

## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא, מחברת של שובל

### הרצאה 1 – מבוא

**דרישה:** הזיכרן לא יקר כמו שהוא היה לפני הרבה שנים אבל בכל אופן יש הגבלות על רוחב הפס, אז או שנרחב את רוחב הפס שזה לא תמיד מתאפשר או שתשתמש באლגוריתמי דחיסה וכך נוכל להעביר יותר נתונים על אותו רוחב פס.

**מטרה:** שיפור ביצועים מבחינתי זמינים וגם מבחינת שטח אחסון.

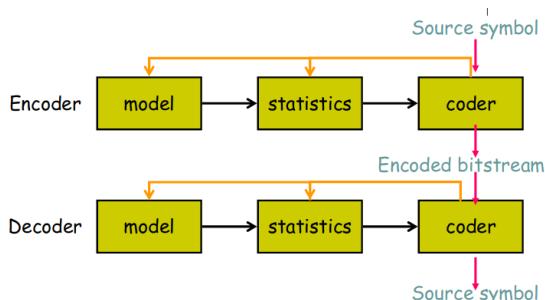
#### מערכת הדחיסה מורכבת משלוש פעולות:

1. שלב המידע: הנחות שעושים על המידע שהינו מתחודדים או שאוספים את המידע על הקובץ. על מנת שהקידוד והדחיסה יהיו מסונכרנים, הקידוד צריך להיות מוכר גם למקודד וגם למפענחו.

2. איסוף סטטיסטי: כאשר יש מידע סטטיסטי על הקובץ, יהיה ניתן לדוחס אותו בצורה שונה. נוכל לדעת למשל האם קיימת תלות מסוימת בין אותיות. או למשל נדע מהי האות השכיחה ביותר ונוכל לבחור לה קידוד קצר יותר.

3. הקידוד עצמו: רוב תוכן הקורס. צריך להכיר את קובץ המקור, ומайлוי אלמנטים מורכב הקובץ.  
שניהם א"ב:

- הא"ב של הקובץ המקורי.
- הא"ב של הקידוד, בדרך כלל עם אפסים אחדות.



יש מקודד ומפענחו, בכל שלב אפשר לעדכן את המודל. העדכן צריך להיות מסונכרן עם מה שהמפענחו עושה. לוקחים א"ב מהמקור (source symbol) ויצרים את מילת הקוד והמפענחו עושה את הפעולה ההיפוכה וממיר את זה חוזרת לקוד הרגיל (source symbol).

#### קוד אונרי Unary Code

קוד אונרי – 0, 1, 10, 110, 1110, ...,  
תפקידם של האפסים הוא בעצם להפריד בין מילות הקוד, במעבר בין מילת קוד לבאה מוסיפים 1.

#### טרמינולוגיה-סימונים:

- א"ב המקורי:  $S = [s_1, s_2, \dots, s_n]$
- הסתברות:  $\sum_{i=1}^n p_i = P$  כך ש  $1 - P$
- מילות הקוד:  $C = [c_1, c_2, c_3, \dots, c_n]$
- אורך מילות הקוד:  $|C| = [|c_1|, |c_2|, \dots, |c_n|]$
- אורך הקוד הצפוי:  
$$E(C, P) = \sum_{i=1}^n p_i * |c_i|$$

## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא, מחברת של שובל

### דוגמה

$s_i$	$p_i$	Code 1	Code 2
a	0.67	000	00
b	0.11	001	01
c	0.07	010	100
d	0.06	011	101
e	0.05	100	110
f	0.04	101	111
Expected length		3.0	2.22

- Code 1:  $|C| = [3, 3, \dots, 3]$

### קוד חסר רישות prefix free codewords

קוד חסר רישות נקרא כך אם אף מילת קוד אינה רישא (תחילית של קוד) של מילת קוד אחרת. לדוגמה: אם קיימת מילת הקוד 1101, אסור שיופיעו מילות הקוד 110, 11, 1, מכיוון שהן כולן רישות של 1101. (כמובן שגם 1101 אסור שתי מילות קוד זהות).

### פיענוח באופן ייחודי uniquely decipherable

כל רצף מסוים של קוד יכול להיות מפוענח בצורה אחת בלבד. בקיצור נכתוב UD. לפני ההגדירה הזאת ניתן לראות שאם הקוד הוא UD אז הוא UD. האם ניתן לומר את הדבר ההפוך? כלומר בהינתן קוד UD האם ניתן להגיד שהוא חסר רישות? לא!

נראה דוגמה נגדית ניקח את הקוד האונרי ונהפוך אותו

a	0	->	0
b	10	->	01
c	110	->	011
d	1110	->	0111

פה כל אות היא פרטיקס של הבאה בתור אבל הקוד הזה הוא UD כי האפסים מפרידים עדין בין מילות הקוד.

### קודשלם complete code

כל מחרוזת אינסופית למחצה הננתנת לפענוח בצורה ייחודית. הכוונה למחרוזת אינסופית היא: .....00111010.....

אין סופית למחצה: שיש נקודות המשך רק מצד אחד של המילה, למשל: ....0101111101\*

\* בשბיל שהקוד יהיה קודשלם, צריך שהצע המציג את הקוד יהיה עצם מלא.

- Which of the following codes is UD?

### תרגיל

$s_i$	$p_i$	Code 1	Code 2	Code 3	Code 4
a	0.5	0	0	0	0
b	0.25	0	1	10	01
c	0.125	1	00	110	011
d	0.125	10	11	111	0111
Expected length		1.125	1.25	1.75	1.875

איזה קוד מהקודים הבאים הוא UD?

קוד 1: לא UD מכיוון ש 6, a, b קודדים באותו אופן.

קוד 2: לא UD מכיוון ש c, aa קודדים באותו אופן.

איזה קוד יותר טוב 3 או 4?

קוד 3 הוא יותר טוב מקוד 4 מפני שהוא מילת הקוד הממוצעת שלו קטנה יותר.

## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא , מחברת של שובל

### קוד מייד' Instantaneous code

המפענה יודע את הפענוח ברגע שבו הושלמה הקריאה של מילת הקוד .  
 לדוגמה, הקוד האונרי, בכל פעם שמופיע "0" אנו יודעים שמלת קוד נגמרה.  
 לעומת זאת הקוד ההפכי לקוד האונרי הוא לא מייד', רק בתחילת מילת הקוד הבאה, כאשר נראה  
 את ה "0" אז נדע בדיעד שלפני כן סימנו לקרוא מילה.  
 \* מיידיות אינה תנאי הכרחי לUD. יתכן קובץ שהוא UD אבל הוא לא יהיה מייד' (הקוד האונרי  
 ההופך).

### Dangling suffix

אם יש 2 אותיות:  $a = 010$ ,  $b = 01011$   
  $a$  היא תחילית של  $b$ , והוא xix dangling suffix הוא מה שנותר, "השארית" = 11.

### בדיקה האם הקוד הוא UD

- עוברים על כל ה xix dangling suffix בקוד, אם במהלך הבדיקה מוצאים מילים שהו  
 אותן מקורית בקוד, כלומר שנייתן לפרש אותה כאות ממש, נדע שהtekst הוא לא UD.
- אם סימנו לעבור על כל הטקסט ולא מצאנו, הטקסט הוא UD.

פירוט התהיליך:

נבחן את כל זוגות המילים הקיימות בקוד.

1. נבנה רשימה של כל מילות הקוד.
2. אם קיימת מילה הקוד  $a$  שהיא תחילית של מילת הקוד אחרת,  $b$ , נוסף לרשימה את ה  
 dangling suffix עד ש:
  - נקלט xix dangling suffix שהוא נמצא במילות הקוד המקוריות -> הקוד לא UD.
  - אין עוד dangling suffix ייחודיות -> הקוד הוא UD.

דוגמאות:

- |   |  |
|---|--|
| <ul style="list-style-type: none"><li> Codewords {0,01,10}</li><li> 0 is a prefix of 01 → dangling suffix is 1</li><li> List - {0,01,10,1}</li><li> 1 is a prefix of 10 → dangling suffix is 0 - which is an original codeword!</li><li> → the code is not UD</li></ul> | <ul style="list-style-type: none"><li> Codewords {0,01,11}</li><li> 0 is a prefix of 01 → dangling suffix is 1</li><li> List - {0,01,11,1}</li><li> 1 is a prefix of 11 → dangling suffix is 1 - already there</li><li> → the code is UD</li></ul> |
|---|--|

## סיכום קורס דחינת נתונים א'

<https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא , מחברת של שובל

### Sardinas–Patterson algorithm

- For given strings  $S$  and  $T$ , the **left quotient** is the residual obtained from  $S$  by removing some prefix in  $T$ .
- Formally  $S^{-1}T = \{d \mid ad \in T, a \in S\}$

```

 $i \leftarrow 1$ 
 $s_1 \leftarrow C^{-1}C - \{\epsilon\}$ 
while true
     $s_{i+1} \leftarrow C^{-1}s_i \cup s_{i-1}^{-1}C$ 
     $i \leftarrow i + 1$ 
    if  $\epsilon \in s_i$  or  $c \in s_i$  for  $c \in C$ 
        print not UD and exit
    else if  $\exists j < i$  such that  $s_i = s_j$ 
        print UD and exit
    
```

$s_1$  – קבוצת כל המילים שמתקבלות כאשר לוקחים מילה ב-  $C$  ומחסרים את הרישא שלה שגם נמצאת ב-  $C$ .

$s_{i+1}$  – קבוצת כל המילים המתקבלות כאשר לוקחים מילה ב-  $C$  ומחסרים את הרישא שלה או שלוקחים מילה ב-  $s_i$  ומחסרים את הרישא שלה ב-  $C$ .  
אם  $\epsilon$  קיימת ב-  $s_i$  או  $c$  קיימת ב-  $s_i$  עבור כל  $C$  in  $C$  אז הטקסט לא UD

### Information Content

המטרה: להציג את הדחיסה הטובה ביותר ביותר האפשרית, על ידי הסרת מידע מיותר.

- \* ככל שיש יותר "הפטעה" יש יותר אינפורמציה.  
"היום יום שימושי באילת" –> זה מאורע שכיח, אין פה הרבה אינפורמציה.  
"ירד שלג באילת" –> יש פה הפטעה גדולה, המון אינפורמציה.  
כל שהסתברות יותר נמוכה כקה הפטעה יותר גדולה.  
איך נוכל למדוד הפטעה/אינפורמציה בביטחון?

Shannon-1948:

שאנו חישב את כמות המידע של תו שמופיע בהסתברות  $p_i$   

$$I(s_i) = -\log_2 p_i$$

\* נרצה לבנות קוד שאורכה של כל מילת קוד בו היא כערך האינפורמציה.  
 $I(s_i) = -\log(p(s_i))$

$$I(s_i * s_j) = -\log(p(s_i * s_j)) = -\log(p(s_i) * p(s_j)) = -(\log(p(s_i)) + \log(p(s_j)))$$

$$H(P) = -\sum_{i=1}^n p_i \log_2(p_i) \quad H(C, P) = -\sum_{i=1}^n p_i I(s_i)$$

## סיכום קורס דחיסת נתונים א'

<https://github.com/shaynaor>

נכתב על ידי: שי נאור

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

Kraft–Mcmillan inequality

המטרה: ליצור קובץ כמה שיותר קצר אך עדיין UD.  
כמה קצר קוד יכול להיות כך שעדין יהיה UD?

- How short can codewords be so that the code is UD?
- Assume each symbol  $s_i$  has probability  $p_i = 2^{-k_i}$  then  $I(s_i) = k_i$  is a whole number. So setting each codeword to a string of  $|c_i| = k_i$  bits results in a code whose expected codeword length equals Shannon's bound.
- Theorem: Let  $C = [c_1, c_2, \dots, c_n]$  be a code with  $n$  codewords of lengths  $|C| = [|c_1|, |c_2|, \dots, |c_n|]$ . If  $C$  is UD then

$$K(C) = \sum_{i=1}^n 2^{-|c_i|} \leq 1$$

- Theorem: if  $K(C) = \sum_{i=1}^n 2^{-|c_i|} \leq 1$  for some code  $C$  then there exists another code  $C'$  such that
  - $E(C, P) = E(C', P)$
  - $|C'| = |C|$
  - $C'$  is prefix free.

### נוסחת Kraft

אם  $C$  הוא UD אז  $K(C) \leq 1$ .

כלומר, אם יצא לנו שהקraqט של  $C$  גדול מ 1 אז  $C$  בהכרח לא UD וגם לא prefix free. משפט: אם  $K(C) \leq 1$  אז קיימם קוד  $C'$ prefix free שמקיים:  
 $E(C', P) = E(C, P)$   
האורך של  $C$  ו  $C'$  טאג שווה.  
 $C$  טאג הוא prefix free.

## הרצאה 2 – modeling

מדוע כדאי לדחוס נתונים?

1. חוסר מקום.
2. מקטין את הקלט / פלאט.
3. קריפטוגרפיה (הצפנה).

### שיטות דחיסה

- Loseless: בדרך הזאת לא מאבדים מידע (עוסק יותר בקבצי טקסט).
- Lossy: בדרך הזאת ניתן לאבד מידע (עוסק בתמונות/אודיו/וידאו).

משפט: קוד הוא מיידי אם ורק אם הוא קוד חסר רישות.

הוכחה:

כיון 1: קוד מיידי הוא חסר רישות.

נניח בשילילה שקוד מיידי הוא לא קוד חסר רישות, כלומר, קיימת מילה שהוא רשא של מילת קוד אחרית, אבל זה אומר שהקוד הוא לא מיידי כי אם נתחיל עם הרישא לא נדע אם לעזר שם או להמשיך למילת הקוד האחרת.

כיון 2: קוד חסר רישות הוא קוד מיידי.

## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא, מחברת של שובל

קוד חסר רישות ניתן לייצג על ידי עצ. הפענו יתחל מהשורש, ונטיל בעז עד שנגיע לעלה. רק בעלים נמצאות מילות הקוד. ברגע שהגענו לעלה אנחנו כבר יודעים את הפענו ולכן הוא מייד.

משפט: קוד מיידי עם אורכי מילوت קוד  $|c| = \sum_{i=1}^n 2^{-k_i} \leq 1$  קיים אם ורק אם

דוגמה:

$$\begin{array}{lll} A = 11 & f_1 = 2 & |C| = \{2, 2, 3, 3, 4\} \\ B = 01 & f_2 = 2 & \\ C = 001 & f_3 = 3 & K(C) = 2 \cdot 2^{-2} + 2 \cdot 2^{-3} + 2^{-4} \\ D = 101 & f_4 = 3 & |K(C)| = \frac{1}{2} + \frac{1}{8} + \frac{1}{16} \leq 1 \\ E = 0001 & f_5 = 4 & \end{array}$$

הוכחה:

כיוון 1: קיים קוד מיידי עבור אורכי מילים מסוימות, ונראה כי הקראפט קטן מ 1. קיים קוד מיידי לאורכים מסוימים, ולכן קיים לו עצ. Z. נבחר את Z להיות עצ מושלם/שלם, אז מספר העלים בעז הוא  $2^{|Z|}$ . ברגע שבחרנו קוד באורך  $i$ , הוצאנו מכלל שימוש  $2^{i-1}$  עליים.

$$\begin{aligned} \text{(להלן)}: \quad \sum_{i=1}^n 2^{-f_i} &\leq 2^{-f_n} / : 2^{-f_n} \\ \sum_{i=1}^n 2^{-f_i} &\leq 1 \end{aligned}$$

הבחנה: קוד הוא שלם complete אם אין מילה שנייה להוסיף אותה לקוד  $\Rightarrow$  הקראפט שווה ל 1.

## Modeling

zero order: התווים הם בלתי תלויים, נסתכל על הסתברויות של כל אות בפני עצמה. ישנו שלושה סוגים עיקריים של מודלים:

- מערכת דחיסה סטטית: static compression system
- מערכת דחיסה סטטית למחצה: semi-static compression system
- דחיסה סטטית למחצה עם סתברויות עצמאיות: semi static self probabilities system

**מערכת דחיסה סטטית:** static compression system  
מניחים שהסתברויות הן בלתי תלויות ומיניהם שהסתברות היא אחידה.  
לדוגמה: לקוד ASCII יש 256 תווים, لكن בממוצע משתמש ב-8 סיביות.

$$P_i = 1/256$$

$$H(P) = -\sum_{i=1}^{256} (1/256) * \log_2(1/256) = 8.0$$

האנטרופיה נתנת לנו חסם תחתון לאורך מילת הקוד הממוצעת.

## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

### **מערכת דחיסה סטטית למחצה: semi-static compression system:**

- Example- character based model:

- **Message:**

```
Bring me my bow of burning gold!
Bring me my arrows of desire!
Bring me my spear! O clouds unfold!
Bring me my chariot of fire!
(William Blake 1978)
```

$$P_i = \frac{1}{25}$$

$$H(P) = -\sum_{i=1}^{25} \frac{1}{25} \log_2 \left( \frac{1}{25} \right) = 4.64$$

במודל זה לא נשתמש בכל 256 התווים של ASCII, אלא בעבר קודם על הטקסט ונבדוק באילו תווים משתמשים מתוך תוו ה ASCII. בשיטה זו ניתן בעצמנו לכל אות את מילת הקוד שלה, ואז נדרש לשЛОח למפענה את כל המידע זהה. גם פה נשתמש בתפלגות אחידה לתווים.

לכן לפי הדוגמה נדרש לבנות עץ בעומק 5. קלומר נוכל במקום לייצג כל תו באמצעות 8 סיביות כמו במערכת הקודמת כעת נוכל לייצג כל תו באמצעות 5 סיביות בלבד.

אבל כעת נוסף לנו HEADER נדרש תחיליה להעביר למפענה את כמות התווים, את התווים עצמם ואת הקידוד.

גודל הפיתיח:

8 ביטים : הודיע על כמות האותיות שאנו משתמשים. (בדוגמה 25)

8\*25 ביטים: 25 אותיות כפול 8 ביטים (האותיות עצמן ב ASCII)

$$\text{סך הכל נקבל: } 4.64 + (208/128) = 6.27$$

4.64 עבור האנתרופיה.

208 גודל התחלית.

128 מספר התווים בטקסט.

6.27 אורך מילה ממוצעת כולל הפיתיח.

קלומר ניתן לראות שבמודל הסטטי עבר כל תו נשתמש ב 8 סיביות בממוצע, ובמודל הסמי סטטי בדוגמה הזאת נשתמש ב 6.27 סיביות בממוצע עבור כל תו.

