קורס דחיסת נתונים א – סמסטר א' 2020 מדעי המחשב

נכתב ע"י צבי מינץ

מרצה: פרופ' דנה שפירא

<u>הרצאה 1 – מבוא</u>

הערה: חלק מסיכום הרצאה 1 מבוסס על סיכום של אורנית כהן – oranit95

ציונים: 4 מטלות – 10% מהציון, שאר הבחינה.

דרישה: הזיכרון לא יקר כמו שהוא היה לפני הרבה שנים אבל בכל אופן יש הגבלות על רוחב הפס, אז או שנרחיב את רוחב הפס שזה לא תמיד אפשרי או להעביר יותר נתונים על אותו רוחב פס. ולכן יש צורך באלגוריתמי דחיסה.

מטרה: שיפור ביצועים גם מבחינת זמנים וגם מבחינת שטח אחסון.

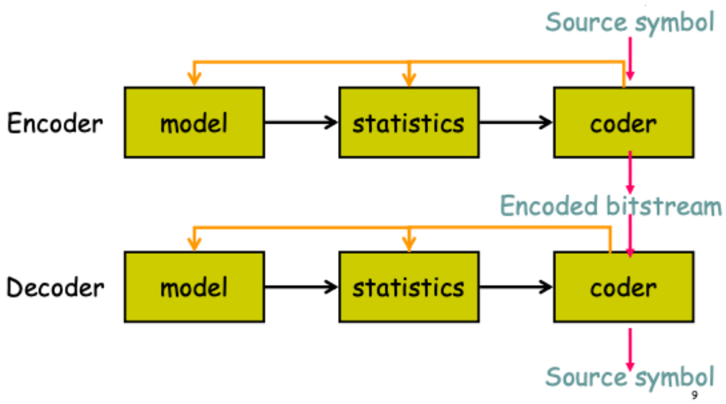
כל מערכת דחיסה מורכבת מ-3 פעולות:

1. שלב המידול – הנחות שעושים על המידע שאיתו אנחנו מתמודדים או שאוספים את המידע על הקובץ. על מנת שהקידוד והדחיסה יהיו מסונכרנים, הקידוד צריך להיות מוכר גם למקודד וגם למפענח.
2. איסוף סטטיסטיקה
3. הקידוד עצמו - רוב הקורס. צריך לדעת את קובץ המקור, מאילו אלמנטים מורכב הקובץ, כלומר:
   - א"ב המקור
   - א"ב ערוצי

לדוגמא: Unary Code: 0,10,110 וכו', כאשר בין כל מילת קוד יש שובר אשר הוא "0".

כך זה נראה:

יש מקודד (Encoder) ומפענח (Decoder) , בכל שלב אפשר לעדכן את המודל. העדכון צריך להיות מסונכרן עם מה שהמפענח עושה. לוקחים א"ב מהמקור (Source symbol), יוצרים את מילת הקוד והמפענח עושה את הפעולה ההפוכה וממיר את זה חזרה לקוד הרגיל (Source symbol).



טרמינולוגיה (המילים שבשימוש)

- **א"ב של קובץ המקור** $S := [s_1, s_2, \dots, s_n]$
- **הסתברות** $P = [p_1, p_2, \dots p_n]$ כך ש- $\sum_{i=1}^{n} p_i = 1$

  ניתן להניח כי $p_1 \geq p_2 \geq \cdots \geq p_n$ (הסתברויות בסדרה מונוטונית יורדת)
- **מילות קוד** $C = [c_1, c_2, \dots, c_n]$

  ככל שההסתברות גבוהה יותר כך מילת הקוד קצרות יותר
- **אורך מילות הקוד, כלומר כמה סיביות כל מילת קוד** $|C| = [|c_1|, \dots, |c_n|]$
- **אורך מילת הקוד הממוצעת** $E(C,P) = \sum_{i=1}^{n} p_i \cdot |c_i|$ - Expected codewords length

לדוגמא:

Example $E(C,P) = \sum_{i=1}^{n} p_i \cdot |c_i|$

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: $|C|=[3,3,\dots,3]$

קורס דחיסת נתונים א – סמסטר א' 2020 מדעי המחשב
נכתב ע"י צבי מינץ
מרצה: פרופ' דנה שפירא

יש 2 סוגי דחיסות:

- דחיסות שמאבדות מידע (**lossy** compression)
  אלגוריתמים של דחיסה אשר מאבדים חלקים מהמידע שהיה לפני הדחיסה.
  מיושם בד״כ על קבצי תמונות, ווידאו וקול.
- דחיסות שאינן מאבדות מידע (**lossless compression**)
  אלגוריתמים של דחיסה אשר מאפשרים לפענח את הדחיסה ולקבל **במדוייק** את הקובץ לפני הדחיסה, מיושם בדרך כלל על קבצי טקסט.

