2/11 ⊖ בס"ד

נושא 3 - קידודים סטטיים

א) הגדרה

אלו קידודים שהשימוש שלהם בהסתברות הוא מינימלי. הם לא לומדים את הקובץ לפני הדחיסה ואין צורך לאחסן ה-header (prelude) מאוד קטן והדחיסה מהירה, אך מצד שני הדחיסה פחות איכותית אם מניחים שהא"ב (Σ) יכול להיות גם אינסופי.

⊛ כל הקידודים שנלמד בקורס כה הם Lossles, כלומר הקובץ המקורי והפענוח הם זהים.

ב) קוד אונארי

זהו קידוד שבכל מילת קוד s_i נותן קידוד באורך i המכיל $i-1$ אחדות ולאחריהם 0 המציינים את סוף הקידוד. קידוד עבור מילת קוד כללית: $S_i = 1^{i-1}0$, אם הא"ב סופי ניתן להוריד את ה-0 של התו האחרון.

0
10
110
1110
11100

שאלה: מתי קוד אונארי הוא "קוד ללא יתירות"?

תשובה: כאשר לכל מילת קוד s_i ההסתברות היא $p_i = \frac{1}{2^i}$, נקבל שהקידוד הוא חסר יתירות. שכן במצב כזה כמות האינפורמציה של s_i הוא: $I(s_i) = -\log_2(\frac{1}{2^i}) = i$, שזה הדיוק אורך הקידוד של s_i בקידוד אונארי.

ג) קוד בינארי

נחלק לשני סוגים:

קוד בינארי פשוט - אם יש n מילות קוד אזי כל מילת קוד תקבל קידוד באורך קבוע $\lceil \log_2 n \rceil$. כל מילת קוד תקבל קידוד של מספר בינארי ~~[illegible]~~ בסדר עולה. כלומר כל מילת קוד s_i תקבל את הקידוד i_2 (המספר i הבינארי) כאשר מרפדים מספר זה ב-0 משמאלו עד שיגיע לאורך $\lceil \log_2 n \rceil$.

קוד בינארי מינימלי - כמו הגרסה הפשוטה אלא שאם n לא חזקה של 2, $2^i < n < 2^j$, אזי $2^j - n$ התווים הראשונים יקבלו קידוד באורך i, ושאר מילות הקוד, שמספרם $2n - 2^j$, יקבלו את הקידוד הארוך באורך j. קידוד זה הוא חסר רישות ו-UD.

$i = \lfloor \log_2 n \rfloor$
$j = \lceil \log_2 n \rceil$

שאלה: נניח ש-n הוא חזקה של 2, $n = 2^k$, אזי מתי קידוד בינארי מינימלי הוא חסר יתירות?

תשובה: כאשר ההתפלגות לכל מילות הקוד היא אחידה, $p_i = \frac{1}{n}$ לכל i, נקבל שהקידוד חסר יתירות. שכן במצב כזה כמות האינפורמציה של s_i הוא: $I(s_i) = -\log(\frac{1}{n}) = -\log(\frac{1}{2^k}) = k$ שזהו הדיוק האורך של הקידוד של כל מילות הקוד.

כדי לעבור מעץ של קידוד בינארי פשוט לעץ של קידוד בינארי מינימלי, יש לבחור את $2^j - n$ הצמתים השמאליים בדרגה אחת לפני אחרונה בעץ ולמחוק את הבנים שלהם. צמתים אלו הם $2^j - 3$ הקידודים הקצרים של התווים שייצגו בשיטת קידוד בינארי מינימלי.

3) קידודי אליאס (Elias Codes)

לפני שמקודדים את התווים מסדרים אותם בסדר יורד לפי התדירות שלהם. לכל תו n נתייחס אליו כמספר n ואז נקודד את n. הקידודים אלו ככל שתו עם תדירות גבוהה יותר כך יקבל קידוד קטן יותר.