\* קיבלנו ש  $0.78 = (C)$  נשאף שהקראפט יהיה שווה ל 1, כי זה אומר שכל העלים בעץ מנוצלים.

### **דחיסה סטטית למחצה עם הסתברויות עצמאיות: semi static self probabilities system:**

במודל זה בנוסף לפיתיח, נחשב עבור כל תו את ההסתברות העצמית שלו לפי הנוסחה:  $p_i = v_i / m$

כלומר, ההסתברות של התו  $i$  היא כמה פעמים מופיע התו  $i$  חלקי כמות התווים בטקסט.

$$\text{עבור הדוגמה האחורונה נקבל כי } 4.22 = (C) \text{ לפי הנוסחה} \sum_{i=1}^n v_i \log_2 \frac{v_i}{m}$$

נשים לב כי זהה נוסחה שונה מהאנתרופיה. המספר שנתקבל

icut זה חסם תחתון על אורך ההודעה. ואנתרופיה זהה חסם תחתון לייצוג של תו בהודעה.

מבנה הפיתיח במודל זה:

8 ביטים: עבור כמות האותיות בהודעה.

8\*25: 25 אותיות כפול 8 ביטים ב ASCII (האותיות עצמן).

4\*25: לכל תו יש הסתברות מסוימת שמשולן שטייצג על ידי 4 ביטים.

סך הכל נקבל:

$$6.63 + (308/128) = 6.63$$

4.63 עבור האנתרופיה.

208 גודל התחלית.

128 מספר התווים בטקסט.

6.63 אורך המילה הממוצעת כולל הפיתיח.

$s_i$	$P_i$	$s_i$	$p_i$	$s_i$	$p_i$
'\n'	4/128	f	5/128	s	4/128
''	22/128	g	6/128	t	1/128
!	5/128	h	1/128	u	3/128
B	4/128	i	8/128	w	2/128
O	1/128	l	3/128	y	4/128
a	3/128	m	8/128		
b	2/128	n	7/128		
c	2/128	o	9/128		
d	4/128	p	1/128		
e	8/128	r	11/128		

$$E(C, P) = 4.26$$

$$K(C) = 1$$

## סיכום קורס דחיסת נתונים א'

<https://github.com/shaynaor>

נכתב על ידי: שי נאור

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

כלומר, קיבלנו תוצאה פחותה טוביה מהמודל הקודם!  
בדוגמה הזאת ההודעה מאד קטנה, ולכן לפטיח יש משקל גדול בקובץ הנדחים. אך התוצאה פחותה טוביה. אך שווה לנו להשתמש במודל זה רק בהודעות גדולות מספיק.

Minimum message length principle

החלילה כוללת שני מרכיבים:

- תאור המודל על ידי ה header.
- הקוד בהתאם לתאור.

כל שהמודל יהיה מורכב כך הפטיח שנשלח יהיה ארוך יותר.

### הרצאה 3 – static codes

**קוד אונרי** Unary code

....., 0, 10, 110, 1110

בעצם יש פה 0 מفرد בין כל TWO. כל מילה בקוד מקבלת רצף של אחדות ואפס בסוף שמודיע שהמילה הסטיימה. ניתן לשים לב שאנחנו יכולים ל"יעיל" קצת את הקוד, עברו א"ב סופי, ניתן להוריד את האפס ממייתת הקוד האחרון.

**קוד בינארי** Binary code

כל מילת קוד מתורח מילימ מקבלת  $[ch_2 \log_2]$  (ערך עליון) ביטים. ניתן לראות שם חזקה של 2, נוכל לנצל את מספר הביטים שנתקבל באופן מלא עבור ח המילים. במקרה ש ח לא חזקה של 2 נשאר עם "עודף" של ביטים לא מנוצלים.

לדוגמא:  $3 = [5 \log_2] = 5$

a-000

b-001

c-010

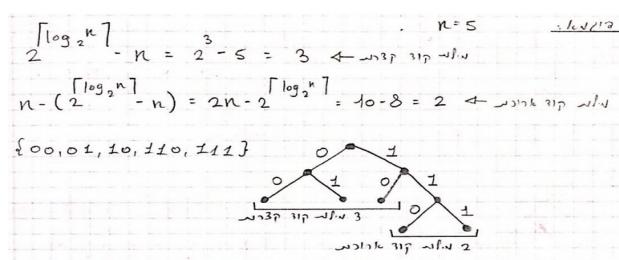
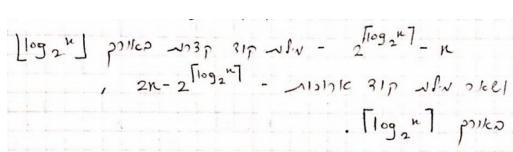
d-011

e-100

ניתן לוlutות שנשארנו עם שלוש מילים שלא נוצלו: {101, 110, 111}.

**קוד בינארי מינימאלי** Minimal binary code

עבור א"ב של A תווים, קוד בינארי מינימאלי מכיל



\* תמיד נמיין את מילויים את הקוד לפי השכיחיות שלחן.

## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא , מחברת של שובל

שאלה: متى הקוד הבינארי היה בעל יתרות מינימאלית ?

נרצה בעצם למצוא את ההסתברויות עבורם הקוד הבינארי המינימאלי יהיה גם אופטימלי .  
  $N = 2^k$ . נרצה שיתקיים:

$$\log(1/p_i) = |c_i| = k$$

מספר הביטים במלת קוד , הוא בדיקן כמו כמות האינפורמציה .  
 כדי לקבל את השוויון הנ"ל נוצרה שהסתברויות יהיו:  $p_i = 1/2^k = 1/(N+1)$ .  
 קלומר נוצר לקוד את כל המילים עם אותו מספר ביטים .

### Elias code

\* נלמד על שתי שיטות קידוד C גמה ו- C דלתא .

שיטה 1 : C גמה :

כל מלת קוד מורכבת מ 2 חלקים :

א. החלק הראשון אונרי , המספר האונרי מייצג את מספר הביטים של מלת הקוד כאשר הוא ב"צוג בינהר". זאת אומרת עבור  $x$  מספר קלשאנו נוצר לרשום את המספר  $[x_2 \log_2 x] + 1$  (ערך תחתון) באונרי .

ב. החלק השני בינהר, אך עם שינוי קל: ללא הביט הראשון. החלק זהה מייצג את המספר עצמו בינהר. נקבל  $[x_2 \log_2 x]$  ביטים (ערך תחתון).

סך הכל נקבל :

$[x_2 \log_2 x] + 1$  (ערך תחתון) עבור החלק האונרי .

$[x_2 \log_2 x]$  ביטים (ערך תחתון) עבור החלק הבינהר .

ב"יחד ,  $1 + [x_2 \log_2 x]$  .

x	Elias $C_\gamma$ Code
1	0
2	100
3	101
4	11000
5	11001
6	11010
7	11011
8	1110000
9	1110001

איך הקוד עובד בשרשור של הרבה מילים ?

מתחילה לקרוא את הקוד עד שנתקלים ב 0 הראשון .

לדוגמה אם קראנו  $A$  ביטים (בינהר) אז אנו יודעים שצריך לקרוא 1-A ביטים ולהוסיף 1 מוביל .

שיטה 2 – C דלתא :

היתרון בקוד זהה לעומת הקוד הקודם הוא גודל הקוד . הציג האונרי שראינו מצד שמאל הוא לא עיל . אפשר לדחוס גם אותו ולהשתמש ב C גמה . קוד זה משתמש ב C גמה באופן הבא: גם כאן אנו מרכיבים מלת קוד מ 2 חלקים , אך הפעם החלק הראשון הוא קידוד מספר הביטים בעזרת C גמה . והחלק השני נשאר קוד בינהר ללא הביט הראשון .

x	Elias $C_\delta$ Code
1	0
2	1000
3	1001
4	10100
5	10101
6	10110
7	10111
8	11000000
9	11000001

## **סיכום קורס דחיסת נתונים א'**

<https://github.com/shaynaor> נכתב על ידי: שי נאור

mbossum על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

ר' ס' נ'  $1 + 2 \lceil \log_2 \log_2 x \rceil$ :  $C_Y \rightarrow 2^{0.13}$  מילון גיבוב נורמליזציה (1)

$\lceil \log_2 x \rceil$ : מילון-can IDF שמיינן כפלי ב-2<sup>3</sup> ומעלה (2)

$1 + 2 \lceil \log_2 \log_2 x \rceil + \lceil \log_2 x \rceil \rightarrow 2^{0.13}$

$S = 53$  :  $1010101$

$C_Y (x=53) = \underline{\underline{110101}} \underline{\underline{10101}}$

$C_\delta (x=53) = ?$

$53 \rightarrow 110101 \rightarrow |53| \rightarrow 6 \text{ bit} // C_Y \rightarrow 6 \text{ bit}$

$6 \rightarrow 110 \rightarrow |6| \rightarrow 3 \text{ bit}$

$\text{unary}(3) \rightarrow 110$

$\text{binary}(6) \rightarrow 110 \rightarrow 10$

$C_\delta (y=53) = \underline{\underline{110101}} \underline{\underline{10101}}$   $\rightarrow$  מילון גיבוב נורמליזציה (3)

$\begin{cases} 11003 & \text{לפניהם} \\ 10101 & \text{אחריהם} \end{cases}$

$C_Y \rightarrow 6$  , ר' ס' נ' 5-5 , ר' ס' נ' -N

אריך דלתא עובד עם שרשור של מספר  
מילים?

1. מתחילה לקרוא את הקוד עד שנתקלים ב 0 הראשוני. במידה וקראנ 3 ביטים אני יודעים שעלינו לקרוא 2 ביטים נוספים.  
לאחר קריאת 2 הביטים, נוסף אחד מוביל
  - 2.

## קוד אוניברסלי Universal code

קוד אוניברסלי זה קוד שמספר הסיביות שלו הוא  $(\chi_2 \log(\chi_0))$ .

טענה:  $X = \{x_1, x_2, \dots, x_n\}$  גניחה שההסתברויות של הא"ב שולנו נתונות בצורה מונוטונית יורדת. גניחה בשליליה ש  $x_1 > x_2 > \dots > x_n$ .

שזה שווה ל 1, כלומר סכום ההסתברויות גדול בעצם מ 1.

$$I(S_x) = -\log_2 p_x \leq -\log_2(1/x) = \log_2 x$$

כלומר הכי טוב שאפשר להציג.

## Golomb code

בקוד Elias ראיינו שיש סדרה עולה של חזקות של 2. מבחינת חלוקת ה"DELIM".

<sup>2</sup>: דלי בגודל 1 עם ספרה 1 בחלק השמאלי.

<sup>1</sup> זל' בבודל 2 עם 2 ספרות בחילק השמאלי.

<sup>2</sup>: דלי בגודל 4 עם 3 ספרות בחלק השמאלי.

כל קבוצה כזאת נקראת דלי.

קוד גולומב אומר לנו שיהיה לנו מספר שונה בכל דלי.

1

### 5. דיל'ים עם ספרה 1 רוחק השמאלי

5. דליים עם ? ספרות בחלק השמאלי.

### 5. דליים עם 3 ספרות וחלק השמאלי.

וְאַתָּה תִּשְׁמַח

## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא, מחברת של שובל



### Golomb codes Algorithm:

```
Golomb_encode(x,b) {
    q←(x-1) div b;
    r←x-q·b;
    Unary_encode(q+1);
    Minimal_binary_encode(r,b);
}
```

### Golomb codes Algorithm:

```
Golomb_decode(b) {
    q←Unary_decode() -1;
    r←Minimal_binary_decode(b);
    return r+q·b;
}
```

דוגמה: עברו קידוד  $x = 5$

$x = 5, b = 5$

$$q = 4/5 = 0$$

$$r = 5 - 0 \cdot 5 = 5$$

$$\text{unary}(1) = 0$$

$\text{MBE}(5, 5) = 111$  (first index-position, second-num of code word)

the result is unary(1)MBE(5, 5), 0111

### rice code

ראינו בקורס גולומב שהחלוקת הימני של הקוד MBE הוא לא בגודל אחד. לכן בא ריס ומתקן את זה – גודל הדליים יהיה רק חזקות של 2. וכך אין צורך בשימוש ב-MBE.

האלגוריתם:

1. נבחר מספר  $X$ ,  $|X|$  קבוע עבר גודל הדלי  $2^k$ .

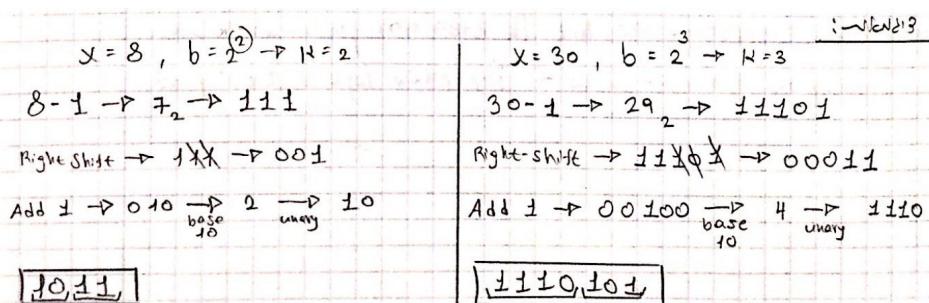
2. נחסיר  $1-X$ , ונחשב את מה שקיבלנו ביבנארי.

3. נבצע right-shift כמספר  $K$ . ונוסף 1.

4. את המספר שקיבלנו ביבנארי נמיר לעשורי, ונחשב באונרי.

5. המספר הסופי יהיה שרשרת של: חלק ראשון אונרי, חלק שני הביטים שמחקנו.

$x$	Golomb code $b=5$	Rice $K=2$
1	0 00	0 00
2	0 01	0 01
3	0 10	0 10
4	0 110	0 11
5	0 111	10 00
6	10 00	10 01
7	10 01	10 10
8	10 10	10 11
9	10 110	110 00



## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

### שיעור 4 Shannon Fano Coding –

נזכר תחילת בנוסחה לכמויות האינפורמציה עבור התו  $s_i$ :  

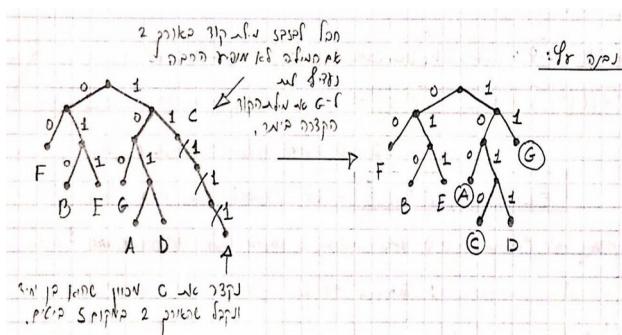
$$I(s_i) = -\log_2(1/p_i) = \log_2(p_i)$$

נאמר ויש לנו קוד  $C$  המכיל את מילות הקוד:  $c_1, c_2, \dots, c_n$   
 כך שאורך של מילת קוד  $c_i$  הוא:  $\lceil \log_2(1/p_i) \rceil$  (ערך עליון)  
 המטרה: למצוא קוד בעל יתירות מינימלית וחתוך ככמה שיותר לאנטרופיה, שהוא החתום  
 התחתום. אפשר לԶוזות שמדובר בקוד prefix-free אם מקבל שהקראפט הינו 1:

$$\begin{aligned} R(C) &= \sum_{i=1}^n 2^{-|c_i|} = \sum_{i=1}^n 2^{-\lceil \log_2 p_i \rceil} \leq \sum_{i=1}^n \frac{1}{2^{\lfloor \log_2 p_i \rfloor}} = \sum_{i=1}^n p_i = 1 \\ \log_2 \frac{1}{p_i} &\leq \lceil \log_2 \frac{1}{p_i} \rceil < \log_2 \frac{1}{p_i} + 1 \quad \therefore e^{H(P)} \\ H(P) &= -\sum_{i=1}^n p_i \log_2 p_i = \\ &= \sum_{i=1}^n p_i \log_2 \frac{1}{p_i} \leq \sum_{i=1}^n p_i \lceil \log_2 \frac{1}{p_i} \rceil < \sum_{i=1}^n p_i (\log_2 \frac{1}{p_i} + 1) \end{aligned}$$

קוד שננו הוא קוד שאורך מילת הקוד נמצאת בטוווח  $1 \leq L \leq H(P)$ .  
 זאת אומרת שיתכן וייה הפרש של סיבית אחת בין האינפורמציה לבין shannon.sh. כמובן שכאשר מוחזר בקוד ארוך הפרש זה מקבל מידדים גדולים.

דוגמה:



Symbol	probability	$\lceil \log_2 \frac{1}{p_i} \rceil$
A	0.1	4
B	0.15	3
C	0.05	5
D	0.09	4
E	0.14	3
F	0.27	2
G	0.2	3

$$H(P)=2.643$$

$$L = \sum_{i=1}^n p_i \cdot \lceil \log_2 \frac{1}{p_i} \rceil = 3.02$$

A better code -  $|C|=[4, 3, 4, 4, 3, 2, 3]$

## סיכום קורס דחיסת נתונים א'

<https://github.com/shaynaor>

נכתב על ידי: שי נאור

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא , מחברת של שובל

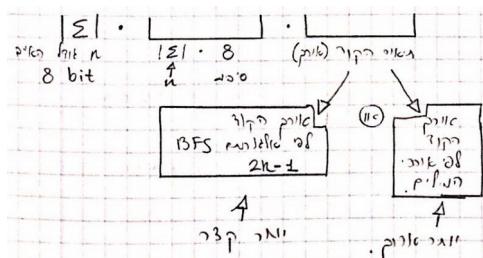
### Coding blocks of symbols

$$\begin{aligned}
 H(p(s_i, s_j)) &= -\sum_{(i,j)} p(s_i, s_j) \log_2 p(s_i, s_j) = \\
 &= -\sum_i \sum_j p(s_i) \cdot p(s_j) \log_2 (p(s_i) \cdot p(s_j)) = \\
 &= -\sum_i \sum_j p(s_i) \cdot p(s_j) (\log_2 p(s_i) + \log_2 p(s_j)) = \\
 &= -\sum_i p(s_i) \sum_j p(s_j) \cdot \log_2 p(s_j) - \sum_j p(s_j) \sum_i p(s_i) \cdot \log_2 p(s_i) = \\
 &= H(P(s)) \sum_i p(s_i) + H(P(s)) \sum_j p(s_j) = 2H(P(s))
 \end{aligned}$$

מדוע שאנו לוקח את הערך העליון של האינפורמציה  
 של כל TWO ?