הערה: הדחיסה ופריסה צריכות להיות פונקציות הפוכות.

<u>קוד חסר רישות – Prefix-free codewords</u>

אף מילת קוד אינה רישא של מילת קוד אחרת, נאמר על קוד אשר מקיימת תכונה זאת ככקוד פרפיקסי.
קוד כזה מאשר מעבר ייחודי (UD) משמאל לימין.

לדוגמא:

$$\epsilon$$
$$0$$
$$01$$
$$011$$
$$0111$$

ניתן לייצג קוד זה ע״י עץ בינארי, כל צלע מיוצגת ב׳0׳ או ב׳1׳, כל עלה בעץ הינו תו כלשהו, כאשר המסלול בין שורש העץ לעלה מייצג את אותו התו, אורך המסלול הוא המסלול מהעץ לעלה.

יתרונות לקוד חסר רישות:

1. קל לקידוד ופענוח
2. ניתן לפיענוח בצורה יחודית UD
3. ניתן להוכיח כי כל דחיסת קוד אופטימלית אשר ניתן ע״י קוד לא חסר רישות אזי ניתן תמיד לדחוס בצורה זהה ע״י קוד חסר רישות ולכן ניתן להתמקד בקוד חסר רישות

הערה: כל קוד UD ניתן להעברה לקוד חסר רישות

<u>קוד Uniquely Decipherable: UD</u>

ניתן לפענוח בצורה יחידה, אם קוד ניתן לפענוח בכמה צורות, קוד זה לא מעניין אם כי לכל קלט יכולים להיות כמה פלטים.

אבחנה:

$$Prefix-free \Rightarrow UD$$

כלומר כל קוד חסר רישות הוא UD

לדוגמא:

עבור מילות הקוד
$$a=1,\quad b=01,\quad c=001,\quad d=0001,\quad e=00001$$
נקבל את הקוד הבא: $1|01|001|0001|00001$

$$UD \nRightarrow Prefix-free$$
$$Not\ Prefix-free \nRightarrow Not\ UD$$

דוגמא: בהינתן המחרוזת $abcde$

עבור מילות הקוד:
$$a=1,\quad b=10,\quad c=100,\quad d=1000,\quad e=10000$$
, קוד לא חסר רישות אבל UD: $1|10|100|1000|10000$

כדי להוכיח שקוד כלשהו הוא לא UD יש צורך לתת מחרוזת בינארית שיש לה שתי פירושים שונים

לדוגמא: עבור הקוד:
$$a=0,\quad b=101,\quad c=100,\quad d=111,\quad e=110,\quad f=1100$$
אזי הקוד
$$1100=1100="f"$$
$$1100=110+0="ea"$$

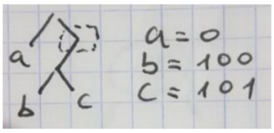
קורס דחיסת נתונים א – סמסטר א' 2020 מדעי המחשב

נכתב ע"י צבי מינץ

מרצה: פרופ' דנה שפירא

הערה: עבור דחיסה, קוד **אופטימלי** חייב להיות מיוצג ע״י עץ מלא (לכל צומת יש או 0 בנים או 2 בנים)

הערה: לא כל קוד חסר רישות הוא עץ מלא אבל קוד חסר רישות אופטימלי הוא עץ מלא, לדוגמא:



קוד שלם Complete Code

קוד עבורו כל מחרוזת אינסופית למחצה, ניתנת לפענוח בצורה ייחודית. קוד שלם יותר עץ מלא.

איך נדע שקוד הוא לא שלם? מספיק להראות מחרוזת **שאינה** ניתנת לפענוח עם הקוד הזה.