יש שני סוגים של קידודים: C_γ (c גמא), ו-C_δ (c דלתא).*

C_γ: הקידוד עבור כל תו n מבוצע בשני שלבים. בשלב הראשון מייצגים את n בבינארי. אורך ייצוג זה הוא $\lfloor\log_2 n\rfloor+1$. בשלב השני מייצגים את אורך הייצוג הבינארי באונארי. הקידוד של n יהיה קידוד הייצוג האונארי שקיבלנו בשלב השני ולאחריו אורך הייצוג הבינארי שקיבלנו בשלב הראשון, אך שהורדנו ממנו את הביט 1 בהתחלה, שכן כל מספר בינארי מתחיל ב-1 ולכן המפענח יוכל להוסיף אותו אוטומטית. אורך הייצוג הוא $\lfloor\log_2 n\rfloor+1$ עבור הקידוד האונארי ו-$\lfloor\log_2 n\rfloor$ עבור הקידוד הבינארי ללא 1 מוביל.

סה"כ $|C_\gamma(n)| = 1+2\lfloor\log_2 n\rfloor = O(\log_2 n)$.

קידוד זה הוא UD שכן המפענח עובר על האחדות עד שמגיע ל-0 ואז יודע את אורך הייצוג l. לאחר מכן לוקח את l הביטים הבאים מוסיף להם 1 ומקבל את הייצוג הבינארי של n. נשים לב שקידוד זה אינו חסר רישות (דוגמה נגדית $C_\gamma(2)$ הוא רישא של $C_\gamma(4)$).

<u>דוגמאות:</u>

$50_2 = \overbrace{110010}^{6\text{ ביטים}}$, $6_1 = 111110$, $C_\gamma(50) = 111110 \vdots 10010$

$30_2 = \overbrace{11110}^{5\text{ ביטים}}$, $5_1 = 11110$, $C_\gamma(30) = 11110 \vdots 1110$

בהודעה / קובץ בהם יש מספר קטן של תווים לא כדאי להשתמש בקידוד זה, אך במקרה שיש הרבה תווים קידוד זה יעיל יותר מקידוד אונארי או בינארי.

<u>הערה:</u> C_γ נקרא גם "exponential and binary search", שכן ניתן לפענח את המספר באמצעות שיטה הדומה לחיפוש בינארי. עבור כל ביט בקידוד נשאל שאלה, כאשר ביט 1 אומר שהתשובה כן וביט 0 התשובה היא לא. בחלק הראשון של הקידוד האונארי נשאל לכל ביט k האם n גדול מ-2^k. בחלק השני נשאל האם קטן מהחצי של המספר אליו הגענו כמו חיפוש בינארי. בסוף סדרת השאלות נגיע למספר באופן "חוקי".

C_δ: כמו C_γ אלא שבחלק הראשון במקום לייצג את אורך הייצוג הבינארי בקידוד אונארי נייצג אותו באמצעות C_γ. המפענח יוכל להבין מהו אורך הייצוג הבינארי כך שכן C_γ הוא UD.

אורך הייצוג הבינארי ב-C_γ והוא: $1+2\lfloor\log_2(1+\lfloor\log_2 n\rfloor)\rfloor = 1+2\lfloor\log_2(\log_2 2+\lfloor\log_2 n\rfloor)\rfloor = 1+2\lfloor\log_2\log_2 2n\rfloor$

סה"כ $C_\delta(n) = 1+2\lfloor\log_2\log_2 2n\rfloor+\lfloor\log_2 n\rfloor$. קידוד זה הוא גם כן UD אך אינו חסר רישות.