כי הוא רצה להציג למספרים שלמים (כי כמות  
 האינפורמציה, הפונקציה לוג מחזירה גם שברים), אין  
 אפשרות להשתמש בשברי ביטים על מנת לקודד TWO  
 מסויים ? נkeh א"ב של זוגות שהם בעלי הסתברות בלתי-  
 תלויות ונחשב את האנטרופיה על הא"ב:

קיבלנו שהאנטרופיה שלנו קטנה גדלה פ' 2. אמנם הקידוד הולך ומתקרב לאנטרופיה אבל היא  
 ארוך יותר:



### Encoding a binary tree

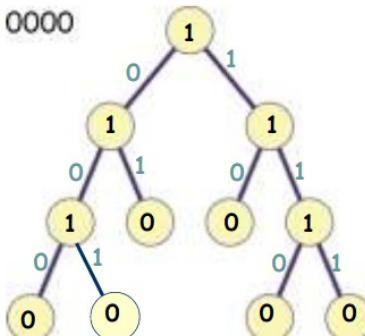
כדי להעביר את הקוד למפענה, ניתן להעביר את העץ המקודד למפענה במקום את כל  
 ההסתברויות.

לעבור על העץ באלגוריתם BFS:

- מסמן צומת פנימית ב 1.
- מסמן עלה ב 0.

לעץ עם N עליים יש  $N-1$  צמתים פנימיים, אז מספר הביטים הדרושים לקודד עץ בינהר' עם N עליים  
 הוא  $N-2$  סיביות.

- The binary string associated with the tree is: 11110010000
- There are  $n=6$  leaves and  $2n-1=11$  bits in the sequence



## **סיכום קורס דחיסת נתונים א'**

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא , מחברת של שובל

### **Shannon Fano Algorithm**

שאנו ופננו המציאו יחד אלגוריתם לבניית קוד תחילהות בעל יתרות מינימאלית המבוססת על קבוצה של סמלים והתדריות שלהם מופיעים.

האלגוריתם:

1. עבור רשימה נתונה של סמלים, יש לחת לכל סמל את התדריות שלו.
2. מין את רשימת הסמלים לפי ההסתברות שלהם בסדר יורד.
3. פצל את הרשימה לשני חלקים כך שהסתברויות של שני החלקים תהיה שווה עד כמה שניתן.
4. החלק השמאלי של הרשימה מקבל את הספרה "0", והימני מקבל את הספרה "1".
5. חוזר באופן רקורסיבי על שלבים 3,4 עבור כל אחד משני החלקים, עד שכל חלק יוכלתו אחד בלבד.

Prob.	0.67	0.11	0.07	0.06	0.05	0.04
	0			1		
		0			1	
		0	1	0		1
				0	1	
Code	0	100	101	110	1110	1111

Prob.	0.67	0.11	0.07	0.06	0.05	0.04
	0			1		
		0			1	
		0	1	0		1
				0	1	
Code	0	100	101	110	1110	1111

נשים לב שקוד זה הוא לא תמיד הכי אופטימלי, לדוגמה:

### **שיעור 5 – Huffman**

בניית העץ הבינארי של הקוד "מלמטה למעלה". ראשית, נמצא את 2 האותיות בעלות ההסתברות המינימאלית, וניצור צומת חדש, כך ששתי האותיות הללו יהיו בניו. נחזיר על התהילן עד שנישאר עם צומת אחד בלבד – שורש העץ. ענף ימין יסומן ב 0, ענף שמאל יסומן ב 1. הפענוח יבוצע על ידי מעבר על העץ מהשורש ועד לכל עלה.

## סיכום קורס דחיסת נתונים א'

<https://github.com/shaynaor>

נכתב על ידי: שי נאור

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא , מחברת של שובל

אלגוריתם:

אלגוריתם של אפמן רץ בסיבוכיות  $O(n \log n)$  כאשר  $n$  - מספר התווים.  
שלב א: נכנסי את כל התווים לתוך עדיפויות  $Q$ .

שלב ב: נכנס לשלב 1-ח פעמים:

- נוצר צומת חדש  $z$ , שני בני הימני.
- נוציא את האיבר הראשון מ  $Q$  ונכנסו אותו לבן השמאלי של  $z$ .
- נוציא איבר נוסף מ  $Q$  ונכנסו אותו לבן הימני של  $z$ .
- נגדיר את השכיחות של  $z$  להיות חיבור השכיחויות של שני בניו.
- נכנס ל  $Q$  את  $z$  המעודכן.

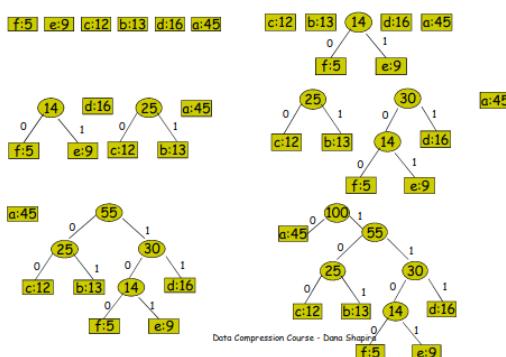
**HUFFMAN ( $\Sigma$ )**

```

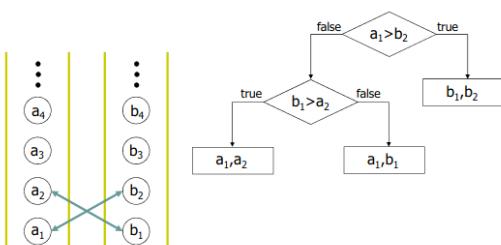
1  $n \leftarrow |\Sigma|$ 
2  $Q \leftarrow \Sigma$ 
3 for  $i \leftarrow 1$  to  $n - 1$ 
4   do ALLOCATE-NODE ( $z$ )
5     left[ $z$ ]  $\leftarrow x \leftarrow EXTRACT-MIN(Q)$ 
6     right[ $z$ ]  $\leftarrow y \leftarrow EXTRACT-MIN(Q)$ 
7     w[ $z$ ]  $\leftarrow w[x] + w[y]$ 
8     INSERT( $Q$ ,  $z$ )
9 return EXTRACT-MIN( $Q$ )

```

What is



דוגמת הריצה:



\* במידה והשכיחות נתונות בצורה ממוקנת, ניתן לבצע את האלגוריתם בצורה יעליה יותר. משתמש ב 2 תורים רגילים. נכנס את התווים בסדר מונוטוני יורד לתוך התור הראשי. בכל פעם נוציא מכל תור שתי איברים קטנים ביותר ואת הסכום שלהם נכניס בסוף התור השני. ממשיכים עד שהטור הראשי ריק, ובטור השני איבר אחד.

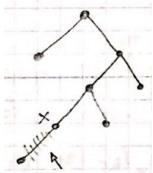
## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

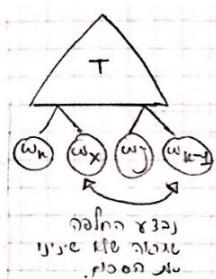
**משפט:** בהינתן משקלות (הסתברויות/שכיחיות)  $w_1, w_2, \dots, w_n$ , אלגוריתם האפמן נותן קוד באורךים  $|w_1|, |w_2|, \dots, |w_n|$ , כך ש  $\sum_{i=1}^n w_i \cdot |w_i|$  מינימלי.

**עוז אופטימלי:** עוז עברו קוד אופטימלי.



למה 1: עוז אופטימלי הוא עוז מלא (לכל צומת 0 או 2 ילדים).  
הוכחה: נניח ביליה שהעוז אינו מלא -  $\leftarrow$  קיימים צומת  $X$  עם בן יחיד. נשים מכך מילויים הקוד המשורשת מ- $X$  את הסיבית המתאימה ונקבל קוד  $C$  עם  $(P, C)$  קטן יותר, בסתריה לאופטימליות הקוד המקורי.

למה 2: בעוז אופטימלי המשקלים ה"זולים" ביותר  $w_1, \dots, w_n$  נמצאים בrama התחתונה.  
הוכחה: נניח בשילילה ש  $w_n$  או  $w_{n-1}$  אינם נמצאים בramaeki התחתונה בעוז. נניח ללא הגבלת הכלליות שזהו  $w_n$ , ככלומר יש איזשהו  $X$  כך ש  $w_X$  נמצא בramaeki התחתונה.  $w_X < w_n$ ,  $|w_X| > |w_n|$ . נבנה עוז חדש שבו נחליף את  $w_n$  ב  $w_X$  ונראה שהעוז החדש שנתקבל אופטימלי יותר.  
 $w_X - w_n > 0$   
 $|w_X| - |w_n| > 0$   
 $(w_X - w_n)(|w_X| - |w_n|) > 0$



למה 3: בעוז אופטימלי  $w_1, \dots, w_n$  יכולים להיות אחים.  
הוכחה: אם הם אחים, סימנו.  
נניח שם לא אחים, שניהם נמצאים בramaeki התחתונה ולכל אחד מהם יש אח  $w_X$ .  
כל האחרים נמצאים בramaeki התחתונה של העוז -  $\leftarrow w_{n-1} = w_n = w_X$ . מכאן שגם נבצע החלפה לא תיגע אופטימליות העוז.

**משפט אופטימליות האפמן**  
בhinatan משקלות  $w_1, \dots, w_n$  אלגוריתם האפמן יוצר קוד בעל אורכים  $|w_1|, |w_2|, \dots, |w_n|$ , כך ש  $\sum_{i=1}^n w_i \cdot |w_i|$  מינימלי.  
הוכחה: באינדוקציה על  $n$  (מספר העלים, מילויים הקוד).

$$\begin{aligned} & \text{בhinatan משקלות } w_1, \dots, w_n \text{ (סדרה מונוטונית יורדת) אלגוריתם האפמן יוצר קוד בעל מילויים } |w_1|, |w_2|, \dots, |w_n|. \text{ והוא שווה ל } \sum_{i=1}^n w_i \cdot |w_i|. \\ & \text{במקרה } n=2, \text{ נסמן } T_1 \text{ כעוזו של } w_1 \text{ ו } T_2 \text{ כעוזו של } w_2. \text{ נוכיח כי } T_1 + T_2 \text{ מינימלי.} \\ & \text{במקרה } n=3, \text{ נסמן } T_1 \text{ כעוזו של } w_1, T_2 \text{ כעוזו של } w_2, \text{ ו } T_3 \text{ כעוזו של } w_3. \text{ נוכיח כי } T_1 + T_2 + T_3 \text{ מינימלי.} \\ & \text{במקרה } n=k+1, \text{ נסמן } T_1 \text{ כעוזו של } w_1, \dots, w_k, \text{ ו } T_2 \text{ כעוזו של } w_{k+1}. \text{ נוכיח כי } T_1 + T_2 \text{ מינימלי.} \\ & \text{במקרה } n=k, \text{ נסמן } T_1 \text{ כעוזו של } w_1, \dots, w_{k-1}, \text{ ו } T_2 \text{ כעוזו של } w_k. \text{ נוכיח כי } T_1 + T_2 \text{ מינימלי.} \\ & \text{במקרה } n=1, \text{ נסמן } T_1 \text{ כעוזו של } w_1. \text{ נוכיח כי } T_1 \text{ מינימלי.} \end{aligned}$$

## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא, מחברת של שובל

לעתה נוכיח  $M_2 \leq M_1$  ו-  $T_2 \leq T_1$  לפי  $w_i$ .

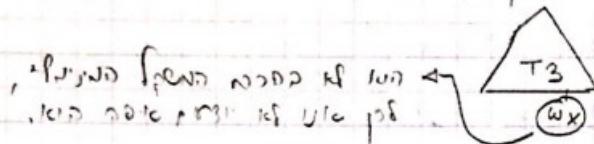
$$M_1 = M_2 + (\omega_n + \omega_{n-1}) \quad \text{⊗}$$

כדי להוכיח  $M_2 \leq M_1$  כוריאנטה ראנר מושג הוכח:

הוכיח  $\varphi_T$  הינו  $\varphi_T \leq \varphi_{T_2}$  ו-  $T_2 \leq T_1$ .

$\omega_1, \omega_2, \dots, \omega_{n-2}, \omega_n = \omega_{n-1} + \omega_n$

הוכחה: נוכיח  $M_2 \leq M_1$ ,  $T_2 \leq T_1$  ו-  $\varphi_T \leq \varphi_{T_2}$

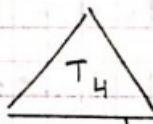


$\varphi_{T_2} \leq \varphi_T$

$w_x - f_1$  נוכיח  $w_x - f_1 \leq w_n$  בדרכם של צעדים

⊗⊗⊗  $\varphi_T \leq \varphi_{T_2}$

$$M_2 = M_1 + (\omega_n + \omega_{n-1}) \leq M_1 + (\omega_n + \omega_{n-1})$$



$T_4 \leq T_1$

$$= M_1 \Rightarrow M_2 \leq M_1$$

⊗⊗⊗  $\varphi_T \leq \varphi_{T_2}$

$T_1 \leq T_2$ , מוכיח  $\varphi_T \leq \varphi_{T_2}$  -  $T_2 \leq T_3$ ,  $T_3 \leq T_1$  ו-  $T_2 \leq T_1$ .

בנוסף  $\varphi_T \leq \varphi_{T_2}$  ו-  $\varphi_{T_2} \leq \varphi_{T_1}$ .

$\varphi_T \leq \varphi_{T_1}$  כוריאנטה  $\varphi_T \leq \varphi_{T_2}$  ו-  $\varphi_{T_2} \leq \varphi_{T_1}$ .

כוריאנטה  $\varphi_T \leq \varphi_{T_1}$  ו-  $\varphi_{T_1} \leq \varphi_{T_2}$ .

האנו נוכיח  $T_2 \leq T_1$  ו-  $T_1 \leq T_2$  ו-  $T_2 \leq T_1$ .

בנוסף  $\varphi_T \leq \varphi_{T_1}$  ו-  $\varphi_{T_1} \leq \varphi_{T_2}$ .

\* בעז האפמן ניתן לשנות בין 0 ו- 1, נקבל  $2^{n-1}$  אפשרויות לבנות עץ האפמן.

\* במצב של משקלים זהים נוכל להחליף ביניהם וכך לקבל עוד עצים שונים.

## סיכום קורס דחיסת נתונים א'

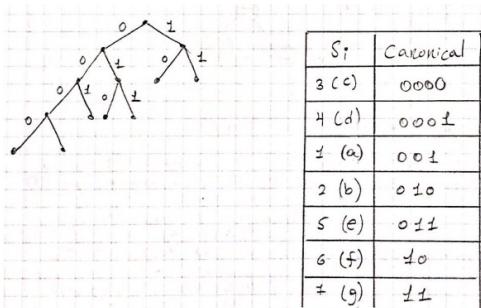
<https://github.com/shaynaor>

נכתב על ידי: שי נאור

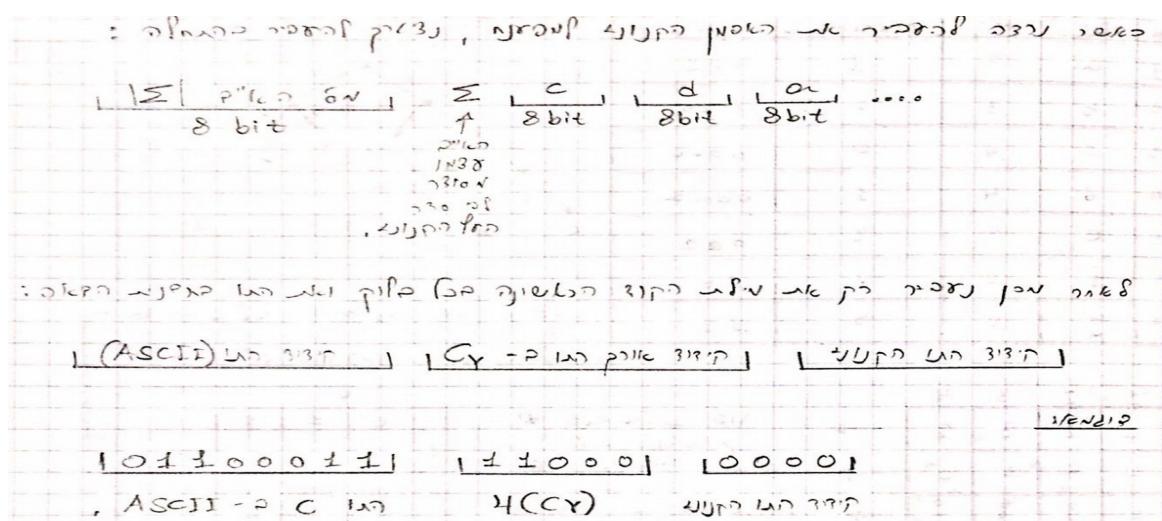
מבוסס על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

### עץ האפמן קנווי

אנו יודעים שסדרת משקלים מסוימת ניתנת לקבל יותר מעץ אחד. נרצה למצוא דרך שבה נוכל לקבל עץ אחד ויחיד במקומם מספר עצים אפשריים עבור סדרת משקלים נתונה. זה חשוב מכיוון שכך המפענה יוכל לבנות את אותו העץ בצד שלישי. (באמצעות בניית האפמן קנווי נוכל להקthin את ה prelude ).



קוד קנווי: עץ המיצג קוד המתחילה עם מילות הארוכות בצד שמאל, ומילות הקוד הולכות ומתקררות, העץ מסודר לפי אורכי מילות הקוד. המילה הארוכה ביותר כולה אפסים, והארוכים יתקרו כבסירה מונוטונית יורדת. נקבל בלוקים של אורכי מילות הקוד, כך שבכל בלוק המספרים הם מספרים עוקבים.



### אלגוריתם למעבר בין קוד האפמן רגיל לקוד האפמן קנווי:

- Given the symbols  $\{i\}$  and optimal lengths  $\{l_i\}$

```
maxlength←maxi{li}  
1. // find the number of codewords of each length  
for l←1 to maxlength  
    num[l]←0  
for i←1 to n  
    num[l]++  
2. // Store the first codeword  
firstcode[maxlength]←0  
for l←maxlength-1 downto 1  
    firstcode[l]←(firstcode[l+1]+num[l+1])/2  
3. for l←1 to maxlength  
    nextcode[l]←firstcode[l]  
4. for i←1 to n  
    codeword[i]←nextcode[i]  
    symbol[i],nextcode[i]-firstcode[i]←i  
    nextcode[i]++
```

הקלט: התווים  $\{i\}$ , ואורכי מילות הקוד  $\{l_i\}$ . ראשית נשמר את אורכי הקוד המלאים.

