס ואת פרמטר העומק.  
checkWord2– רצף התווים הארוך ביותר הנוכחי שאינו נמצא במילון.  
newCode – המילה הבאה שנרצה לקודד (אם זה תו אחד נקודד לפי האסקי, במידה וזה רצף תווים נקודד לפי המילון.  
printWord – משתנה באורך הרצף המקודד, העוזר לנו להתקדם בטקסט  
Count – משתנה שמאותחל לעומק שהמשתמש בחר (מס' הצעדים).

נתחיל בלולאת for המתחילה ב-0 עד אורך הטקסט. בכל איטרציה נבדוק האם המילה הנבדקת כעת לא קודדה כבר והאם אורכה ארוך מתו אחד. במידה והמילה הנבדקת עונה על שני התנאים, לאחר מכן מתבצעת בדיקה באמצעות שליחת המילה הארוכה ביותר שנמצאה במילון וזו שלפניה לפונקצייה depthcheck , הפונקציה תחזיר את האופציה הטובה ביותר לקידוד עפ"י מספר הצעדים שהמשתמש בחר, וכך נמשיך כל X צעדים. את המילה שנבחרה נקודד לקובץ שלנו ואת המילה + התו הבא נכניס למילון.

**פונקציית depthCheck:**

הפונקציה תבדוק איזה מילה מחזירה לנו את המחרוזת הארוכה ביותר לפי מספר הצעדים שהמשתמש בחר וכך נדע איזה מבין המילים לבחור להמשך הקידוד.

**פונקציית numbersCompress:**

את הקובץ שקידדנו למספרים באמצעות הפונקציה הקודמת נהפוך כל מספר לשני char בדרך הבאה – נקרא את הקובץ ובכל פעם שנמצא רווח נדע שלפני זה קיים מספר שאותו נהפוך לשני char בתבנית הבאה:

מספר ראשון = AA

מספר שני = AB

מספר שלישי = AC

וכן הלאה..

כל מספר והקידוד שלו נשמור במבנה נתונים HASH MAP.  
את הדחיסה שיצרנו נשמור לתוך קובץ שהמשתמש יבחר את שמו (באמצעות ממשק GUI)

**פונקציית restoreNumbers:** את הקובץ המקודד שיצרנו באמצעות הפונקציה הקודמת נפענח בדרך הבאה: כלשני תווים נשלח ל-HASH MAP ונבקש בחזרה את המספר המתאים. את התוצאה נשמור לתוך קובץ חדש בשם end2 .

**פונקציית lzwDecoded:** מקבלת את הטקסט המקודד (string code) ומפענחת.

Check1 **–** הטקסט הנבנה המפוענח.

הטקסט שקיבלנו מהפונקציה הקודמת נחלק למערך באמצעות split שיפריד בין הרווחים ונקבל מערך מלא במספרים. נעבור על המערך ונבצע את הבדיקות הבאות:  
בדיקה ראשונה האם המילה היא חלק מקוד ההאסקי – כלומר תו אחד מפענח לאות המקודדת ומוסיף אותה לטקסט השלם המפוענח. אחרת, ניגש למילון ובודק את המילה הנבדקת שם, ואת המילה המפוענחת המתקבלת מוסיף לטקסט המפוענח. ולבסוף מחזיר את הטקסט המפוענח.