<u>דוגמאות:</u>

$30_2 = \overbrace{11110}^{5\text{ ביטים}}$, $5_2 = \overbrace{101}^{3\text{ ביטים}}$, $3_1 = 110$, $C_\gamma(5) = 110 \vdots 01$, $C_\delta(30) = 11001 \vdots 1110$

$143_2 = \overbrace{10001111}^{8\text{ ביטים}}$, $8_2 = \overbrace{1000}^{4\text{ ביטים}}$, $4_1 = 1110$, $C_\gamma(8) = 1110 \vdots 000$, $C_\gamma(143) = 1110000 \vdots 0001111$

ההבדל בין C_γ ל-C_δ אינו משמעותי במספרים קטנים אך במספרים גדולים שאורך הייצוג הבינארי שלהם מאוד גדול, C_δ יציג שיפור משמעותי שכן הייצוג ב-C_γ הוא הרבה יותר טוב מייצוג אונארי

לדוגמה, עבור 10^9 (מיליארד) נקבל כי C_δ חוסכת 21 ביטים על פני C_γ.
אך עבור קבוצה מצומצמת של מספרים, שהיא 2-3 ו-8-15, דווקא נקבל ש-C_γ טוב יותר מ-C_δ.* הסיבה לכך היא ש-C_γ ארוך יותר מקידוד אונארי עבור $n=2$ ו-$n=4$.

*מעבר 3 שקופית 11 ניתן לראות זאת.

<u>רעיון:</u> נוכל להמשיך את המעבר מ-C_γ ל-C_δ בכך שגם את החלק הראשון של C_δ, שהוא C_γ של אורך הייצוג הבינארי, גם אותו נעביר ל-C_δ, וכן הלאה בצורה רקורסיבית.* האם זה ישפר את הקידוד? מסתבר שאכן יהיה שיפור אבל רק עבור מספרים שמבחינתנו הם אינסופיים. לדוגמה עבור 10^9 לא יהיה הבדל בכלל.

*קידוד זה נקרא C_ϵ (c אפסילון).

ה) קוד אוניברסלי

ראינו האנטרופיה של התפלגות שהקידוד הקצר ביותר שניתן לתת לתו s_i הוא כמות האינפורמציה שלו $I(s_i)=-\log_2 p_i$. למעשה אם נסדר את כל התווים לפי תדירות שלהם $p_1\ge p_2\ge\ldots\ge p_n$ נקבל שתמיד מתקיים $p_x\le\frac{1}{x}$ לכל $1\le x\le n$. שהרי אם יש $p_x>\frac{1}{x}$ אזי נקבל שסכום כל ההסתברויות עד s_x, שגם כן גדולות מ-$\frac{1}{x}$, נקבל $\sum_{j=1}^{x}p_j>\sum_{i=1}^{x}\frac{1}{x}=1$, סתירה שכן סכום ההסתברויות לא יכול להיות גדול מ-1.

כעת נחשב חסם לכמות האינפורמציה $I(s_x)=-\log_2 p_x\le-\log_2\frac{1}{x}=\log_2 x$. נמצא כי $\log_2 x$ הוא חסם איכותי לאורך קידוד של תו s_x.

<u>קוד אוניברסלי:</u> זהו כל קוד שאורכו הוא $O(\log_2 n)$ עבור מילת קוד s_n.

נשים לב שקידודי אליאס C_γ ו-C_δ הם קידודים אוניברסליים שכן ראינו בשלב קודם שאכן מספקים קידודים באורך זה.

$$\left.\begin{aligned}C_\gamma&:\ \log_2 n+\Theta(\log_2 n)\\ C_\delta&:\ \log_2 n+o(\log_2 n)\end{aligned}\right\}$$

ו) קוד גולומב (Golomb)

בקוד זה מחלקים את כל התווים לקבוצות בגודל b, כך שבכל קבוצה הקידוד של כל התווים באותו אורך. לקבוצות שונות יש קידודים באורכים שונים. הקידוד משתנה בהתאם ל-b שאנו בוחרים.
האלגוריתם באמצעותו מקודדים כל תו X הוא:

- הגדר $q=\frac{x-1}{b}$. זוהי חלוקה רק בשלמים, מתעלמים מהשארית.
- הגדר $r=x-qb$.
- צור את הקידוד האונארי של $q+1$ - $(q+1)_1$.
- חשב את הקידוד הבינארי המינימלי לקבוצת מספרים בגודל b. הגדר את הקידוד במקום r - $mbe(r,b)$.
- החזר $X_{Golomb}=(q+1)_1\,mbe(r,b)$, להיות הקידוד גולומב של X עבור b נתון.