1. נרצה לחשב את מספר מילות הקוד באותו אורך. נעבור על סדרת הארכומים, וכל פעם שנתקל באורך מסוים, נקדם אותו במערך num.

2. תמיד מילת הקוד הארוכה ביותר תהיה 0 (בעשרוני), לכן נאתחל אותן הכר. נróż מהאיבר הלפני אחרון עד לאיבר הראשון ונملא לפי הנוסחה.

3. ניצור מערך עזר נוסף המציין את המספר שיש לכתוב באותו מאורך מסוים.

## **סיכום קורס דחיסת נתונים א'**

<https://github.com/shaynaor>

## נכתב על ידי: שי נאור

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

פערנשטיין קנווי

```
1. v ← nextInputBit()
   l ← 1
2. while v < firstcode[l] do
   v ← 2v + nextInputBit()
   l++
3. // the integer v is a valid codeword of l bits
4. return symbol[l, v-firstcode[l]]
// the index of the decoded symbol
```

ניצור משותנה 7 המקבל את הבית הראשון.  
7 יחזק את הערך המציגו.  
המשותנה | יאמר כמה ביטים במיליה.

שיעור 6 – Huffman

# האפטן D-ary

עד עכשו דיברנו על האפמן בינהרי, שלכל צומת יש 2 בניים. האפמן ary-S הוא הכללה של האפמן עברו S כלשהו.

**משפט:** בעץ  $D$ -ary מלא עם  $K$  צמתים פנימיים, יש  $(D-1+K^*)$  עליים.

- בעז אופטימלי (בינארי או ary-C):
  - לפחות 2 עליים בرمה התחתונה.
  - אוטם צמתים עם משקלות נומכיב.

נניח ונרצה לבנות עץ עבור  $3 = M$ , ונניח שיש לנו  $8 = n$  מילוט קוד. ננסה לבנות עץ עם 8 עלים. נקבל עץ מלא עם 9 עלים.

במקרה זה תלמיד נקבע מספר אי זוגי של עליים 1+2.

הפתרון הוא לזרור על צומת אחד ברמה התחתונה. נוכל להוסיף צומת פיקטיבית עם משקל 0 במקום הצומת שיתרנו עליה על מנת להציג לעצם האפמן אופטימלי.

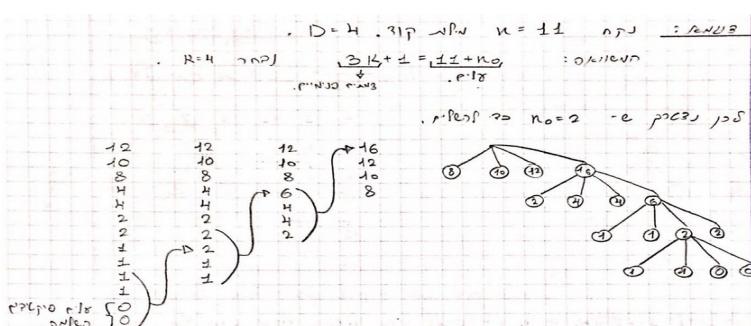
### אלגוריתם האפמן:D-ary

1. נשלים את המספר האמתי שנרצה להגע אליו עם צמתים בשכיחות 0 ונגיע לכך לעצם.

2. נאחד את סמלות הקוד עם  
הסתברויות הנמוכות ביותר.

3. נסמן את הרצף D-ary (אם  $D=3$ ) על ענפי העץ.  
אז:  $(0, 1, 2)$

4. נצטרך בנוספַּה להעביר את ה-D-ary (אם 3 = D אז 0, 1, 2: ) לבינארִי.



## סיכום קורס דחיסת נתונים א'

<https://github.com/shaynaor>

נכתב על ידי: שי נאור

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

### האפמן דינמי – Dynamic Huffman

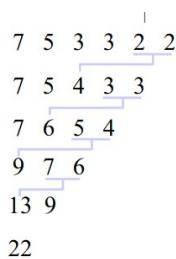
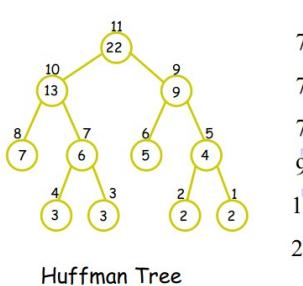
המטרה: לעבור פעם אחת על קובץ הטקסט ולקודד את התווים א' באמצעות כל התווים הקודמים לו. הערה: לא נרצה בכל פעם ליצור עץ האפמן חדש, אלא לעדכן את העץ על סמך התו הנוכחי והתוים הקודמים.

#### תכונת האחים – sibling property

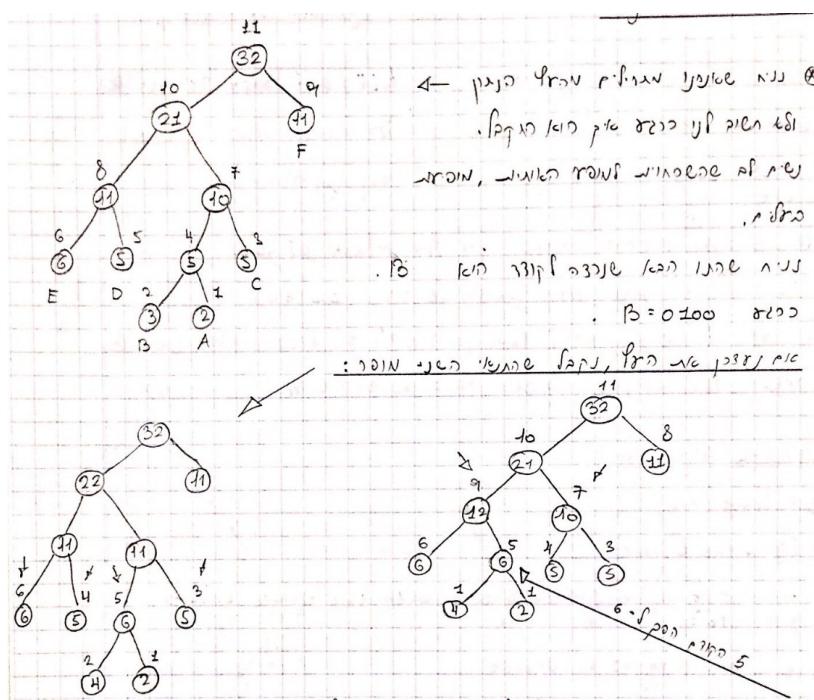
נשתמש בתוכנה זו על מנת לבנות את האפמן דינמי.

- המשקל של צומת 7 יהיה סכום המשקלים של הבנים שלו.
- הצמתים ימוספרו בסדר עולה, כך שמספר שני אחיהם הם מספרים עשרוניים עוקבים. ( 1-2 גז 2 המ אחיהם ).

עż המקיים את תכונת האחים אם ורק אם העץ יכול להיות האפמן.



עż שנוצר על ידי אלגוריתם האפן יקיים את תכונת האחים. לעומת זאת ניתן ליצור עץ אופטימלי אחד קלשחו רק שנתקח את 2 ו- 3 להיות המינימליים (העץ עדין אופטימלי כי עדין האורכים אותו הדבר).



דוגמא:

\* נעביר את הצמתים 2 | 4 להיות תחת 5 כדי לפתור את הבעיה, נשים לב שעדיין העץ יהיה אופטימלי ולכן נוכל להעביר אותם אבל כעת עדין תכונת האחים מופרת.

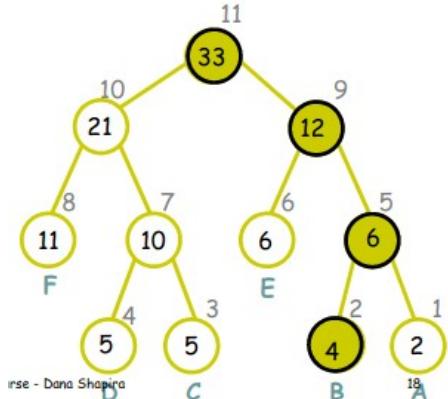
\* הבעיה נוצרת כאשר יש משקל זהה, ואני לא עוברים באינדקס המשקל של אותו המשקל, لكن במקרה זה נחליף בין שתי העצים כך שהתנאים יישמרו.

## סיכום קורס דחיסת נתונים א'

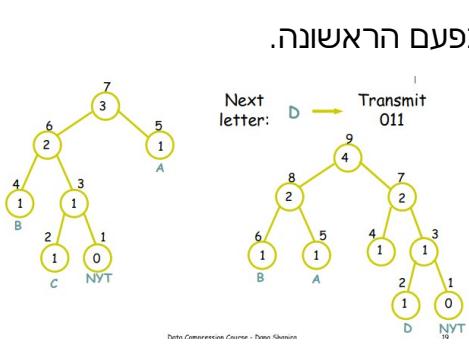
<https://github.com/shaynaor>

נכתב על ידי: שי נאור

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא, מחברת של שובל



icut 110 = B קיצרנו את המילה ל 3 ביטים.



ישנו מקרה נוסף בו נדרש לטפל, המקרה בו נתקלים באות בפעם הראשונה.

מצב 1: כאשר מתחילה מעץ ריק, נגדיר צומת (not yet transmitted) (NYT) הצומת ריק. נעביר את הקוד האסקי של האות ASCII(X) 8 bit.

מצב 2: כאשר לא מתחילה מעץ ריק, ומעברים בפעם הראשונה את הקידוד של NYT, ואת הקוד האסקי של התו.

\* בשני המקרים נגידל שני ילדיים: בן שמאל' לתו החדש ובן ימני לNYT.

אלגוריתם העדכון של העץ:

```

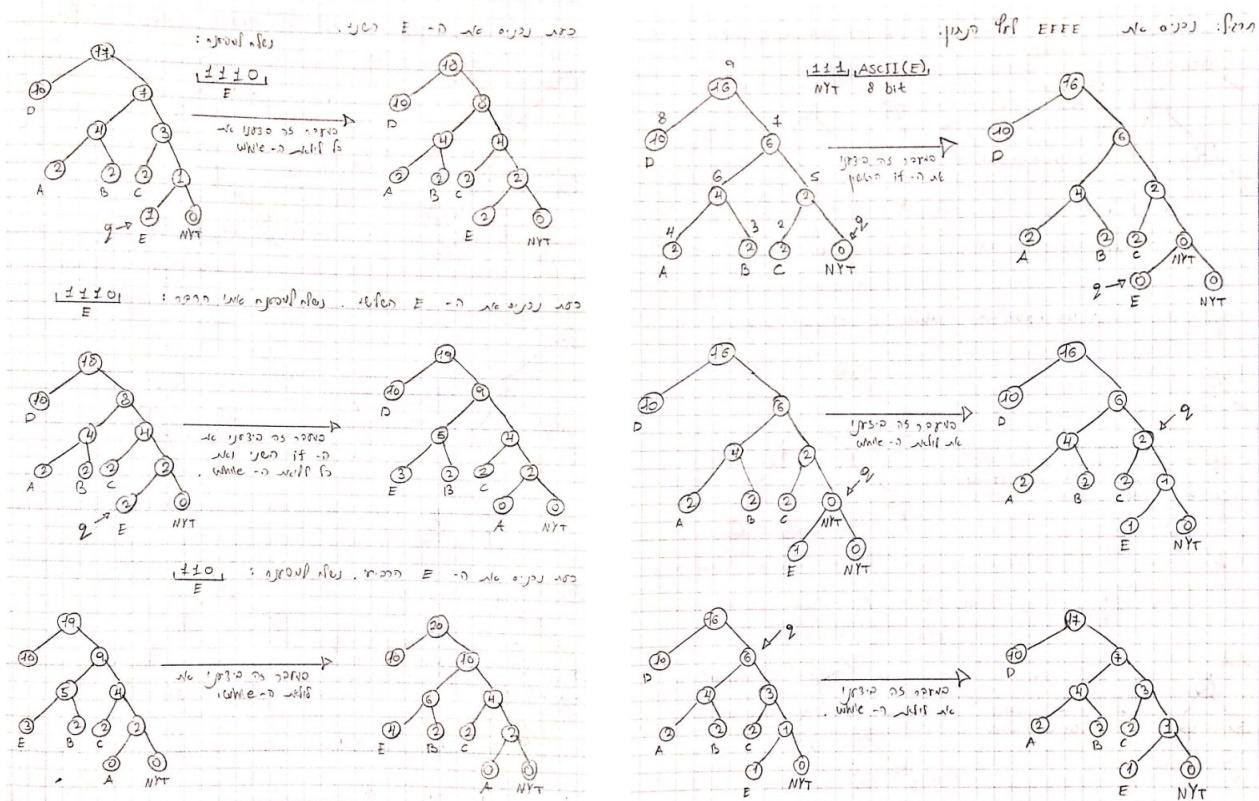
q = leaf(x_t)
if (q is the 0-node)
    replace q by a parent 0-node with two 0-node children;
    q = left child;
if q is a sibling of a 0-node
    interchange q with the highest numbered leaf of the
    same weight;
    increment q's weight by 1;
    q = parent of q
while q ≠ root
    interchange q with the highest numbered node of the
    same weight;
    increment q's weight by 1;
    q = parent of q
increment q's weight by 1;

```

## סיכום קורס דחיסת נתונים א'

<https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל



### הרצאה 7 – Skeleton trees

המעבר בין כל בלוק הוא להוסף 1 ולהכפיל ב 2 (=לשרשר 0). זה נובע מהמספרים הבינאריים העוקבים בכל בלוק. המוטיבציה היא שללא צריך לזכור את כל העץ, אלא נכון לדעת לפ' בלוקים. נקח עץ האפמן קוני ונקוץ תתי עצים שלמים שהעומק שלהם גדול מ 1. צריך לשמור על הסדר של הא'ב כדי שנדע להתאים קוד לתו. העליים ישמרו את האורך של מילת הקוד המקודדת.

הגדרות:

- ח- מספר מילوت הקוד באורך  $i$ .
- ח- נרצה לדעת מאייה להתחילה, لكن נסמן את  $w$  בתור מילת הקיצור ביותר, ונניח שאין לנו חורים (מילה באורך 4 ולאחר מכן מילה באורך 5 וכו...)

$m = \min\{i | n_i > 0\}$  –> הבלוק ההתחלתי.

– מילת הקוד הראשונה בכל בלוק שאותה נתרגם למספר עשרוני.

–  $\text{base}(m)$  – תמ"ד 0.

$\text{base}(i) - 2(\text{base}(i-1) + n_{i-1})$

- $B_s(k)$  - the  $s$ -bit binary representation of the integer  $k$  with leading zeros if necessary.
- The  $j^{\text{th}}$  codeword of length  $i$  for  $j = 0, 1, \dots, n_i - 1$  is  $B_j(\text{base}(i) + j)$
- $\text{seq}(i)$  - the sequential index of the first codeword of length  $i$
- $\text{seq}(m) = 0$
- $\text{seq}(i) = \text{seq}(i - 1) + n_{i-1}$
- $I(w)$  - the integer value of the binary string  $w$ , i.e. if  $w$  is of length  $\ell$ ,  $w = B_\ell(I(w))$ .
- $I(w) - \text{base}(\ell)$  is the relative index of codeword  $w$  within the block of codewords of length  $\ell$ .
- $\text{seq}(\ell) + I(w) - \text{base}(\ell)$  is the relative index of  $w$  within the full list of codewords.
  - This can be rewritten as  $I(w) - \text{diff}(\ell)$ , for  $\text{diff}(\ell) = \text{base}(\ell) - \text{seq}(\ell)$ .
  - Thus all one needs is the list of  $\text{diff}(\ell)$ .

## סיכום קורס דחיסת נתונים א'

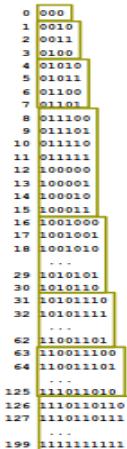
<https://github.com/shaynaor>

נכתב על ידי: שי נאור

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא, מחברת של שובל

### An example for values

$\ell$	$n_\ell$	$base(\ell)$	$seq(\ell)$	$diff(\ell)$
3	1	0	0	0
4	3	2	1	1
5	4	10	4	6
6	8	28	8	20
7	15	72	16	56
8	32	174	31	143
9	63	412	63	349
10	74	950	126	824



אלגוריתם הפענה:

```

1. tree_pointer<-root
2. i<-1
3. start<-1
4. while i<length_of_string
   1. if string[i]=0
      tree_pointer<-left(tree_pointer)
   2. else tree_pointer<-right(tree_pointer)
   3. if value(tree_pointer)>0
      codeword<-string[start...(start+value(tree_pointer)-1)]
      output<-table[I(codeword)-diff[value(tree_pointer)]]
      tree_pointer<-root
      start<-start+value(tree_pointer)
   5. i<-start
   4. else i++

```

0 = (v) אם מדובר בזאת פנימי של העץ.  
אם v הוא עלה, האורך בביטים של מילת הקוד  
הנוכחית.

(j) עברו ציון 1 ל n – טבלה עם הא"ב  
מסודר לפי העלים.

Length of string – המחרוזת הדוחסה.  
\* נעבור סיבית סיבית כל עוד אנו נמצאים בזאת  
פנימית. ברגע שנגיע לעלה אנו יודעים כמה  
סיביות נוספות נצטרך לקרוא בבת אחת כדי  
לעוזת זאת מהר.

\* כל עוד לא סיימנו לעבור על המחרוזת הדוחסה:  
– אם הביט הבא בתור הוא 0 –> נלך שמאליה.  
– אחרת –> נלך ימינה.

– אם 0 <(v), זה אומר שהגענו לעלה:

–> נכניס את מספר הסיביות של העלה בבת אחת ל codeword.  
–> נחשב את האינדקס של המילה ובעזרת table נכניס את הפלט ל משתנה output.

