

אוניברסיטת אריאל  
המחלקה למדעי המחשב

**שם הקורס: דחיסת נתונים א'**

מס' קורס: 7060410

מרצה: פרופ' דנה שפירא

העורכים: משה חנוקוגלו, אביב רובשיץ, עדי אחואל, הדר מרגלית, נעמה שטאובר, מירב סלמה, יצחק שרון, אוראל שלום, רעות דביר

תאריך: תשע"ט סמ' א

קובץ זה הינו דף נוסחאות בקורס דחיסת נתונים א'.

דף נוסחאות זה כולל הגדרות, כללים ואלגוריתמים.

**אורך קוד ממוצע**  $E(C, P) = \sum_{i=1}^n p_i |c_i|$

כל מחרוזת אינסופית למחצה (מקרא complete), ניתן לפענח בצורה יחידה.

**קוד מידי (Instantaneous)**: קוד יקרא מייד אם המפענח יודע מיד שמילה נגמרה (ללא צורך לעבור לאות הבאה); מיידיות אינו תנאי הכרחי ל UD; **קוד פרפקטי**  $UD \Leftarrow$  **אלגוריתם לבדיקת UD**:

- בבני רשימה של כל מילות הקוד
  - אם קיימות ברשימה שתי מילים שאחת רישא את השנייה אזי הוסף לרשימה את ה dangling שלהם (אם הוא לא קיים כבר ברשימה)
  - חזור על תהליך 2 עד: אם קיבלת dangling שווה למילה מה codewords –הקוד לא UD אם אין יותר danging חדשים (שלא הופיעו כבר) –הקוד הוא UD
- הגדרה**: קוד בעל יתירות מינימלית: עבור  $E(C, P)$  או  $C$  יקרא קוד בעל יתירות מינימלית אם אין עוד קוד  $C'$  שעבורו מתקיים  $E(C', P) < E(C, P)$  עבור כל  $n$  מילות הקוד. נסמן את אורך מילה הקוד המינימלית בביטים לתי  $s_i$  כ:  $I(s_i) = -\log_2(p_i)$
- גמיה אנטרופיה**:  $H(P) = -\sum_{i=1}^n p_i \log_2 p_i$  ובלכל מקרים  $H(P) \leq E(C, P)$ .
- משפט**: עבור קוד  $C$  עם הסתברות  $P$  את  $C$  הינו  $UD$  אתי מקיים:

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

**משפט**: אם  $|C| > |C'|$ ,  $C'$  קוד פרפקטי.

**משפט**: אם  $|C| > 1$  אזי הקוד הוא לא יכול להיות פרפקטי

קוד פרפקטי  $\Leftrightarrow$  קוד מידי

**משפט**: קיים קוד מידי עם מילות קוד באורך  $l_1, \dots, l_n \Leftrightarrow \sum_{i=1}^n 2^{-l_i} \leq 1$

**קוד הוא שלם** אם  $\sum_{i=1}^n 2^{-l_i} = 1$

**קוד טטטי**:  $p_i = \frac{1}{|S|}$ ; תיים אפשריים ( $ascii$ )  $\Sigma$  (ההסתברות לא תלויה בהודעה שנחשפת)

**קוד ממי סטטי**:  $p_i = \frac{1}{|S|}$ ; התיים בטקסט הזה  $\Sigma$

**Header**: 8 ביט לייצוג כמות התווים ועוד 8 ביט לכל תו.

**ממי סטטי והסתברויות עצמיות**:  $p_i = \frac{s_i}{m}$  כאשר  $s_i$  זה מספר ההופעות של תו  $s_i$  בטקסט בגודל  $m$

**Header**: 8 ביט לייצוג כמות התווים ועוד 8 ביט לכל תו, 4 (לפי דוגמא במצגת) ביט לייצוג השכיחות של כל תו.

**קודים מסדר ראשון**: ההסתברות של כל תו תלויה בתו הקודם לו.

**Unary code**: מספר  $X$  מיוצג ע"י  $X-1$  ביטים של '1' ובסוף ביט של '0'

**Binary code**: קוד פשוט-כל סימן מיוצג-כל סימן מיוצג קוד שאורכה בדיוק  $\lceil \log_2 n \rceil$

**קוד מינימלי**-עבור א"ב בגודל  $n$ , מכיל  $n - 2^{\lfloor \log_2 n \rfloor}$  מילות קוד באורך  $\lceil \log_2 n \rceil$  ביטים ו  $2n - 2^{\lfloor \log_2 n \rfloor}$  באורך  $\lceil \log_2 n \rceil$