מחרוזת אינסופית למחצה: מחרוזת שהיא אינסופית בכיוון אחד, כלומר יש לנו התחלה ברורה ואח״כ יש המשך אינסופי

קוד מיידי Instantaneous code

המפענח יודע את הפענוח ברגע שמילת הקוד מסתיימת (משמאל לימין).

זה קורה בקוד חסר רישות (פרפיקסי).

כאשר יש לנו קוד רגעי, אנו יכולים לזהות מילת קוד ברגע שסיימנו לקרוא אותה. אנחנו יודעים שהסימן הבא שייך למילת הקוד הבאה.

משפט: לא כל קוד בעל פענוח יחיד הוא קוד מיידי.

עבור המילה 0111111111111111 עם מילות הקוד $\{0,01,11\}$ כי לקלט יש פענוח יחיד, אבל ניתן לדעת את מילת הקוד רק אחרי קריאת כל הקלט.

מבחן זיהוי יחודי Unique Decipherability Test

הגדרה: יהיו a,b שתי מחרוזות בינאריות כאשר $|a| = k$ ביטים ו- $|b| = n$ כאשר $k < n$

אם k הביטים הראשונים של a הינם זהים ל-k הביטים הראשונים של b אזי הם נקראים **prefix**

ושאר הביטים נקראים **dangling suffix**

לדוגמא: $a = 010,\ \ b = 01011 \Rightarrow \boldsymbol{dangling\ suffix} = 11$

אלגוריתם:

- Examine all pairs of codewords:
1. Construct a list of all codewords.
2. If there exist a codeword, a, which is a prefix of another codeword, b, add the dangling suffix to the list (if it is not there already), until:
   I. You get a dangling suffix that is an **original** codeword → the code is not UD
   II. There are no more unique dangling suffixes → the code is UD

אלגוריתם Sardinas–Patterson

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

```
i ← 1
S₁ ← C⁻¹C −{ε}
while true
      Sᵢ₊₁ ← C⁻¹Sᵢ ∪ Sᵢ⁻¹C
      i=i+1
      if ε ∈ Sᵢ or c ∈ Sᵢ for c in C
            print not UD and exit
      else if ∃ j<i such that Sᵢ=Sⱼ
            print UD and exit
```

17

דוגמת הרצה: (קוד UD)

יהיו מילות הקוד $\{0,01,11\}$

1. 0 היא רישא של 01, ולכן suffix = 1
   נעדכן את הרשימה - $\{1, 1\}$
2. 1 היא רישא של 11 ולכן נוסיף את dangling suffix = 1 לרשימה, אולם היא גם ככה ברשימה ולכן אין שינוי.
3. אין עוד מילות קוד שהם רישא של מילת קוד אחרת, ולכן אין יותר dangling suffixes ולכן הקוד הוא UD

קורס דחיסת נתונים א – סמסטר א' 2020 מדעי המחשב
נכתב ע"י צבי מינץ
מרצה: פרופ' דנה שפירא

הערה: אם היה מילת קוד 1 אז הקוד לא היה UD כי dangling suffix שווה למילת קוד אחרת.

דוגמא להרצה לקוד שהוא *אינו UD*:

- Codewords {0,01,10}
- 0 is a prefix of 01 → dangling suffix is 1
- List - {0,01,10,1}
- 1 is a prefix of 10 → dangling suffix is 0 - which is an original codeword!
- → the code is not UD

קודי יתרות מינימליים *Minimum Redundancy Codes*

בהינתן הסתברויות או התפלגויות מסויימות, אין עוד אפשרות של קידודים אחרים שיכולים להפחית את אורך מילת הקוד הממוצעת מעבר למה שנקבל בקוד בעל יתירות מינמלית.

<u>באופן פורמלי:</u>

יהי $E(C,P)$ אורך קוד ממוצע עבור C, אזי C הוא קוד בעל יתירות מינימלי עבור ההסתברות P אם $E(C,P) \le E(C',P)$ עבור כל קידוד C'

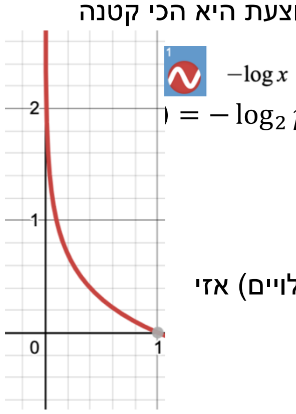
לדוגמא:

s_i	p_i	Code 3
a	0.67	0
b	0.11	100
c	0.07	101
d	0.06	110
e	0.05	1110
f	0.04	1111
Expected length		1.75

Can do even better with Arithmetic coding -
1.65 bits per symbol

<u>משפט הקידוד של שנון *Shannon*</u>

המטרה היא לבנות קוד בעל יתירות קוד מינימלית, משמע אורך מילת הקוד הממוצעת היא הכי קטנה שאפשר.