### Reduced skelton tree

- tree\_pointer<-root
- i<-start<-1
- while i<length\_of\_string
  - if string[i]=0 tree\_pointer<-left(tree\_pointer)
  - else tree\_pointer<-right(tree\_pointer)
  - if value(tree\_pointer)>0
    - len<-value(tree\_pointer)
    - codeword<-string[start...(start+len-1)]
    - if flag(tree\_pointer)=1 and 2\*I(codeword)>=base(len+1)
      - codeword<-string[start...(start+len)]
      - len++
    - output<-table[I(codeword)-diff[len]]
    - tree\_pointer<-root
    - i<-start<-start+len
  - else i++

קידוד נוסף עושים ל skelton , המטרה שלנו  
היא לעשות פענוח מהיר. אם הזאת היא שורש של  
תתי עצים שה הפרש בין מלית הקוד הקצרה ביותר  
למלית הקוד הארוכה ביותר הוא לפחות 1 הוא  
"הה עלה ב reduced ."

מגדירים עברו כל צומת v :  
אם הוא עלה (v) = upper(v) = value(v) – lower(v)  
אחרת (v) = lower(left(v)) –> שמאליה.  
(v) = lower(right(v)) –> ימינו.

## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא, מחברת של שובל

### הרצאה 8 – קוד אריתמטי

קוד האפמן:

- התאמנו לתווים של הא"ב מילות קוד באמצעות הסתברויות.
- המעביר לקוד האפמן דינמי היה קשה ומסובבל. השימוש ב sibling property והרוטציות הם פועלות כבדות.
- היה צריך לאחסן את הקוד בדרך כלשהי, לדוגמה בצורה טבלה שתמפה את מילות הקוד לקוד עצמו.
- חסרונו – אורך מילת הקוד המינימלית הוא בית אחד. לא יוכל לקודד מילת קוד בפחות מביט אחד. לכן עשוי להיות זבוז שמצויבר.
- לכל היוטר נהיה רוחקים בביט אחד מהאנטロפייה. אם יש לנו קובץ עם מיליון תוים שאותו נרצה לקודד, זה אומר שאנחנו עלולים להיות רוחקים (במקרה הגראן) במליאן סיביות מהדחיסה האופטימלית.

קוד אריתמטי:

- מה שנעשה זה שנחליף את הקלט שלנו במספר float בין 0 ל 1.
- נדרש הסתברות מסוימת כדי להתחיל אליה, (בזמן נלמד אלגוריתם אדפטיבי), ולאט לאט נלמד את הסתברויות תוך כדי המעביר על הקובץ.
- יהיה קל יותר לעבוד עם הגרסה האדפטיבית, אבל נתחיל עם הגרסה הսטטיסטית.
- אין צורך בטלה שתתרגם לנו את מילות הקוד, מה שהמקודד יעשה גם המפענח עשויה.
- נשתמש במספר לא שלם של ביטים למילת קוד.

character	probability	Huffman Code
A	0.95	0
B	0.02	11
C	0.03	10

דוגמה: נניח שהא"ב שלנו בני שלושה תוים בסה"כ, ונניח שההסתברות של האות A מאד גדולה. לאחר הרצת האפמן נקבל – <

נחשב את האנטロפייה ונקבל 0.335 bps

אורך מילת הקוד הממוצע המתקבל מהתוצאות מהאפמן 1.05 bps.

ההפרש מהאנטロפייה גדול!

### Reduced interval

שיטה שבאמצעותה נוכל להתקרב יותר לאנטרопיה.

- נפח אינטראול עברו הסתברות של תו בתווך  $[0, 1]$ .

נחלק את האינטראול בצורה פופוציונלית

לחתני אינטראולים בהתאם להסתברויות של התווים. נדרש לדעת מה הסדר של התווים, תת-האינטראולים הם באופן ייחסי להסתברות של התו.

$s_i$	$p_i$	low_bound	high_bound
A	0.67	0.0	0.67
B	0.11	0.67	0.78
C	0.07	0.78	0.85
D	0.06	0.85	0.91
E	0.05	0.91	0.96
F	0.04	0.96	1.0

$low\_bound(s_i) \leftarrow \sum_{j=1}^{i-1} p_j$   
 $high\_bound(s_i) \leftarrow \sum_{j=1}^i p_j$

נצמצם כל פעם את תת האינטראול בהתאם להסתברות ורק נמשיך עבור עליתו.

## **סיכום קורס דח'יסט נתוניים א'**

<https://github.com/shaynaor> נכתב על ידי: שי נאור

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא . מחברת של שובל

הסתברות של A היא 0.67 וכאן נבחר לחת ל-0 את תת האינטראול זהה. נוסף ל-0.67 את הסתברות של B, וזה יהיה האינטראול שלו וכך הלאה..

הערה: אין נקודה שהיא משותפת ליותר מאנטרכז אחד.

```

low    ← 0.0
high   ← 1.0
while input symbols remain{
    range ← high - low
    Get symbol
    high ← low + high_bound(symbol)*range
    low  ← low + low_bound(symbol)*range
}
Output any value in [low, high)

```

M[i]	Low	High	Range
-	0.0000	1.0000	1.0000
A	0.0000	0.67	0.67
B	0.4489	0.5226	0.0737
A	0.4489	0.498279	0.049379
A	0.4489	0.48198393	0.03308393
A	0.4489	0.47106623	0.02216623
E	0.46907127	0.47017958	0.00110831
A	0.46907127	0.46981384	0.00074257
A	0.46907127	0.46956879	0.00049752
B	0.46940461	0.46945934	0.00005473
A	0.46940461	0.46944128	0.00003667

$S_i$	Low bound	High bound
A	0.0	0.67
B	0.67	0.78
C	0.78	0.85
D	0.85	0.91
E	0.91	0.96
F	0.96	1.0

## Binary representation

- נשים לב שה `low` and `high` מתחילהם עם ביטים משותפים, נרצה למצוא מספר קצר ביותר בין `low` ל `high`.
  - `high = 0.0111100000101010111010` במקרה זה נדרש לפחות שלושה ביטים לאחר הביטים הכהולים, תוך שמירה על האינטראול הנוכחי וללא גלישה לאינטראול הבא. (כך צמצמנו את חלק מהביטים השחורים).

\* נקח את ה range האחרון,  $- < 3.67 \cdot 10^{-5} - 0.00003667$ . למעשה יש קשר בין הטווח האחרון, לבין גודל המספר אותו נדרש להعبر למספר העונח.

נחשב את כמות האינפורמציה:  $[3.67 \cdot 10^{-5} \log_2(15)]$  (ערך עליון) –-> החסם התיכון לקומות הביטים אותן נדרש להعبر.

איך נודיע למספר העונח מתי הוא צריך להפסיק? לשון 2 דרכים לפתרון הבעיה:

  - נאמר למפענה כמה תווין יש בהודעה המקורית. יהיה לנו `prelude` המתאר את המספר התווך בהודעה המקורית.
  - נוסיףתו דמה לא'ב (\$) , התו זהה יופיע רק בסוף הקובץ. וההסתברות שלו תהיה מאוד נמוכה (1 חלקי אורך ההודעה) וברגע שהמפענה יגע לתו זהה הוא ידע שהוא סימן.

- $E(M) = 0.572$
- $D(M) \in CCA BD$

המפענה מקבל מספר בין 0 ל 1. בגרסה הסטטיטית גם המפענה ידוע את ההסתברויות. ובנוסף הוא ידוע את סדר הא"ב באינטראול.

ההודעה שלנו  $M = 0.572$  מפערת. נשאל איפה ( $M$ ) נופל?

## סיכום קורס דחיסת נתונים א'

<https://github.com/shaynaor>

נכתב על ידי: שי נאור

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

הוא נמצא בין 0.4 ל 0.8 ככלומר התו C.  
 נתען את ה range להיות בין 0.4 ל 0.8.  
 $(A = 1/5, B = 1/5, C = 2/5, D = 1/4)$   
 נחלק כל פעם את האינטראול לפי אותו יחס  
 קיבלנו ב prelude שאורך המילה הוא 5 ולכן נבצע 5 איטרציות.

אלגוריתם:

```

encoded ← Get (encoded number)
do{
    Find symbol whose range contains encoded
    Output the symbol
    range ← high(symbol) - low(symbol)
    encoded ← (encoded - low(symbol))/range
}until (EOF)
    
```

E = .109375	Symbol	Low	High
between 0.0, 0.25; output 'B'	B	0	0.25
E = (.109375 - 0.0) / 0.25 = .4375	I	0.25	0.50
.4375 between 0.25, 0.5; output 'I'			
E = (.4375 - 0.25) / 0.25 = 0.75	L	0.50	1.00
0.75 between 0.5, 1.0; output 'L'			
E = (0.75 - 0.5) / 0.25 = 0.5			
0.5 between 0.5, 1.0; output 'L'			
E = (0.5 - 0.5) / 0.25 = 0.0 → STOP			

### Compute size of interval

האינטראול האחרון:  $[r, l + r]$ , כלומר:

גודל האינטראול האחרון ( $r$ ) של ההודעה ( $r$ ) :

$$r = \prod_{i=1}^k p(m_i)$$

נראה לנו הטענה: כביכול נוציא נזק מפער. בפער:  $\sum_{j=1}^n \omega_j = 1$

$$P_i = \frac{\omega_i}{\sum_{j=1}^n \omega_j} = \frac{\omega_i}{1} \quad (M = m_1 m_2 \dots m_k)$$

הטענה מוגדרת:

$$-\log\left(\prod_{i=1}^k P(m_i)\right) = -\sum_{i=1}^k \log_2 P(m_i) =$$

$$= -\sum_{j=1}^n \omega_j \log_2 \omega_j = -K \sum_{j=1}^n \frac{\omega_j}{K} \cdot \log_2 \omega_j = K \underbrace{\left(-\sum_{j=1}^n \omega_j \log_2 \omega_j\right)}_{\text{הטענה}}$$

הטענה מוגדרת:

$$-\log\left(\prod_{i=1}^k P(m_i)\right) = H$$

הטענה מוגדרת:

למעשה הוכחנו שהקוד האריתמטי הוא הקוד הכיiesel שאפשר למצאו.

### Adaptive arithmetic coding

אלגוריתם אדפטיבי – אלגוריתם שעובר פעם אחת בלבד על הקובץ ובמהלך המעבר על הקובץ הוא לומד את הסתברויות וטור כדי המעביר הוא דוחס את הקובץ. כל פעם שנראהתו חדש נעדכן את השכיחות שלו כמו בהאפן דינמי. האינטראול בין 0 ל 1, בנקודה להתחלה בכל פעם לפי הסתברויות ההסתברויות שהתחלו איתם, הסתברויות התעדכנו והאינטראול יחולק באופן שונה לפי הסתברויות החדשנות.

נסמן ב  $(X)$  את מספר המופעים של התו  $X$  עד נקודה זו. נניח בהתחלה שכל התווים מופיעים בהסתברות שווה.

## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

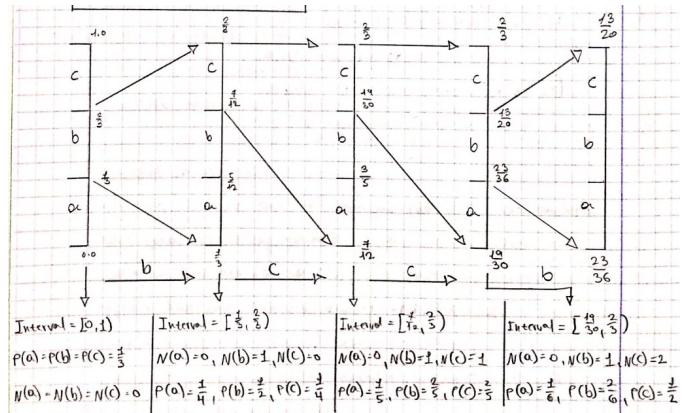
מבוסס על הרצאות והמצגות של פרופ' דנה שפירא, מחברת של שובל

- Alphabet {a,b,c}
- String bccb
- $N(x)$  - number of occurrences of symbol  $x$  in the already received substring

$$p(a) = \frac{N(a)+1}{N(a)+N(b)+N(c)+3}$$

$$p(b) = \frac{N(b)+1}{N(a)+N(b)+N(c)+3}$$

$$p(c) = \frac{N(c)+1}{N(a)+N(b)+N(c)+3}$$



### חרוגנות Drawbacks

-דיק: הדיק של המחשב ב point floationg point הוא מוגבל. אם מעביר קובץ בגודל 125kb –> נקבל מספר בין 0 ל 1 עם בערך מיליון סיביות של דיק. כדי ליצור קובץ דחוס בגודל 125, נדרש מחשב שיכול לחשב מיליון סיביות דיק.

-בעיה נוספת היא שלא ניתן לאפשר stream. המקודד מחשב עד האינטראול האחרון, ואז מעביר למפענה ורק לאחר שהמפענה מסים נוכל לעבוד.

-הקוד האריתמטי איטי יותר מהאפמן הרגיל (הأدפטיבי מהיר מהאפמן).

-access-random: גישה רנדומלית לאיזה מקום בקובץ הדחוס. בקוד אריתמטי לעומת האפמן אין שום משמעותות להגעה לאמצע הקובץ, יש לנו מספר עשרוני אחר שציריך להיות מפוענה מההתחלת ועד לסוף.

### Huffman vs Aritmetic

	English	Gadsby	German	Finnish	French
English	0.6	1.4	-2.1	-5.1	0.4
Gadsby	-2.4	1.0	-2.9	-0.8	-3.3
German	-1.8	-3.6	0.9	-5.2	-0.1
Finnish	2.4	1.2	2.7	0.8	2.9
French	0.8	2.2	-4.6	-11.3	0.9

בtabla נסתכל על האלכסון –> דוחסם אנגלית באמצעות הסטטיסטייה של אנגלית ומשווים את אחוז הדחיסה של האפמן והאריתמטי ומסתכלים על ההפרש ביניהם. מקבלתי שהאריתמטי יותר טוב מהאפמן בלבד היוטר אחד אחד. הצביע הכתוב –> הביצועים של האפמן יותר טובים.

### הרצאה 9 – Arithmetic coding

הקוד האריתמטי יותר טוב מהאפמן מכיוון שהוא מתקרובים יותר לאנתרופיה. בשיעור הקודם דיברנו על החסרונות של הקוד האריתמטי עם קבועים גדולים. נוצרך יכולת דיק גבוהה במספרים ולכן נרצה למצוא דרך להתמודד עם זה.

שתי דרכים לפתור הבעיה:

1. נעשה שימוש של הקוד האריתמי באמצעות מספרים שלמים.
2. נראה איך ניתן להשתמש בקוד בינהו.

## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

נרצה להגיע לדיווק אינסופי כאשר משתמש במספר קבוע של ספרות שנכונות ל'integer', ונעשה להם shift בכל פעם שהMSB يتלכדו. הפלט מייצג לנו את ההודעה שקודדנו עד כה.

Underflow

זהו מקרה שבו האינטראול שלנו קטן מדי, הוא so และ Hugh מתלכדים עם אותו מספר ואז יכולת הדיק שלנו לא טובת. נצטרך להרחיב את האינטראול, בהמשל נרחיב על נושא זה.

### Implementation

האינטראול הרוחני הוא מ 0 ל 1 (לא כולל 1). הייצוג הוא בסיס עשרוני. בזיכרון שבעצם עבורי

היצוג של 1.0 יש אינסוף 9-יות.

למשל אם יש לנו 5 ספרות דיווק:

לעומת, אם יש לנו 5 ספרות דיווק :

low = 0.00000  
high = 99999 (99...)  
range = high - low = 100,000 (99...)  
low = 0 + 0.2 \* 100,000 = 20,000  
high = 0 + 0.2 \* 100,000 = 39,000  
low = 0 + 0.2 \* 100,000 = 20,000  
high = 0 + 0.2 \* 100,000 = 39,000  
low = 20000 (00...)  
high = 29999 (99...)  
low = 20000 (00...)

המקרה הבא :

high = 999999 (99...)  
low = 00000  
low = 0 + 0.2 \* 100,000 = 20,000  
high = 0 + 0.2 \* 100,000 = 39,000  
low = 20000 (00...)  
high = 60000 + 0.8 \* 20,000 = 60,000  
low = 60,000 + 0.6 \* 20,000 = 72,000

### underflow problem

יכול להיווצר מצב שהצטמצמנו לאינטראול שהוא פחות מיכולת הדיק שלנו. נניח שבאייזהו שלב בטליר לקחנותו שההסתברות שלו היא כל כך נמוכה כך שהגענו ל

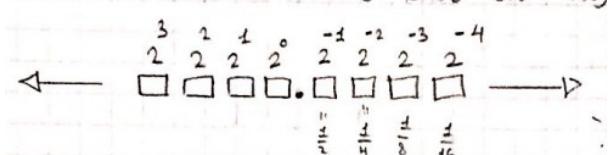
high = 40004 low = 39995

הה MSB לא מתלכד, והוא רואים שאם נמשיך בתטליר הזה למעשה נקבל אינטראול מאוד מצומצם, וכל מספר מתוך האינטראול ממופה לאותו חלק. נצטרך יכולת דיק גובה יותר. כאשר המצביע הנ"ל מתרחב, משתמש ב counter שיספור לנו כמה פעמים הגיעו למצביע הנ"ל. כאשר הגיעו למצביע של התלכדות: אם הגיעו למצביע שני מטלכנים עם 3, נקח את 3 ולאחריו 9 כערך ה counter. אם אנחנו מטלכנים עם 4, נקח את 4 ולאחריו 0 כערך ה counter.

### Binary fractional representation

ברגע שיש התלכדות -> נוציא לפולט ייצוג בינארי.

איך מיצגים בינארי מספר שהוא שבר?



## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

נקח את המספר  $x$  שלנו, ונשווה אותו כל פעם מול החזקות השליליות, אם  $x$  גדול או שווה במסמן 1. אחרת במסמן 0. וכך נדע איך לציג בבינהiri את המספר.