**Elias code**

$C_\gamma$  -חלק ראשון: קוד אונרי של מספר הביטים לייצוג  $X$

חלק שני: קוד בינארי של  $X$  ללא ה'1' הראשון

$C_\delta$  -חלק ראשון:  $C_\gamma$  של מספר הביטים לייצוג  $X$

חלק שני: קוד בינארי של  $X$  ללא ה'1' הראשון

ניתן לעשות זאת בצורה קורסטיבית ולהשתמש ב  $C_\delta$

**Golomb code**

$Golomb\_encode(x, b)$

$q \leftarrow (x-1) \div b;$

$r \leftarrow x - q \cdot b;$

$l = Unary\_encode(q+1);$

$n = Minimal\_binary\_encode(r, b);$

return l+n

$Golomb\_decode(b)$

$q \leftarrow Unary\_decode(l)-1;$

$r \leftarrow Minimal\_binary\_decode(b);$

return r+q\*b;

**Rice code**

בבחר את  $b$  להיות  $2^k$  עבור  $k$  כלשהו.

חלק ראשון: קוד אונרי של  $(1+x-1)$  (right shift  $k$  bits) של  $x-1$

חלק שני:  $k$  הביטים הנמוכים של הייצוג הבינארי של  $x-1$

**Shannon-Fano Algorithm**: (לא תמיד הכי יעיל)

נמייין את ההסתברויות בסדר יורד.

כל עוד קבוצת ההסתברות מכילה יותר מ-1:

נחלק את הקבוצה ל-2 חלקים כך שסכום ההסתברויות בכל חלק פחות או יותר זהה.

קבוצה ראשונה תקבל 1, והשנייה 0.

נאסוף את הסיביות שהקצנו בפצול כדי לבנות את הקוד עבור כל תו.

**קוד הפנו**:

**משפט**: עבור תווים עם משקולות  $w_1, \dots, w_n$  אלגוריתם הפנו מייצר מילות קוד באורכים  $l_1, \dots, l_n$  כך  $\sum_{i=1}^n w_i l_i$  מינימלי.

**משפט 1**: כל קוד אופטימלי מיוצג ע"י עץ מלא (קדקוד הוא או אבא לשני ביטים)

**משפט 2**: בעץ אופטימלי שתי המשקולות הקטנים ביותר נמצאות ברמה הנמוכה ביותר.

**משפט 3**: בעץ אופטימלי שתי המשקולות הנמוכים ביותר יכולים להיות אחים.

בעץ הפנו קבוצת נמוכה למפענח את הקוד של כל התווים (לפי העץ הקבוצה) וכן טבלה שמראה מה התו הראשון בכל אחד מהאורכים וכן מה הקידוד שלו.

**אלגוריתם לבניית הטבלאות שמורים למפענח**:

אחרי הרצת הפנו קיבלנו את האורכים של כל אחת מהמילים (נסמנו  $l_1, \dots, l_n$ )

- האורך של המילה הארוכה ביותר  $\leftarrow maxlength$ .
- for  $i = 1$  to  $maxlength$  do  $\{num[i] = 0\}$
- for  $i = 1$  to  $n$  do  $\{num[l_i] + 1\}$
- $firstcode[maxlength] \leftarrow 0$
- for  $i = maxlength - 1$  downto 1 do
- 5.1.  $firstcode[i] \leftarrow (firstcode[i + 1] + num[i + 1]) / 2$
6. for  $i = 1$  to  $maxlength$  do
- 6.1.  $nextcode[i] \leftarrow firstcode[i]$
7. for  $i = 1$  to  $n$  do
- 7.1.  $codeword[i] \leftarrow nextcode[l_i]$
- 7.2.  $symbol[l_i, nextcode[l_i] - firstcode[l_i]] \leftarrow i$
- 7.3.  $nextcode[l_i] + 1$