הפענוח של x יעשה בדיוק בכיוון ההפוך

- פענח את החלק של הקידוד האונארי וחסר 1 לקבל את q.
- מתוך קבוצת הקידוד הבינארי מינימלי לקבוצת מספרים בגודל b, הגדר r להיות המיקום של $mbe(r,b)$.
- החבר $x=r+qb$ לקבלת מיקום התו המקורי.

__דוגמאות:__ עבור $b=4$ נחשב קידוד של 9.

$q=\frac{9-1}{4}=2$, $r=9-2\cdot4=1$, $3_1=110$, $mbe(4,4)=\{\overset{\text{מיקום 1}}{00},01,10,11\}$, $9_{Golomb}=110;00$

עבור $b=5$ נחשב קידוד של 7.

$q=\frac{7-1}{5}=1$, $r=7-1\cdot5=2$, $2_1=10$, $mbe(2,5)=\{00,\overset{\text{מיקום 2}}{01},10,110,111\}$, $7_{Golomb}=10;01$

ג) __קוד רייס (Rice)__

זהו מקרה פרטי של קוד גולומב. גם כן מחלקים את כל התווים לקבוצות שוות בגודל b, אלא ש-b תמיד תהיה חזקה כלשהי של 2, כלומר $b=2^K$ עבור $K\in N$ כלשהו. כאשר מגדירים את b בצורה כזו נוכל להשתמש באלגוריתם שיתן אותה תוצאה כמו גולומב רק מהיר יותר שכן במקום כפל וחילוק משתמש בפעולת shift. עבור $b=2^K$ האלגוריתם באמצעותו מקודדים כל תו x הוא:

- הפעל shift ימינה של k ביטים על הייצוג הבינארי של x-1. על המספר שהתקבל הוסף 1, ואז הפוך אותו לקידוד אונארי - $(x-1)'_1$
- קח את k הביטים הנמוכים של הייצוג הבינארי של x-1 - $(x-1)_{2,k}$.
- החבר $x_{Rice}=(x-1)'_1\,(x-1)_{2,k}$, להיות הקידוד רייס של x עבור k נתון.

__דוגמה:__ עבור $K=2$ $(b=4)$ נחשב קידוד של 9.

$(9-1)_2=1000 \xRightarrow{\text{shift ביטים 2}} 10 ; 2+1=3$, $3_1=110$, $(9-1)_2=10\underbrace{00}_{\text{ביטים 2 נמוכים}}$, $9_{Rice}=110;00$ (כמו 9_{Golomb})

⊕ נשים לב שעבור $b=1$ או $k=0$ נקבל את הקוד האונארי.

Scanned with CamScanner

ח) קוד שנון (Shannon)

ראינו באנטרופיה של התפלגות שהקידוד הקצר ביותר שניתן לתת לתו s_i עם התפלגות p_i הוא כמות האינפורמציה שלו: $I(s_i)=-\log_2 p_i=\log_2\frac{1}{p_i}$. המטרה של שנון בבניית הקידוד שלו היא להגדיר קוד C שניתן לכל תו s_i קידוד c_i שאורכו $|c_i|=\lceil\log_2\frac{1}{p_i}\rceil$, מאחר והאינפורמציה של s_i אינה בהכרח מספר שלם.

<u>משפט:</u> קוד המקיים את התנאי לעיל הוא אכן UD וגם יכול להיות חסר רישא.

הוכחה: נוכיח שהקראפט שלו קטן או שווה 1.

$$K(C)=\sum_{i=1}^{n}2^{-|c_i|}=\sum_{i=1}^{n}2^{-\lceil\log_2\frac{1}{p_i}\rceil}\le\sum_{i=1}^{n}2^{-\log_2\frac{1}{p_i}}=\sum_{i=1}^{n}p_i=1$$

<u>משפט:</u> בקוד המקיים את התנאי לעיל, בממוצע כל מילת קוד תהיה לכל היותר בביט אחד מהאנטרופיה. כלומר מתקיים: $H(P)\le E(C,P)<H(P)+1$.