לעומת	
$x = \frac{5}{7}$	
$\frac{5}{7} > \frac{1}{2}$ (0.1) $\rightarrow \frac{5}{7} - \frac{1}{2} = \frac{3}{14}$	
$\frac{3}{14} < \frac{1}{2}$ (0.01)	
$\frac{3}{14} > \frac{1}{8}$ (0.101) $\rightarrow \frac{3}{14} - \frac{1}{8} = \frac{5}{56}$	
$\frac{5}{56} > \frac{1}{16}$ (0.1011) $\rightarrow \frac{5}{56} - \frac{1}{16} = \frac{1}{56}$	<span style="font-size: small;">⊗ ניכר מהלך סדרה של מינוסים ומשלים</span> <span style="font-size: small;">כמפורט בסעיפים הנוחים מטה</span> <span style="font-size: small;">ולא כפלה של מינוסים ומשלים</span>

דרך נוספת:

- Binary representation of  $x \in [0,1] x = .b_1 b_2 \dots$  where  $b_i \in \{0,1\}$
- ```

L ← 0; H ← 1; i ← 1
while x > L
    if x < (L+H)/2
        bi ← 0
        H ← (L+H)/2
    else if x ≥ (L+H)/2
        bi ← 1
        L ← (L+H)/2;
    i ← i + 1
end{while}
bj ← 0 for all j ≥ i

```

| לעומת  |   | L              | H              | $\frac{L+H}{2}$ |
|--------|---|----------------|----------------|-----------------|
| 0.1    | 1 | $\frac{1}{2}$  | $\frac{1}{2}$  | $\frac{1}{2}$   |
| 0.01   | 0 | $\frac{1}{4}$  | $\frac{1}{4}$  | $\frac{1}{4}$   |
| 0.101  | 1 | $\frac{1}{8}$  | $\frac{1}{8}$  | $\frac{1}{8}$   |
| 0.1011 | 1 | $\frac{1}{16}$ | $\frac{1}{16}$ | $\frac{1}{16}$  |
| לעומת  |   | $\frac{1}{2}$  | $\frac{1}{4}$  | $\frac{1}{8}$   |
| לעומת  |   | $\frac{1}{16}$ | $\frac{1}{32}$ | $\frac{1}{64}$  |

| לעומת  |   | L              | H              | $\frac{L+H}{2}$ |
|--------|---|----------------|----------------|-----------------|
| 0.0    | 0 | $\frac{1}{2}$  | $\frac{1}{2}$  | $\frac{1}{2}$   |
| 0.01   | 1 | $\frac{1}{4}$  | $\frac{1}{4}$  | $\frac{1}{4}$   |
| 0.010  | 0 | $\frac{1}{8}$  | $\frac{1}{8}$  | $\frac{1}{8}$   |
| 0.0101 | 1 | $\frac{1}{16}$ | $\frac{1}{16}$ | $\frac{1}{16}$  |
| לעומת  |   | $\frac{1}{2}$  | $\frac{1}{4}$  | $\frac{1}{8}$   |

scaling

דרך יותר קלה, מזכירה קצת את דרך הטיפול ב integer, נרצה להיכנס לאינטראול יותר עדין עם חזקות של 2.

אם  $x/2$  → נוציא לפט 1, ו"נבטל את השפעתו" על ידי הכפלת ב 2 וחיסור 1.  
אחרת → נוציא לפט 0, וככפוף ב 2.

אלגוריתם:

```

y ← x; i ← 0
while y > 0
    i ← i + 1;
    if y < 1/2
        bi ← 0
        y ← 2y
    else if y ≥ 1/2
        bi ← 1
        y ← 2y - 1;
bj ← 0 for all j ≥ i+1

```

- Invariant:  $x = .b_1 b_2 \dots b_i + y/2^i$

## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

incremental coding idea

נרצה לחת את הקוד האריתמטי ולבטא אותו בbinary.

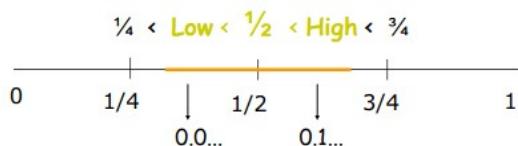
למשה נבצע  $in\text{-}zoom$  ונכפיל את ה range שלנו.

נבדוק האם אנחנו מעל החצי או מתחתיו.

אם מתחת –  $\rightarrow$  נכפיל ב 2 ונוציא 0.

אחרת  $\rightarrow$  נכפיל ב 2 ונוציא 1.

גם כאן יש לנו מצב של underflow. אם אנחנו נמצאים בין 0 ל 1. אם אנחנו נמצאים בחצי התיכון, אז נדע שהטיבית שנוציה לפולט תരהה 0. אם אנחנו בחצי העליון נוציא 1. המצב שעלה לחובביל אותנו ל underflow הוא שאמ נהיה בחצי האמצעי.



אם נהיה ברבע ה

- low

 של החצי האמצעי, נוציא 1

ולאחר מכן בoodoot 0. אם נהיה ברבע התיכון של

החצי האמצעי נוציא 0 ולאחר מכן בoodoot נוציא 1.

אנו חוששים שה low ו-high יהיו קרובים מדי אחד

לשני. המקודד חייב להבטיח שהאנטרואל בין ה  $low$  ל  $high$  יהיה מספיק גדול כדי למנוע זאת. (שניהם שונים לא ימכו באותו מקום). אם אנחנו נמצאים בחצי האמצעי, נעשה  $in\text{-}zoom$  אם קטן מ  $2/1$  נכפיל ב 2. אם גדול מ  $2/1$  נכפיל ב 2 ונוריד 1.

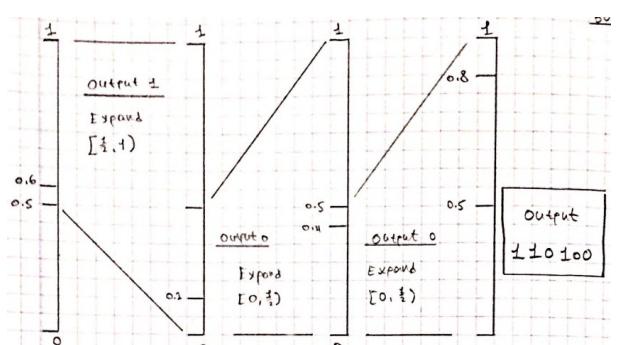
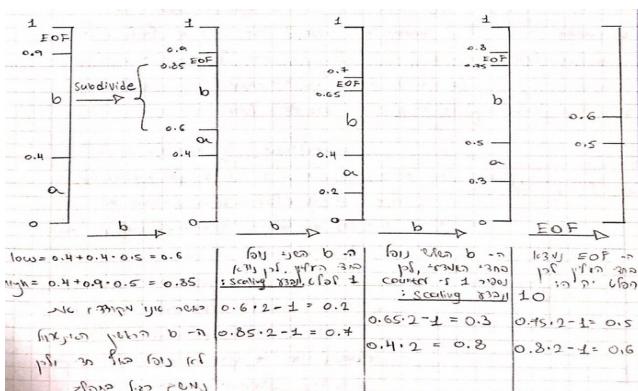
כאשר מקרה זה חוזר על עצמו נספר את האמצעות counter.

אם בסוף נגיע לחצי התיכון –  $\rightarrow$  הפלט יהיה 0 ואחריו 1 כמספר ה counter.

אם בסוף נגיע לחצי העליון –  $\rightarrow$  הפלט יהיה 1 ואחריו 0 כמספר ה counter.

ה counter נוחז רק במקורה של underflow, שקרה רק כאשר אני נמצאים בחצי האמצעי.

דוגמה: נקודד  $bbb$  ו EOF.



### יתרונות של הקוד האריתמטי:

- מגע לאנטרופיה.

- אפשר להפוך אותו לאדפטיבי בצורה מאוד פשוטה.

### חסרונות של הקוד האריתמטי:

- לוקח הרבה זמן לדחוס ולפענה.

- ביחס להאפקם היותר שלו בדחסה לא צזה גדול.

|            | Huffman | arithmetic | cost |
|------------|---------|------------|------|
| English    | 4.185   | 4.160      | 0.6% |
| Finnish    | 4.045   | 4.013      | 0.8% |
| French     | 4.000   | 4.037      | 0.9% |
| German     | 4.148   | 4.113      | 0.8% |
| Hebrew     | 4.285   | 4.249      | 0.8% |
| Italian    | 4.000   | 3.973      | 0.7% |
| Portuguese | 4.010   | 3.986      | 0.6% |
| Spanish    | 4.047   | 4.023      | 0.6% |
| Russian    | 4.470   | 4.443      | 0.6% |
| English-2  | 7.445   | 7.416      | 0.4% |
| Hebrew-2   | 8.037   | 8.009      | 0.4% |

## סיכום קורס דחיסת נתונים א'

כתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא, מחברת של שובל

### קוד שלא ניתן לסטנכרן

אם ב fixed length code איבדנו סיבית אחת או כל מספר סיביות שקטן מה fixed length code ב ASCII ו איבדנו מספר סיביות שהוא לא מכפלה של 8, לעומת זאת לא נסתנכרן עם הקוד הנוכחי. גם אם הקוד הנכון. לא ניתן לסטנכרן.

### הרצאה 10 LZSS, LZ77

static vs. Adaptive dictionary

דחיסה סטטיטית:  
המילון ידוע מראש לפני הקידוד. נדרש ידע קוד ס על הטקסט שנרצה לדחוס, כדי שהדחיסה תהיה טובה. המילון צריך להתחאים לקובץ שאנו מקודדים.

דחיסה אדפטטיבית:  
LZ77 – מעבור על הטקסט, ובמהלך המעבר על הטקסט גבינה את המילון ונעדכן אותו. גישה סטטיטית: אנו מנסים לחזור את התו הבא בתור באמצעות סטטיסטיות שפתחנו עד המיקום הנוכחי.

גישה ההחלפה: ק"מ ב LZ77, נמיר תתי מחרוזות שאנו עומדים לקובד באינדקם למילון, בהנחה שהאינדקם יתפוס פחות מקום מהמחרוזות. במבט הכלל נרצה להציגות היו "זולות" יותר מהמחרוזות.

| Dictionary     |        |
|----------------|--------|
| c <sub>i</sub> | symbol |
| 10             | a      |
| 1111110        | b      |
| 111110         | c      |
| 0              | aa     |
| 1111110        | aaaa   |
| 1111111        | ab     |
| 110            | baa    |
| 1110           | bccb   |
| 1110           | bccba  |

יש לנו מילון עם מילות קוד משתנות עבור כל אחת מן המחרוזות. למשל עבור "baa" הקוד 110. בכל פעם שנתקל ב "baa" נוכל להמיר אותה למילת הקוד שלו. נתונה לנו המחרוזת הבאה: "aaabccbaaaa" השאלה היא איך לבצע חלוקה של המחרוזת לפי הכנסיות שיש לנו במילון.

### שנון כמה שיטות:

1. מחמדני: מכל מקום במחזרות נkeh את המחרוזת הארוכה יbijoter במילון. כל פעם נוסיף לו ונבדוק האם ק"מ במילון. ברגע שנגיעה למילה שלא קיימת במילון נעצור.

לדוגמא: תחילת נבדוק את ה a הראשונה. היא נמצאת במילון, נבדוק עכשו אם aa במילון. מכיוון שהוא בא במילון נשיל לו הבא aa מכיוון שלא נמצא במילון. החלוקה המתבקשת בחמדני:

aa — ab — c — b — aa

מספר הביטים הדרושים לייצוג בשיטה הגרידית:  $2 + 6 + 6 + 7 + 1 = 30$  ביטים.

2. longest fragment. בשיטה זו נצטרך ללקת באופן רקורסיבי. נkeh את המחרוזת הארוכה ביותר שנמצאת במחזרות הנתונה וגם במילון, במקרה שלנו היא: "bccba" ונפריד את הקצוות שלה ונקבל aaa — bccba — aaa

נחזיר על התהילה עבור הקצוות שהפרידנו. נקבל בסוף:  
aa-a-bccba-aa-a

מספר הביטים הדרושים לקידוד: 11 ביטים.

## **סיכום קורס דחיסת נתונים א'**

<https://github.com/shaynaor> נכתב על ידי: שי נאור

mboss על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

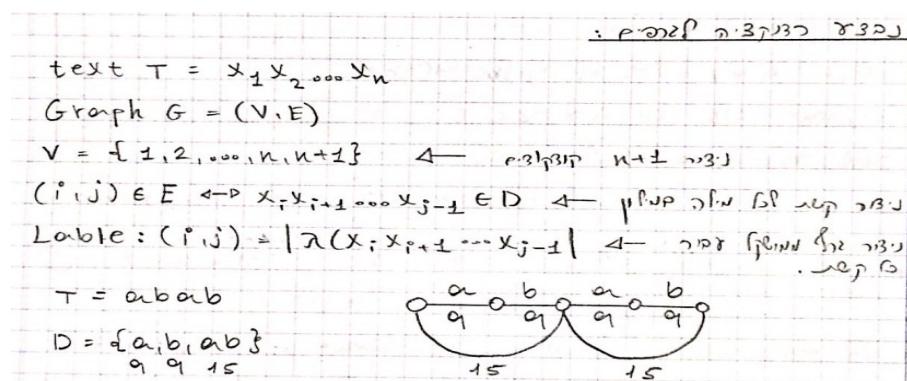
3. words min: נרצה למצוא את החלוקה המינימאלית לבלוקים או מספר כנישות למילון. במצב בו אוורך מיליות הקוד הוא קבוע חלוקה כזו תהייה טובה, אך במקרה שלנו זה לא נכון. מספר הביטים החדשניים לקידוד: 15 ביטים.

:optimal .4

aa-a-bccb-aa-aa

**מספר הביטים הדרושים לקידוד: 9 ביטים.**

בහינתן מילון סטטי שקבוע מראש, יש פונקציה למדוד שממפה את המחרוזות למילת קוד. מטרתנו למצוא חלוקה של  $\Sigma$  (הטקסט) למילים  $w_1w_2w_3\dots = T$  שכל מילה  $i$  נמצאת במילון  $S$ , כך שסכום מספר הביטים הוא מינימאלי.



ונכל להפעיל `dijkstra()` ולמצוא את המסלול המינימאלי בין קודקוד התחלה לקודקוד האחרון. וכן למצוא את החלוקה הכי טובה, שניתן ל`יצא` את הטקסט בכמה שפחות סיביות.

1777

## sliding window compression

הדחיסה אדפטיבית, אנו דוחשים את הטקסט ועוברים ערי' כרגע משמאל לימין ובנקודה מסוימת יש לנו את הטקסט שדחסנו שחלק ממנו נקרא לו window (לא נצטרך לחזור עד התחלה אלא להשתמש בwindow שהגדכנו). כאשר אנחנו נמצאים בנקודה מסוימת (אחרי מה שדחסנו) יהיה לנו את הטקסט שאנו עומדים לדוחס lookahead, נדוחס אותו בעזרת מה שידוע שנמצא במלון (window).

## Window | Look Ahead Buffer

- text window : recently encoded text.
  - look ahead buffer : those to be encoded.

## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

### הקובץ הדחוס מרכיב שלישי

- offset הסט: כמה תווים לאחר ציריך ללקת כדי לקרוא את המחרוזת הנוכחית משם. (במקרה של שיליה חוזרת על עצמה).
- Length אורך: אורך המחרוזת המועתקת.
- Symbol: התו הבא בתור אחרי המחרוזת החזרת.

כל שלשה כזו מייצגת תת מחרוזת באורך משתנה, שרצים לדחוס.

דוגמת הריצה LZ77:

- String:

A\_walrus\_in\_Spain\_is\_a\_walrus\_in\_vain.

- Encoded String

(0,0,'A')(0,0,'\_')(0,0,'w')(0,0,'a')(0,0,'l')(0,0,'s')(7,1,'i')(0,0,'n')(3,1,'S')(0,0,'p')(11,1,'i')(6,2,'i')(12,2,'a')(21,11,'v')(20,3,'.)

תחילה הדחיסה לא טובה אבל אחר כך זה הולך ומשתפר.

### LZ77 encoding

```
1. p←1 // The next character to be coded
2. while there is text remaining to be
   coded{
  1. search for the longest match for S[p...]
     in S[p-W..p-1] suppose that the match occurs
     at position m with length l
  2. Output the triple (p-m, l, S[p+l])
  3. p←p+l+1
}
```

כל עוד יש מה לדחוס, חפש את המחרוזת הארוכה המתאימה ביותר בראיראט בwindow, אם קיימת, תמציא שלשה מתאימה ווציאה את התו הבא בתור, ואז תתקדם לתו הבא בתור.

```
1. p←1 // The next character to be decoded
2. For each triple (f, l, c) in the input{
  1. S[p..p+l-1]←S[p-f..p-f+l-1]
  2. S[p+l]←c
  3. p←p+l+1
}
```

### LZ77 decoding

| input     | 12 11 10 01 8 + 6 5 4 3 2 1 |
|-----------|-----------------------------|
| (0,0,'a') | a                           |
| (0,0,'b') | a b                         |
| (0,0,'r') | a b r                       |
| (3,1,'c') | a b r a c                   |
| (2,1,'a') | a b r a c a d               |
| (7,4,'d') | a b r a c a d a b r a d     |

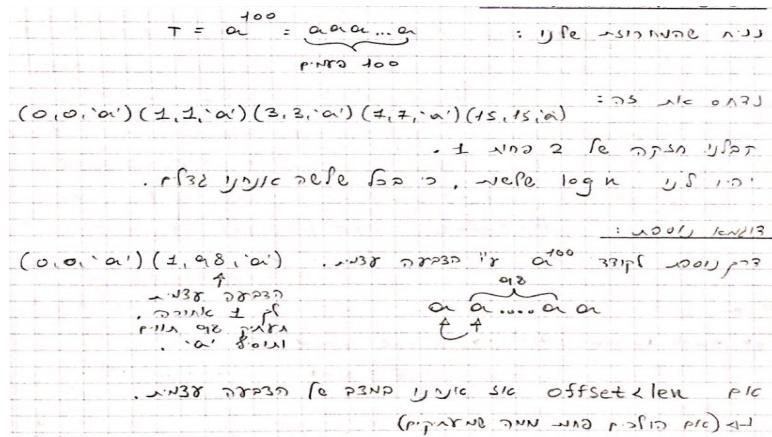
### דוגמה לפענוח

## **סיכום קורס דחיסת נתונים א'**

<https://github.com/shaynaor> נכתב על ידי שי נאור

mbossum על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

## Self reference



## problem with LZ77

איך מוצאים את המחרוזת הנוכחית הארוכה ביותר בעיתיות במבנה של השלשה – כל פעם שאין לנו מחרוזת שאנו יכולים להעתיק, אנחנו מבזבזים שלשה.

נניח שה `window` שלנו הוא בגודל 4096 תווים.

וה lookahead buffer גודלט 16 תווים.