**אלגוריתם לפענוח של הפנו קבוי ע"י הטבלאות**:

קבלת הביט הראשון מהקובץ החוס/()  $nextInputBit()$

- $v = 1$
- while  $v < firstcode[i]$  do
- 3.1.  $v = nextInputBit()$
- 3.2.  $i + 1$

אם יצאנו מהלולאה אזי  $v$  מכילה מילה קוד תקינה//

4. return  $symbol[l, v - firstcode[l]]$

**משפט**: בעץ  $D$ -ary שלם שיש לו  $k$  צמתים פנימיים ישנם  $(D-1)k^*$  עלים.

**משפט**: בעץ אופטימלי  $D$ -ary מתקיים כמו אופטימלי גייל ובנוסף ישנם לפחות 2 צמתים ברמה הנמוכה ביותר.

**אלגוריתם להפנו D-ary**:

1. הוסף עוד  $x$  צמתים עם הסתברות 0 לרשימה ההתחלתית של הצמתים (בגודל  $n$ ) כך שיתקיים  $k + (D - 1) + x + n = 1$  (כאשר  $k$  זה מספר צמתים פנימיים)

2. כל עוד יש יותר מצומת אחד ברשימה, מזג את ה  $D$  צמתים עם ההסתברויות הנמוכות ביותר ותיצור צומת חדש עם הסכום שלהם (בדומה להפנו גייל)

3. תיתן ערך שרירותי בין 0 ל  $D-1$  לכל אחת מהקשתות מהצומת החדש לכל הצמתים שמוזגו בשלב הקודם

4. אחרי שסיימנו למזג את כל הצמתים, מילת קוד של כל תו היא רצף של מספרים (בטווח  $D-1$ ) שמיצגים את המסלול מהשרוש עד לעלה ששייך לו זה.

**הפנו דינמי (עץ בינארי ולא D-ary)**:

**משפט**: עץ מקיים את Sibling Property  $\Leftrightarrow$  העץ הוא עץ הפנו

**אלגוריתם לעדכון עץ הפנו דינמי**:

1.  $q = leaf(x_i)$

2. 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)

החליף את  $q$  עם העלה עם המספר הכי גבוה שיש לו את אותו המשקל

3.2. תעלה את הערך של המשקל של  $q$  באחד.

3.3.  $q = parent\ of\ q$

4. while ( $q \neq root$ )

החליף את  $q$  עם העלה עם המספר הכי גבוה שיש לו את אותו המשקל

4.1. תעלה את הערך של המשקל של  $q$  באחד.

4.2.  $q = parent\ of\ q$

4.3.  $q = parent\ of\ q$

4.4. תעלה את הערך של המשקל של  $q$  באחד.

**אלגוריתם לעץ הפנו**:

- Find a Huffman tree with lengths  $l_1, \dots, l_n$  for the items.
- Sort the items according to their lengths.
- Assign to each item the first  $l_i$  bits after the binary point of  $\sum_{j=1}^{i-1} 2^{-l_j}$

**הגדרות לאלגוריתמים של עץ הפנו**:

$base(i) = 2^{(base(i-1) + n_{i-1})}$

$seq(i) = seq(i-1) + n_{i-1}$

$diff(i) = base(i) - seq(i)$

**Decoding Algorithm**

- tree\_pointer <- root
- $i < 1$
- start <- 1
- while  $i < length\_of\_string$
- if  $string[i] = 0$
- 4.1.1. tree\_pointer <- left(tree\_pointer)
- 4.2. else tree\_pointer <- right(tree\_pointer)
- 4.3. if  $value(tree\_pointer) > 0$
- 4.3.1. codeword <- string[start ... (start+value(tree\_pointer)-1)]
- 4.3.2. output <- table[(codeword)-diff[value(tree\_pointer)]]
- 4.3.3. tree\_pointer <- root
- 4.3.4. start <- start+value(tree\_pointer)
- 4.3.5.  $i < start$
4. else  $i++$