הוכחה:

$$H(P)=\sum_{i=1}^{n}p_i\cdot\log_2\frac{1}{p_i}\le\sum_{i=1}^{n}p_i\cdot\lceil\log_2\frac{1}{p_i}\rceil=E(C,P)<\sum_{i=1}^{n}p_i\left(\log_2\frac{1}{p_i}+1\right)=H(P)+1$$

קוד שנון בונה עץ בינארי שבו הקידוד של כל תו s_i עם הסתברות p_i הוא בדיוק באורך $\lceil\log_2\frac{1}{p_i}\rceil$. אמנם ממשפט קודם הוכחנו שעץ זה הוא לא אופטימלי ולמעשה גם לא שלם (דוגמה בשקופית 8 במצגת), ולכן אינו בעל יתירות מינימלית ואפשר לשפר אותו.

ישנו רעיון תיאורטי שבו במקום לקודד את קבוצת התו"ים Σ, מקודדים בקוד שנון את קבוצת התו"ים Σ^n שבה מפורשים כל n תו"ים אפשריים. ניתן להניח שההסתברויות תלויות או בלתי-תלויות. במקרה כזה נקבל שקידוד מילה ממוצעת הוא קטן מאורך האנטרופיה ועוד $\frac{1}{n}$. $\frac{1}{n}L(s_1,\dots,s_n)<H(P)+\frac{1}{n}$ (הוכחה בשקופיות 9-10), אמנם אז אם הנחנו שהסתברויות תלויות נצטרך ב-Header לדעת את $|\Sigma|^n$ ההסתברויות, ואם הנחנו שההסתברויות לא תלויות נקבל קידוד נחות טוב מפני שאינו מתאים למציאות.

ט) קוד שנון - פאנו (Shannon-Fano)

זהו קוד המנסה לשפר את קוד שנון על ידי יצירת עץ בינארי מלא (שבו לכל הצמתים הפנימיים יש שני בנים). האלגוריתם של קוד זה יוצר קידוד של עץ בינארי מלא, שבו 1 מייצג צומת פנימית ו-0 עלה. לכל עץ מלא יש n-1 צמתים פנימיים ו-n עלים, לכן מספר הביטים הדרוש כדי לייצג אותו הוא 2n-1 (n-1 אחדות ו-n אפסים). תיאור האלגוריתם:

- סדר את כל התו"ים בקבוצה אחת לפי ההסתברויות בסדר יורד.
- כל עוד יש קבוצה המכילה יותר מתו בודד
- חלק כל קבוצה לשתי חלקים כך שסך ההסתברות בשניהם יהיו קרובים ככל הניתן.
- הוסף לקבוצה עם יותר איברים מתוך ה-2 שפיצלנו ערך 1 ולקבוצה הקטנה יותר ערך 0. אם הם באותו גודל אין חשיבות מי 0 ומי 1.

בכל איטרציה נחלק כל קבוצה ל-2 חלקים.