השאלה היא כמה תווים אנו צריכים על מנת לקודד את  $(0,0,"c")$  ?

$$2^{12} = 4096$$

LAB 16 =  $2^4$  גודל ה

נרצה לחשב מה הגודל של השלשה:

offset = 12 bit

length = 4 bit

symbol = 8 bit (ASCII)

## עלות יציג השלישי 24 ביטים.

בעם קיימן "זבוז" ביטים – שליטה של שלשה עברותו חדש. ככל שנרצה חלון גדול יותר, נשלם מחיר גדול יותר עבור כלתו.

LZSS

LZSS משפר את LZ77 המקורי.

ב-SSZ מוצא מחרוזת ארוכה ביותר שהופיע קודם לכן ידי שימוש בעץ חיפוש בinner' מאוזן. SSZ לא משתמש בשלשות. או שיש לותו בודד או שיש לו מחרוזת חוזרת. אם המחרוזת חוזרת נשתמש בזור סדור.

האוסף של SZSS בנוי מ 2 סוגים העתקות

## 1. בודדים : תווים בודדים

.offset | length : מצביעים על pointers (offset, length) .2

איך נבדיל בינהם?

## סיכום קורס דחיסת נתונים א'

<https://github.com/shaynaor>

נכתב על ידי: שי נאור

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

נקח בית אחד נוסף שיצין האם הביטים שאחורי הם تو בודד או מחרוזות מועתקת. את כל התווים הבודדים נרשום בצורה מפורשת, ואת השאר נרשום כזוגות סדורים (כਮון ללא פסיק וסוגרים).  
כעת,תו בודד יעלה לנו 9 סיביות – 8 עבור יצוג ASCII + בית בודד.

מהו גודלו של המצביע לפי שיטת LZSS ?

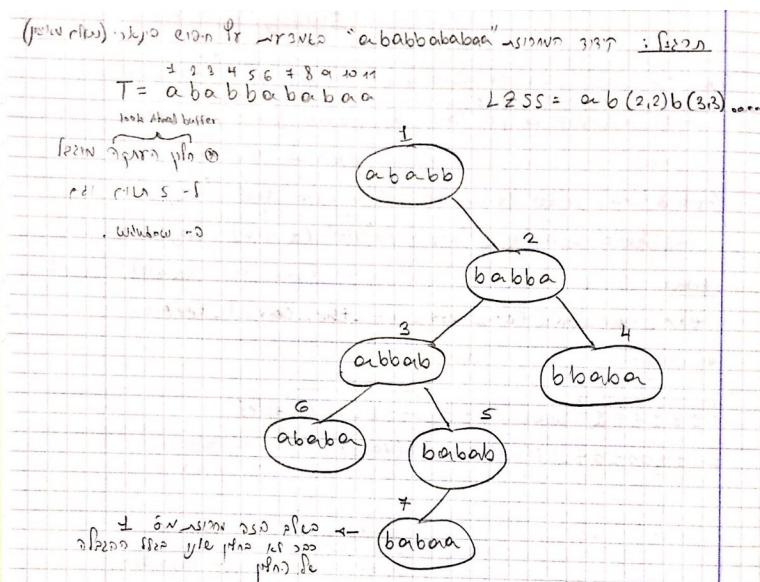
מקודם יעלה לנו 24 ביטים, וכעת יעלה לנו  $1 + 4 + 12 = 17$  ביטים.

offset:12

length:4

1: בית לסימן (תו או מצביע).

לקודד 2 תווים בודדים יעלה לנו  $9 + 9 = 18$  סיביות, ולכן עבור מחרוזות באורך 2 ומעליה נעדיף להשתמש בהצבעה במידת האפשר.



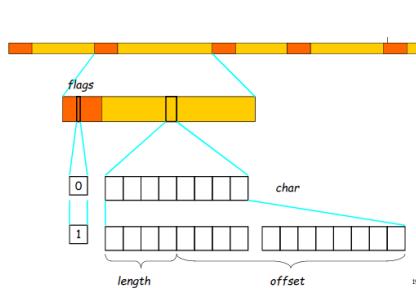
מסודר בסדר לקסיקוגרפי, הגدول נכנס למינה.

-בהתחלת החלון ריק, נקודדתו ראשון ונוסיף לעצם המחרוזת ababb –  
חמשת התווים הראשונים החל מהמיקום הראשון.

-נעבור לאות הבא b. חמשת התווים הבאים הם babba, אף שתת מחרוזת לא מוכלת במחוזת שבשורש (1) ולכן נקודד את b ונוסיף את המחרוזת לעצם.

-נעבור לאות הבא a, חמשת התווים הבאים הם abbab, נשים לב ש a מוכל במחוזת שבשורש. וכך כתוב (2,2) ונוסיף את המחרוזת.

הערה: בדוגמה העץ לא מאוזן, במימוש למציאות משתמש בעץ מאוזן.



Encoding the element

מחשב מתמודד טוב יותר עם בתים מאשר עם ביטים. הביט המוביל שאנו מוסיפים כדי להבדיל ביןתו בודד לבין סדר סדור יכול מאד לסרבל את העניין זהה. לכן נעדיף לאחד ביטים מובילים יחידות של בתים.

## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא , מחברת של שובל

Locating previous occurrences

עץ חיפוש בינארי מאוזן הוא מבנה נתונים קבוע ומסורבל. נוכל להשתמש ב-table hash. נפעיל פונקציית hasing על התווים שנרצה לקודד. הפונקציה תמפה את המילה למקום בטבלה שם יהיה החישוב של מספר הצעדים שעליינו ללקת אחרת. במצב של התנגשות נוכל לפטור זאת באמצעות רשימה מקוורת.

### LZS

#### LZS stacker

|          |    |        |                |    |
|----------|----|--------|----------------|----|
| • Char   | 0  | 8 bit  | 00             | 2  |
| • Offset | 11 | 7 bit  | 01             | 3  |
|          | 10 | 11 bit | 10             | 4  |
|          |    |        | 1100           | 5  |
| • Length |    |        | 1101           | 6  |
|          |    |        | 1110           | 7  |
|          |    |        | 1111 0000      | 8  |
|          |    |        | 1111 0001      | 9  |
|          |    |        | ...            |    |
|          |    |        | 1111 1110      | 22 |
|          |    |        | 1111 1111 0000 | 23 |
|          |    |        | 1111 1111 0001 | 24 |

פונקציית הלמדא: char -> סיבית מובילה 0 ועוד 8 סיביות (ASCII)  
 חלקות ה offset לשני סוגים:  
 offset קצר -> עד 128 תווים שלוקחים 7 סיביות (ביט מוביל 11).  
 offset ארוך -> עד 2048 תווים שלוקחים 11 סיביות (ביט מוביל 10).  
 length -> מוצג על ידי משחו שדומה ל C גמא.

### הרצאה 11 – LZ78 LZ77

דחיסה מילונית נוספת.

ב LZ77 המילון הוא בעצם הchlou שבו אנחנו בודקים האם מחרוזת הופיע קודם. ב LZ77 המילון הוא בלתי מוגבל (יכול לגודל כמה שנרצה) אבל נגביל אותו כדי שנתעסק במשהו מעשי.

בניית המילון על ידי כך שכל פעם נוסיף מילה (שכבר נמצאת במילון) + تو חדש. ב LZ77 השתמשנו ב offset שאמור לנו כמה צעדים ללקת אחרת והעתקנו מחרוזת באורך מסוים. ב LZ77 לא נעשה ככה. המפענה יבנה את אותו מילון שהמקודד בונה בזמן הדחיסה. זו תהיה דחיסה אדפטיבית כמו ב LZ77.

נתחיל עם מילון ריק ומטרתנו היא לנקח את הטקסט ולהחלק אותו לתחתי מחרוזות כך שכל תחת מחרוזת תופיע במילון. החלוקת של Z בהתאם למחרוזות שנמצאת במילון נקראת parsing . בניית את המילון בצורה אדפטיבית, נחלק את Z לתחתי מחרוזות כך שכל תחת מחרוזות היא אינדקס במילון. כל מילה חדשה נכנסת למקום הפניו הבא במילון, שמתחליל כמיילון ריק. שיש בתוכו את Z שמייצג את המילה הריקה. ומילון {....., Z<sub>0</sub>, Z<sub>1</sub>, ...} = D . נחפש את המילה הארכיה ביותר שיש לה את הפרספקט של ה milah ahead , אוסף כל התווים עד האורך של Z . כל מחרוזות שנוסיף למילון נקודד באמצעות אינדקס, והתו הנוסף שאוטנו הוספנו.

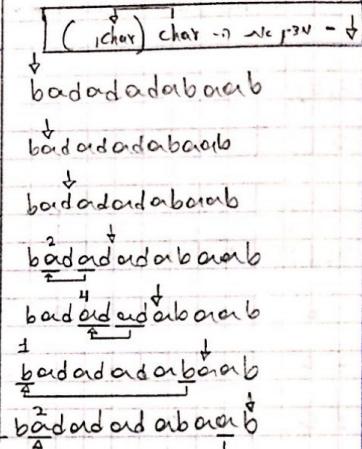
## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא, מחברת של שובל

$T = \text{badadadabaaab}$

| Index | Phrase     | Encoding | # of bits |
|-------|------------|----------|-----------|
| 0     | $\epsilon$ |          |           |
| 1     | b          | (0,b)    | 0+2       |
| 2     | a          | (0,a)    | 1+2       |
| 3     | d          | (0,d)    | 2+2       |
| 4     | ad         | (2,d)    | 2+2       |
| 5     | ada        | (4,a)    | 3+2       |
| 6     | ba         | (1,a)    | 3+2       |
| 7     | aab        | (2,b)    | 3+2       |



$T = (b)(a)(d)(ad)(ada)(ba)(aab)$

## 7.2.7 אלגוריתם

last matching index = 0

next available index = 1 (אינדקם הכניסה במילון)

כל עוד לא הגיעו לסופם הקלט (מרזות):

نחות אם קיימת תת מחרוזת בקלט, שנמצאת במילון החל מהמקום הנוכחי.

- אם לא קיימת צואת תת מחרוזות:

- ניצור כניסה חדשה במילון (last matching index, char) כאשר char זה התו עליו אנחנו עומדים כרגע.

- נAPS את next available index ונקדם את last matching index

- אם קיימת צואת תת מחרוזות:

- אז ניצור כניסה חדשה (last matching index, char) כאשר char זה התו הנוכחי שעליו את next available index של הכניסה המתאימה שמצאנו. char זה התו הנוכחי שעליו אנחנו עומדים.

$T = \text{badadadabaaab}$

Decoding

| Input | Decoding |
|-------|----------|
| (0,b) | 1 b      |
| (0,a) | 2 = a    |
| (0,d) | 3 = d    |
| (2,d) | 4 = ad   |
| (4,a) | 5 = ada  |
| (1,a) | 6 = ba   |
| (2,b) | 7 = aab  |

## סיכום קורס דחיסת נתונים א'

<https://github.com/shaynaor>

נכתב על ידי: שי נאור

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא , מחברת של שובל

87Z מעתה משמש בعزيز TRIE, נאותה את המילון כגודל הא"ב שלנו, אם אנחנו לא מכירים את הטקסט, נאותה את המילון ב 256 תוים ASCII נכניס כל phrase במילון ב TRIE ונטען בו: ברגע שהגענו לעלה סימן שהגענו למחרוזת היכי ארוכה הנוכחית אז נרchieb אותה.

LZW

### Encoding algorithm

```

1. Dictionary ← single Characters
2. w ← first char of input
3. repeat{
    1. k ← next char
    2. if(EOF)
        1. output code(w)
    3. else if (w + k) ∈ Dictionary
        1. w ← w + k
    4. else
        1. output code(w)
        2. Dictionary ← w + k
    3. w ← k
}

```

Data Compression Course - Penn Chandra

1. נאותה את המילון עם הא"ב שלנו. (ASCII אם הטקסט לא ידוע).

2. W מקבל את התו הראשון של הקלט.

3. כל עוד יש מה לדוחס (הטקסט לא נגמר):

3.1. K מקבל את התו הבא בתור.

3.2. אם הגיעו לסוף הקובץ -> נוציא את הקוד של W

3.3. אחרת נבדק האם K\*W בmilion -> K = W\*

3.4. אחרת:

נוציא את W

נרכיב את המילון עם K

W = K

### דוגמא

|    |          | Dictionary |        |
|----|----------|------------|--------|
|    |          | Code       | Phrase |
| 0  | " - "    |            | w      |
| 1  | " a "    |            | a      |
| 2  | " b "    |            | b      |
| 3  | " o "    |            | o      |
| 4  | " w "    |            | w      |
| 5  | " wo "   |            | wo     |
| 6  | " ab "   |            | ab     |
| 7  | " bb "   |            | bb     |
| 8  | " ba "   |            | ba     |
| 9  | " a - "  |            | a-     |
| 10 | " - w "  |            | -w     |
| 11 | " wab "  |            | wab    |
| 12 | " bba "  |            | bba    |
| 13 | " a-w "  |            | a-w    |
| 14 | " wabb " |            | wabb   |
| 15 | " ba - " |            | ba-    |
| 16 | " - wa " |            | -wa    |
| ⋮  | ⋮        |            | ⋮      |

|         | W | K     |
|---------|---|-------|
| " w "   | w | ☒ Dic |
| " o " △ | o | ☒ Dic |
| " b " △ | b | ☒ Dic |
| " b " △ | b | ☒ Dic |
| " a " △ | a | ☒ Dic |
| " - " △ | - | ☒ Dic |
| " w " △ | w | ☒ Dic |
| " w " △ | w | ☒ Dic |
| " wo "  | o | ☒ Dic |
| " wo "  | b | ☒ Dic |
| " b " △ | b | ☒ Dic |
| " bb "  | a | ☒ Dic |
| " ba "  | - | ☒ Dic |
| " a - " | w | ☒ Dic |
| " - w " | a | ☒ Dic |
| " wab " | w | ☒ Dic |
| " wab " | b | ☒ Dic |
| " b " △ | a | ☒ Dic |
| " ba "  | - | ☒ Dic |
| " - " △ | w | ☒ Dic |
| " - " △ | a | ☒ Dic |
| " a " △ | b | ☒ Dic |
| " a b " | b | ☒ Dic |
| ⋮       | ⋮ | ⋮     |

## סיכום קורס דחיסת נתונים א'

<https://github.com/shaynaor>

נכתב על ידי: שי נאור

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא, מחברת של שובל

## WZJ אלגוריתם הפענו

```
1. Initialize table with single character strings
2. OLD = first input code
3. output translation of OLD
4. while not end of input stream{
    1. NEW = next input code
    2. if NEW is not in the string table
        1. S = translation of OLD
        2. S = S . C
    3. else
        1. S = translation of NEW
    4. output S
    5. C = first character of S
    6. Translation(OLD) . C to the string table
    7. OLD = NEW
    .
```

1. נאתחל את המילון עם הא"ב שלו.
2. OLD מקבל את הקלט הראשון של הקובץ הדוחס.
3. נוציא את OLD לפלט.
4. כל עוד לא הגיעו לסוף הקובץ:
  - 4.1 NEW מקבל את הקלט הבא בתור.
  - 4.2 אם NEW לא נמצא במילון S 4.2.1 מקבל את התרגום של OLD  
 $S = S * C$  4.2.2
  - 4.3 אחרת:
    - 4.3.1 S מקבל את התרגום של NEW
    - 4.4 נוציא את S לפלט
    - 4.5 C מקבל את התו הראשון של S
    - 4.6 נעדכן את המילון עם התרגום של OLD משורשר עם C  
OLD = NEW 4.7

## דוגמה

| Dictionary |        |  |  |
|------------|--------|--|--|
| Code       | Symbol |  |  |
| 0          | a      |  |  |
| 1          | b      |  |  |
| 2          | c      |  |  |
| 3          | ab     |  |  |
| 4          | ba     |  |  |
| 5          | abc    |  |  |
| 6          | cb     |  |  |
| 7          | bab    |  |  |
| 8          | babk   |  |  |
| 9          | baba   |  |  |
| 10         | aa     |  |  |
| 11         | aaa    |  |  |
| 12         | aaaa   |  |  |
| 13         | aaaaa  |  |  |

OLD    NEW    C    S

0       1     b    b  
1       2     a    ab  
2       3     c    ba  
3       4     ab    bab  
4       5     ba    babk  
5       6     abc    baba  
6       7     cb    aaaa  
7       8     bab    aaaaa  
8       9     babk    aaaaaa  
9       10    a    a  
10      11    a    aa  
11      12    a    aaa  
12      13    a    aaaa  
13      -     -    -

## Applications

- אחד הראשונים להשתמש ב WZJ היה UNIX. הם התחילו עם מילון בגודל 512 כניסה, כאשר החצי הראשון שלו היה מלא בקוד ASCII (256 תווים) והחצי השני ריק. וכך כל כניסה במילון יוצגה על ידי 9 סיביות. כאשר המילון מתמלא, הם מגדילים אותו ל 1024 כניסה עם ייצוג של 10 סיביות. הם הגיעו עד לגודל של  $2^{10}$  ושם הם עצרו. המילון נהפר להיות סטטי, אבל הם בדקו כל הזמן את איות הדחיסה, קלומר את כמות הביטים שהם מצויים במילון, מול מספר התווים בהם קראו ונדרשו. אם כמות זו יורדת מתחת לערך סף כלשהו קבוע, הם איתחלו את המילון מחדש.
- GIF משמש לדחיסה של תמונות דיגיטליות שנוצרו על ידי המחשב.
- PNG החליף את GIF והוא חינמי מבוסס WZJ, גודל החלון A 32K והוא משתמש גם בהאפן כאחד מהמרכיבים.

## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא, מחברת של שובל

### LZ78/LZW vs. LZ77/LZSS

LZ77 דוחס בדרך כלל הרבה יותר טוב מ LZW/LZ8/LZSS

LZW יש לי כניסה במילון שלועלם לא משתמש בהן, ישן תמי מחוזות שלא נמצאות במילון.

LZ77 מכיל את המילון שהעבכנו אליו ובלז יש מילון מפורש.

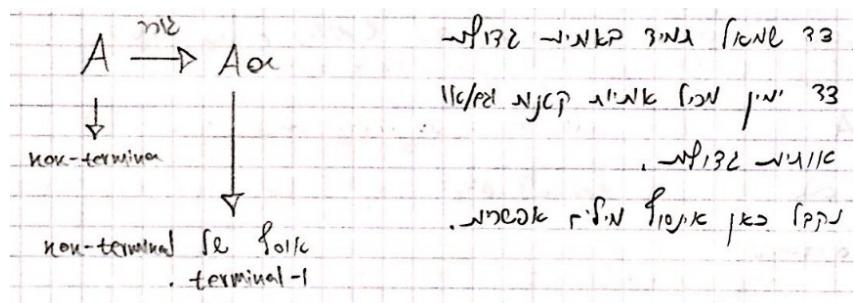
### Grammar Compression

כל שפה טبيعית מורכבת מתרוים וסימני פיטוק שהם יוצרים מילים ומהמילים יוצרים משפטים, וכך שמספרם הוא אינסופי. כדי להפוך את זה לשפה פורמלית, יש כמה כלליים שנרצה לאכוף, נקרא להם כללי דקדוק.