**אלגוריתם לעץ הפנו reduced**

- tree\_pointer <- root
- $i < start - 1$
- while  $i < length\_of\_string$
- 3.1. if  $string[i] = 0$  tree\_pointer <- left(tree\_pointer)
- 3.2. else tree\_pointer <- right(tree\_pointer)
- 3.3. if  $value(tree\_pointer) > 0$
- 3.3.1.  $len < value(tree\_pointer)$
- 3.3.2. codeword <- string[start... (start+len-1)]
- 3.3.3. if  $flag(tree\_pointer) = 1$  and  $2 \lfloor (codeword) \rfloor \geq base(len+1)$
- 3.3.3.1. codeword <- string[start... (start+len)
- 3.3.3.2.  $len++$
- 3.3.4. output <- table[(codeword)-diff[len]]
- 3.3.5. tree\_pointer <- root
- 3.3.6.  $i < start + start + len$
- 3.4. else  $i++$

**אלגוריתם לקידוד ע"י קוד אריתמטי בטווח [0,1]**:

$low \leftarrow 0.0$ ;  $high \leftarrow 1.0$

while input symbols remain

range  $\leftarrow high - low$

Get symbol

$high \leftarrow low + high\_bound(symbol) * range$

$low \leftarrow low + low\_bound(symbol) * range$

Output any value in [low, high]

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

**הגדרה**:  $high\_bound(s_i) \leftarrow \sum_{j=1}^i p_j$

**אלגוריתם לפענוח קוד אריתמטי בטווח [0,1]**:

encoded  $\leftarrow$  Get (encoded number)

do{ Find symbol whose range contains encoded

Output the symbol

range  $\leftarrow high(symbol) - low(symbol)$

encoded  $\leftarrow (encoded - low(symbol)) / range$

} until (EOF)

**קוד אריתמטי אדפטיבי**:

$N(x)$  זה השכיחות של  $x$  בטקסט שנקרא עד עכשיו (רק החלק שכבר פוענח).

$p(a) = \frac{N(a)+1}{|S| + \sum_{y \in S} N(y)}$

**אלגוריתם קוד אריתמטי אדפטיבי**:

כל פעם שפענחנו תו חדש, נקדם את השכיחות של אותו התו באחד, נעדכן את ההסתברויות ונאז נעדכן את הטווח לפי:

$high \leftarrow low + high\_bound(symbol) * range$

$low \leftarrow low + low\_bound(symbol) * range$

כאשר ההסתברות של כל תו משתנה בקריאת כל תו חדש ובסוף הפלט הוא מספר בטווח (low, high)

**אלגוריתם אריתמטי עם מספרים שלמים**:

1. בחר מספר לזמנות ספרות לטווח והגדרו כרצף '0' בגודל המספר שברמת high כמו low ורק עם '0'

2. חשב כמו באריתמטי גייל את ה low, range, high בהתאם לתו שקראת

3. אם הסיפורה השמאלית ביותר ב low ו high שווה, הוצא אותה לפלט והכנס במקומה

מחד ימין למספר high '9' ו low '0'

4. אם הוצא השמאלי ביותר ב low ו high הם הברפש של 1 וגם התו השני ב high הוא '0' ובלאו הוא '9' אז תחמק את ה '9' או ה '0' האלו, תבצע זהה שמאלה של שאר הספרות והכנס high למימין '9' ו low '0' ובנוסף תעלה את המונה ב 1.

5. חזור על שלב 4 עד שלא נבחרים יותר '0' ו '9', ואז הוצא את המספר הקרוב ביותר ולאחריו הוסף כמות של המונה בהתאם ל high - '0' ו low - '1'

**Incremental Coding**:

בחר את התת אינטרוול המתאים לסימון קלט האב

חזור על השלבים הבאים:

- אם התת האינטרוול החדש לא לגמרי מוכל באחד מהאינטרוולים הבאים  $(0, 1/2)$ ,  $(1/4, 3/4)$ ,  $(1/2, 1)$  הבא בתור.
- אם התת אינטרוול החדש נמצא כולו בתוך  $(0, 1/2)$  הוצא 0 ובפלט (ובמידת הצורך הוצא '1' לפי המונה, אם היינם בשהייה) והכפל את הטווח
- אם התת אינטרוול החדש נמצא כולו בתוך  $(1/2, 1)$  הוצא 1 (ובמידת הצורך הוצא '0' לפי המונה, אם היינם בשהייה) והכפל את הטווח .
- אם התת אינטרוול החדש נמצא כולו בתוך  $(1/4, 3/4)$ , מקדמים את המונה ב 1, המשך בתהליך ובדוק בהמשך מה צריך להוציא והכפל את הטווח.