בס"ד

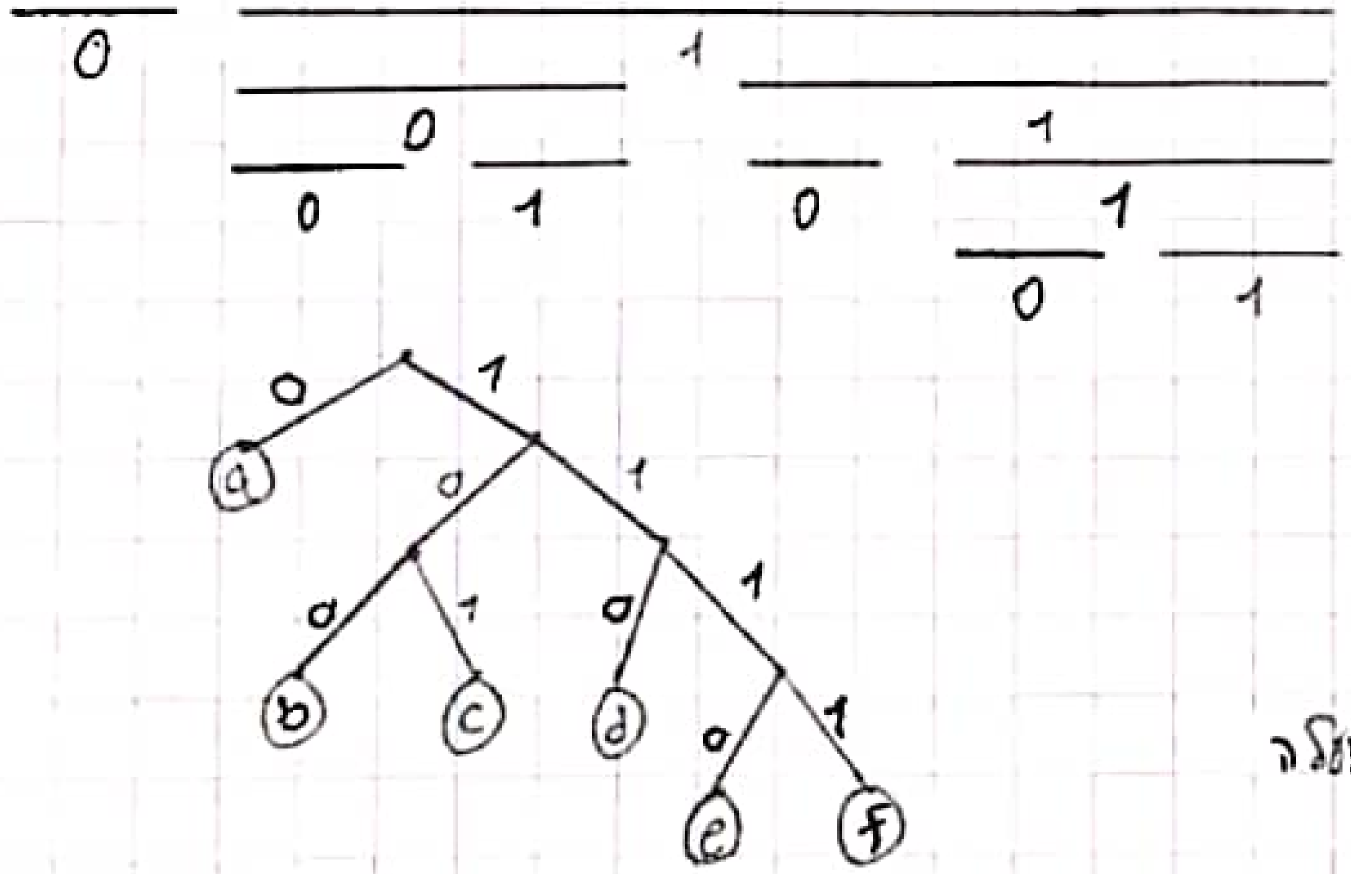
16/11

(12)

בשלב האלגוריתם נקבל קידוד של עץ בינארי מלא, המקיים קידוד חסר רישות. UD, מיידי ושלם.

דוגמה: עבור 6 תווים שההסתברויות שלהם מסודרים כך:

a	b	c	d	e	f
0.67	0.11	0.07	0.06	0.05	0.04



האלגוריתם של שנון-פאנו יתן לנו את קידוד העץ

הבינארי: (01 01 0101 01).

אופטימליות: קוד שנון-פאנו אינו אופטימלי. בשקיפית 15

ניתן לראות שיש קידוד c שהוא טוב יותר ממה שקיבל

שנון-פאנו נתן. קוד זה בונה את העץ הבינארי מלמעלה

למטה. בפרק הבא נלמד את קוד הופמן שהוא אופטימלי

ובונה את העץ מלמטה למעלה.