את הכללים נציג באופן הבא:

Terminal symbols: אבני היסוד של השפה, הא"ב (מסומן באותיות קטנות בדרך כלל)  
Production rules: הדרך לבנות מילה או מחוזות חדשה (מסומן באותיות גדולות בדרך כלל)

דוגמה:



נשתמש בהם דקדוק כדי לדוחס.

למד שני אלגוריתמים: Sequiter Repair I –Repair, שניהם דומים אחד לשני וمبוססים על כללי הדקדוק. נdag שהמפענה מקבל את אותם כללי הגזירה ויכול לבנות מהם מחוזות אחד כדי שהיא טע. בנויגוד לכך דקדוק כלליים ששם אפשר לקבל כמה מחוזות, כאן נאפשר רק אחת.

### Context-free grammar

דקדוק חסר הקשר

A → ab , B → Ac , C → BdA

דוגמה:

ונס הולכת מכאן הרין כז גזין נס האיזוטר חוגהן:

... abc, abcd, abc,

באמצעות כללי הדקדוק נוכל לדעת האם המילה היא חוקי או לא. מכיוון שככל הדקוק מורידים את היתרונות, מכיוון שככל הדקוק מורידים את היתרונות נוכל באמצעות מספר סופי של כלליים להגדיר אינסוף מחוזות. לצורך דקדוק שמתאים למחרוזות הנווכheit ונעביר אותו (דוחס) למפענה, ישנו כל

## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

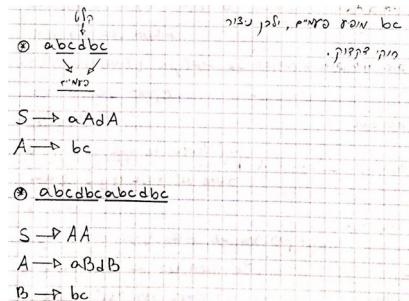
מini דרכם לדחוס את הכללים ולהעביר אותם למפענה. לא בהכרח נדרש להעביר את הדקדוק בצורתו המפורשת. נתמקד במקרה של יצירת הדקדוק ולא בדחסתו.

Straight line grammar  
זהו מקרה פרטי שבו נוצר מחרוזת אחת בלבד.

### Sequitur

משמעותה "it follows" בלטינית. הרעיון של האלגוריתם הוא לקרוא את התווים אחד אחד משמאלי לימין, אך דורש עיבוד נספף. יקח לנו (A) 0 ליצור את המחרוזות כולה.

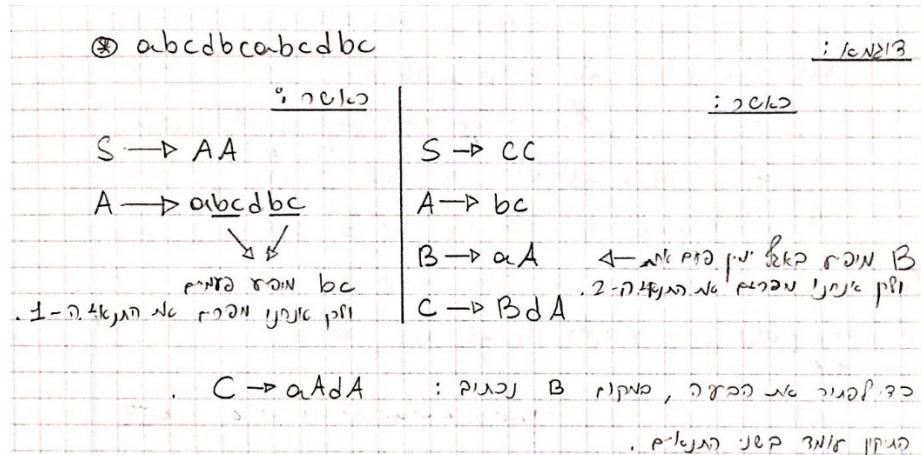
איך זה עובד: כל עוד הוא יצא שני צוים צמודים החוזרים על עצם, הוא ייצור חודד דקדוק חדש, ובסופו של דבר ניקח את כל חוקי הדקדוק ואוטם עברך ייחסה נוספת.



דוגמה:

נ לצורך לאכוף שני תנאים באלגוריתם sequiter:

- לא אפשר לשני תווים צמודים להופיע יותר מפעם אחת בדיאק.
- אם משתמשים בחוק פעמיים אחת בלבד, אז נקוץ אותו, נרצה להשתמש הכל חוק לפחות פעמיים כדי שהיה יעיל.



האלגוריתם אומר לנו איך לבנות את הדקדוקים שלב אחר שלב, כאשר אנו עומדים בשני החוקים.



## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא, מחברת של שובל

Re-pair

1. Find the most frequent pair  $ab$  in the text  $S$ .
  2. Create the rule  $A \rightarrow ab$ .
  3. Replace all occurrences of  $ab$  in  $S$  by  $A$ .
  4. Iterate until every pair appears only once
- $T = \text{singing\_do\_wah\_diddy\_diddy\_dum\_diddy\_do}$

|                    |                                                |
|--------------------|------------------------------------------------|
| $A \rightarrow _d$ | $\text{singingAo\_wahAidddyAidddyAumAidddyAo}$ |
| $B \rightarrow dd$ | $\text{singingAo\_wahAiByAiByAumAiByAo}$       |
| $C \rightarrow Ai$ | $\text{singingAo\_wahCByCByAumCByAo}$          |
| $D \rightarrow By$ | $\text{singingAo\_wahCDCDAumCDAo}$             |
| $E \rightarrow CD$ | $\text{singingAo\_wahEEAumEAo}$                |
| $F \rightarrow in$ | $\text{sFgFgAo\_wahEEAumEAo}$                  |
| $G \rightarrow Ao$ | $\text{sFgFgG\_wahEEAumEG}$                    |
| $H \rightarrow Fg$ | $\text{sHHG\_wahEEAumEG}$                      |

## Tunstall and Fibonacci codes

### Motivation

| Dictionary word | Codeword |
|-----------------|----------|
| aaa             | 0000     |
| aac             | 0001     |
| ab              | 0010     |
| ac              | 0011     |
| bca             | 0100     |
| bb              | 0101     |
| c               | 0110     |
| ccca            | 0111     |
| cccb            | 1000     |
| ccb             | 1001     |

עד עכישו בנוינו מילوت קוד באורךים משתנים, כתעת נרצה להתעסק עם מילות קוד באורך קבוע, אך התווים הם באורך משתנה. נניח שאנו רוצים ליצור מילות קוד באורך 4, מ-0000 עד 1111. והשאלה היא איזה מהירות כראוי לנו להכין לתוך המילון. לא משנה אילו מילים נכנים לתוך המילון, תמיד נשתמש במילות קוד באורך 4, בניגוד למה שראינו עד עכשוו, למשל, בהאפען היה לנו תווין בודדים, ורצינו לבנות מילות קוד באורך משתנה, כאן בעצם נרצה מילת קוד באורך קבוע, וקילט (מילים) באורך משתנה.

– בהאפען לקחנו תווים בודדים באורך 1 והפכנו אותם למילות קוד באורך משתנה.  
– יכולנו להשתמש בהאפען עבור מילים באורך משתנה ויצור עבורים מילות קוד באורך משתנה.  
– בהמשך נדבר על מהירות באורך משתנה שנוצר להם מילות קוד באורך קבוע.  
ונכל בעצם לומר שהבעיה שלנו כתעת היא הפוכה לבעיה שהייתה בהאפען, בהאפען נתנו לנו הא"ב ואנו רוצים למצוא את מילות הקוד באורך משתנה כך שסכום א' יהיה מינימאלי.

## **סיכום קורס דחיסת נתונים א'**

<https://github.com/shaynaor> נכתב על ידי: שי נאור

mbos על הרצאות והמצגות של פרופ' דינה שפירא, מחברת של שובל

בבעיה הנוכחית אנו יודעים מהי מילת הקוד ונצטרך לבחור מחורזות מתוך הא"ב לאוֹתָה מילת קוד. הדגש בהאפקט הוא על הקידוד, רצינו ליצור דחיסה כמה שיותר טובה. במקרה זהה הדגש הוא על הפענוח, אנו רוצים ליצור מילות קוד בעלות אורך קבוע על מנת שהפענוח יהיה כמה שיותר מהיר. בכוונה נבנה את האורך של מילות הקוד בכפולה של בתים כדי שהפענוח ישבור את הקידוד לפי מילים וכך הפענוח יהיה מהיר יותר.

```

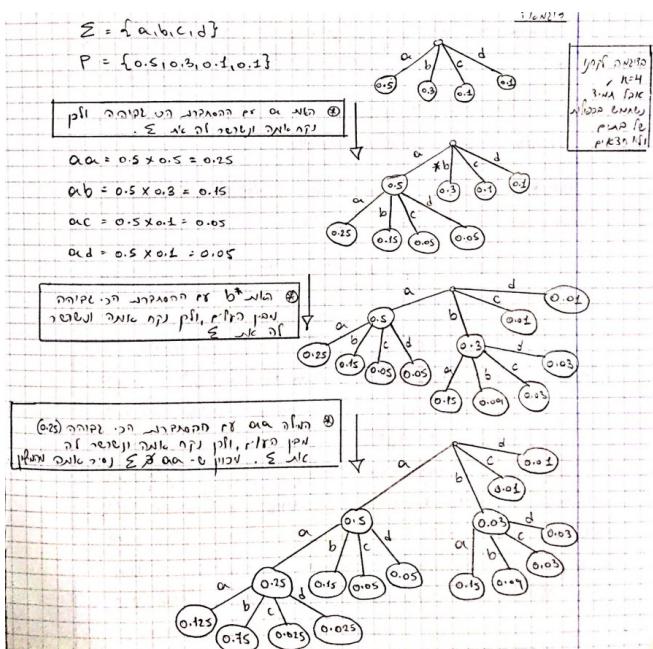
Tunstall(n) {
    D<-Σ
    while (|D|≤2n - |Σ|) {
        Let dεD be with highest probability
        D<-D ∪  $\left( \bigcup_{σ ∈ Σ} dσ \right)$ 
        if d∉Σ
            D<-D - {d}
    }
}

```

Tunstall Algorithm

יש לנו מילון שמאתחל עם הא"ב, כל עוד הגודל של המילון הוא קטן או שווה ל  $2^d$  פחות גודל הא"ב נקח מילה d (עליה עם ההסתברות הכי גבוהה, נשරר לה כל פעםתו אחר מהא"ב ונוסף למילון, אחרי שהוספנו את כל השירטורים אם המילה d לא הייתה לא"ב, נסיר אותה מmiloon).

## דוגמאות:



| Dictionary word | Codeword | Dictionary word | Codeword |
|-----------------|----------|-----------------|----------|
| <i>aaa</i>      | 0000     | <i>bb</i>       | 1000     |
| <i>aab</i>      | 0001     | <i>bc</i>       | 1001     |
| <i>aac</i>      | 0010     | <i>bd</i>       | 1010     |
| <i>aad</i>      | 0011     | <i>a</i>        | 1011     |
| <i>ab</i>       | 0100     | <i>b</i>        | 1100     |
| <i>ac</i>       | 0101     | <i>c</i>        | 1101     |
| <i>ad</i>       | 0110     | <i>d</i>        | 1110     |
| <i>ba</i>       | 0111     |                 |          |

נחשב את ההסתברויות בהנחה שהן בלתי תלויות. חישור – המחרוזות הן בלתי תלויות ולכן מילים עם הסתברות גובה לא בהכרח מופיעות הרבה.

## סיכום קורס דחיסת נתונים א'

<https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דנה שפירא , מחברת של שובל

$$F_0 = 0$$

$$F_1 = 1$$

$$F_i = F_{i-2} + F_{i-1}$$

0 1 1 2 3 5 8 13 21 34 55 89 144 ...

$$F_n = \frac{\phi^{n+1} - \bar{\phi}^{n+1}}{\sqrt{5}}$$

where  $\phi = \frac{1+\sqrt{5}}{2} = 1.618$  and  $\bar{\phi} = \frac{1-\sqrt{5}}{2} = -0.618$

## Fibonacci Numbers

נשתמש בסדרת פיבונacci' כבסיס למספר (בבנייה הקוד) עד עכשו ביצוג בינארי בסיס המספר היה חזקות של  $2^n$ .

- To encode an integer  $n$ :

- Find the largest Fibonacci number  $\leq n$
- Repeat recursively with  $n - F_k$  until  $n - F_k = 0$

- $n_k n_{k-1} \dots n_1$        $n_i = 0$  or  $n_i = 1$

$$n = \sum_{i=1}^{i=k} n_i F_{i+1}$$

דוגמאות:

Basis elements: 128 64 32 16 8 4 2 1  
 $73 = 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1$

Basis elements: 128 64 32 16 8 4 2 1  
 $45 = 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1$

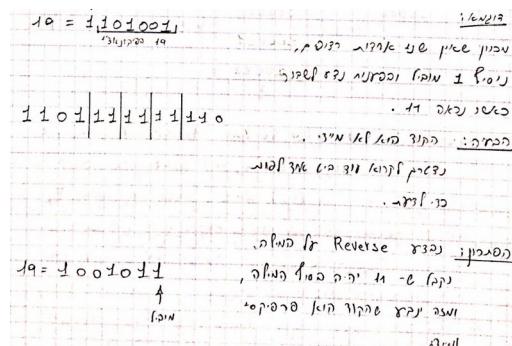
Fibonacci: 55 34 21 13 8 5 3 2 1  
 $73 = 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0$

Fibonacci: 55 34 21 13 8 5 3 2 1  
 $45 = 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0$

נשים לב:

- סכום של מספר פיבונacci' הוא סכום של אחדות ואפסים. או שמספר פיבונacci' משתתף בסכום או שלא.
- אין לנו שני אחדות רצופים ביצוג פיבונacci'. נשתמש בתוכנה זאת עבור הדחיסה.
- "יצוג פיבונacci' ידרש יותר סיביות כדי ל"ציג מספר מאשר "יצוג בינארי". "יצוג בינארי גדול מהר יותר ולכן נדרש פחות סיביות.

עד עכשוי דיברנו על "יצוג פיבונacci' של מספר , וכעת נדבר על קוד פיבונacci'.



## סיכום קורס דחיסת נתונים א'

נכתב על ידי: שי נאור <https://github.com/shaynaor>

מבוסס על הרצאות והמצגות של פרופ' דינה שפירא , מחברת של שובל

| <i>i</i> | Fibonacci Binary Representation | Fibonacci Code |
|----------|---------------------------------|----------------|
| 1        | 1                               | 11             |
| 2        | 10                              | 011            |
| 3        | 100                             | 0011           |
| 4        | 101                             | 1011           |
| 5        | 1000                            | 00011          |
| 6        | 1001                            | 10011          |
| 7        | 1010                            | 01011          |
| 8        | 10000                           | 000011         |
| 9        | 10001                           | 100011         |
| 10       | 10010                           | 010011         |

דרך שcola להגדיר קוד פיבונאצ'י היא להגדיק שנקח את כל המחרוזות שמסתיימות עם "11", ובנוסף אין שני אחדות סמוכים במליה. וכך בעצם נקבל סדרת מחרוזות כפי שהגדירנו בעמוד הקודם. נשים לב שבכל bucket נקבל מספר מיילים בגודל מסווד פיבונאצ'י. זה קורה מכיוון שאחת התכונות של פיבונאצ'י היא שיש לנו  $F_k$  מיליות קוד באורך 1 +  $k$  (כולל התוספת של -1), באלגוריתם שלנו אנחנו לוקחים כל פעם את המספר הפיבונאצ'י היכי גדול שנכנס למספר, אז אם  $j$  מספר קלשו מק'ם  $F_{k+2} > j = > F_{k+1}$  נרצה ש  $j$  יהיה גדול יותר מ  $F_{k+1}$  אבל לא הדול מהמספר הבא בתור. וכך לכל המספרים  $F_{k+2}$  יהיה להם "1" במקום שמתחאים לו  $F_{k+1}$ . מספר האיברים האלה הוא  $F_{k+2} - F_{k+1}$ .

קיבלנו קוד שהוא פרפקטי כי "11" מופיע רק בסוף של מילת קוד. וכך הוא גם מייד.

אחדות: פיבונאצ'י קוד עמיד במצב הוספה או השמטה של סיבית. אפשר לכל היוטר לאבד 3 מיליות קוד בעקבות הוספה/השמטה סיבית.

### Fibonacci Variant – Fib2

אפשרה נוספת לקוד פיבונאצ'י.  
שנים 2 דרכם לייצור את Fib2:

- דרך 1:

- נוריד את ה "1" שהוספנו.
- נוריד את כל מיליות הקוד המתחילה ב "0".

- דרך 2:

- נוריד את ה "1" שהוספנו.
- נוסיף "10" לכל מילת קוד.
- נוסיף מילת קוד חדשה "1" כמיילת הקוד הראשונה.

שתי הדרכים שקולות.

| <i>i</i> | Fib1   | Fib2    |
|----------|--------|---------|
| 1        | 11     | 1       |
| 2        | 011    | 101     |
| 3        | 0011   | 1001    |
| 4        | 1011   | 10001   |
| 5        | 00011  | 10101   |
| 6        | 10011  | 100001  |
| 7        | 01011  | 101001  |
| 8        | 000011 | 100101  |
| 9        | 100011 | 1000001 |
| 10       | 010011 | 1010001 |

- כל מילת קוד מתחילה ב "1" ומסתיימת ב "1" ואין רצף "11" בתוך מילת קוד. הקוד הוא לא פרפקטי, אבל עדין ניתן לפענוח בצורה ייחודית כי עדין ישנה הפרדה בין מילה למילה.
- יש  $F_{k-2}$  מיליות קוד באורך  $k$ .
- ניתן להוכיח כי Fib2 תמיד יותר טוב מ Fib1 גמא אבל פחות טוב מ C דלתא.