**LZ77 – דחיסה אדפטיבית**:

(התו הבא, אורך העתקה, כמה תוים ללכת אחורה)

**אלגוריתם קידוד LZ77**:

- $p \leftarrow 1 //$  The next character to be coded
- while there is text remaining to be coded
- 2.1. search for the longest match for  $S[p..]$  in  $S[p-W\dots p-1]$  suppose that the match occurs at position  $m$  with length
- 2.2 Output the triple  $(p-m, l, S[p+m])$
- 2.3  $p \leftarrow p+m+1$

**אלגוריתם פענוח LZ77**:

- $p \leftarrow 1 //$  The next character to be decoded
- For each triple  $(f, l, c)$  in the input
- 2.1  $S[p\dots p+l-1] \leftarrow S[p-f\dots p-f+l-1]$
- 2.2  $S[p+l] \leftarrow c$
- 2.3  $p \leftarrow p+l+1$

**הכנת עצמת** – כאשר  $offset > length$

**LZ78**: דחיסה מילוגית כאשר מקודדים כוונות (התו הבא, הפניה לאינדקס במילון)

**LZW**: מבצע עם שימוש בעץ trie.

Dictionary  $\leftarrow$  single Characters

$w \leftarrow$  first char of input

repeat

$k \leftarrow$  next char

if (EOF) output code( $w$ )

else

if  $(w \cdot k) \in$  Dictionary then  $w \leftarrow w \cdot k$

else

output code( $w$ )

Dictionary  $\leftarrow w \cdot k$

$w \leftarrow k$

**אלגוריתם פענוח ל LZW**:

Initialize table with single character strings

OLD = first input code

output translation of OLD

while not end of input stream

NEW = next input code

if NEW is not in the string table

$S =$  translation of OLD

$S = S \cdot C$

else

$S =$  translation of NEW

output S

$C =$  first character of S

Translation(OLD)  $\cdot C$  to the string table

OLD = NEW

**Re-Pair (Larsson & Moffat)**

מצא את הזוג התווים הסמוכים שמופיעים הכי הרבה פעמים.

תצור חוק: "אות גדולה" מצביעה לזוג התווים הקטנים שנמצאו בשלב הקודם.

החליף כל מופע של זוג התווים שנמצאו בשלב הראשון ב"באות הגדולה".

חזור על התהליך עד ששכל זוג מופיע פעם אחת בלבד.

**Tunstal code Algorithm**

$Tunstal(n) \{$

$D \leftarrow \Sigma$

while  $(|D| \leq 2^n - |\Sigma|) \{$

let  $d \in D$  be with **highest probability**

$D \leftarrow D \cup (U_{\sigma \in \Sigma} d\sigma)$

if  $d \notin \Sigma$

$D \leftarrow D - \{d\}$

**קוד פיבונאצ'י**:

**הכונות**:

- $F_k$  מילים באורך  $k + 1$
- מספר  $j$  המקיים  $F_{k+1} \leq j \leq F_{k+2}$  זקוק ל  $k + 1$  ביטים.
- $F_k = F_{k+1} - F_{k+2}$  (מסמל את המספר הפיבונאצ'י ה  $k$ )
- פרפקטי  $\leftarrow$  קוד מידי.

**Fib2 - רשימה של Fib1**

- השמט את ה-1 הימני ביותר בכל מילת קוד
- מחק את כל מילות הקוד המתחילות ב 0 שקולו ל:
- השמט את ה-1 הימני ביותר בכל מילת קוד
- הוסף 10 לכל תחילת מילה
- הוסף 1 במילת הקוד הראשונה

ב Fib2 יש מילת קוד אחת באורך  $1 + F_{k-2}$  מילות קוד באורך  $k$ .