<u>אינפורמציה:</u> בהינתן סמל s_i והסתברות p_i אזי כמות האינפורמציה של s_i היא $) = -\log_2 p_i$
זה אומר כמות הביטים המינמלית כדי לייצג את s_i

תכונות של אינפורמציה:

אם $p_i = 1$ אזי $I(s_i) = 0$
אם $p_i = 0$ אזי $I(s_i) = \infty$

אם ישנו רצף של תווים $s_i s_j$ כך שההסתברויות שלהם הם $p_i \cdot p_j$ (התווים בלתי תלויים) אזי

$$I(s_i s_j) = I(s_i) + I(s_j)$$
$$-\log_2(p_i p_j) = -\log_2 p_i + (-\log_2 p_j)$$

ככל שההסתברות היא גדולה יותר, אז רמת האינפורמציה קטנה יותר

לדוגמא עבור הקוד הקודם נקבל כי:

Code 1: $p(s_1)=0.67$, $I(s_1)=0.58$, $p(s_6)=0.04$, $I(s_6)=4.64$

הבעיה כרגע היא איך ניתן להקצות 0.58 ביטים ל-s_1?
לכן שנון הגדיר:

$$H(P) = -\sum_{i=1}^{n} p_i \cdot \log_2 p_i$$

אשר זהו ממוצע משוקלל של האינפורמציה אשר מהווה **חסם תחתון**, כלומר עבור כל קידוד C נקבל כי
$H(P) \le E(C,P)$

זה נקרא ***Entropy* (אנטרופיה)**

נשים לב כי ה-Entropy של הדוגמא הינה 1.65 אשר מהווה חסם תחתון עבור כל קידוד אפשרי.

$$-0.67*\log_2 0.67-\ldots-0.04*\log_2 0.04=1.65$$

אי-שיוון קראפט *Kraft's Inequality*

אי-שיוון קראפט מתאר תנאי מספיק והכרחי לשיוך קבוצת מילים לצמתי עץ, כך שלא תשויך יותר ממילה אחת לאורך כל מסלול היוצא מהראש.

שאלה: כמה קצר יכול להיות קוד שהוא UD?
נניח שעבור כל תו s_i יש הסתברות $p_i = 2^{-k_i}$
אזי $I(S_i) = k_i$ אזי לקבוע כל מילת קוד להיות מחרוזת $|c_i| = k_i$ ביטים תגורר למילת הקוד הממוצעת (התוחלת) להיות החסם של שנון

משפט: יהי $C = [c_1, c_2, \ldots, c_n]$ להיות קוד עם n מילות קוד עם אורכים $|C| = [|c_1|, \ldots, |c_n|]$
אזי אם C הוא UD אזי

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

משפט: אם $K(C) = \sum_{i=1}^{n} 2^{-|c_i|} \le 1$ עבור קוד C כלשהו (לא בהכרח יחודי) אזי קיים קוד C' **אחר** כך ש:
1. $E(C,P) = E(C',P)$
2. $|C'| = |C|$
3. C' הוא קוד חסר רישות

במילים אחרות, קיים קוד רגעי (חסר רישות) אופטימלי (בפרט או ייחודי)

משפט: אם קוד C הוא יחודי אז בערך הקראפ קטן שווה מ-1 (זה תנאי הכרחי ליחודיות)
משפט: אם $K(C) = \sum_{i=1}^{n} 2^{-|c_i|} > 1$ עבור קוד C כלשהו אזי הוא **לא קוד חסר רישות** ובפרט לא יחודי.

e.g. there is no prefix code C with 5 codewords
that satisfy |C|=[2,2,2,2,2]

דוגמה: האם ניתן לבנות קוד חסר רישות בינארי ממילות קוד באורך 1,1,2?
פתרון: לא כי $K(C) = \frac{1}{2^1} + \frac{1}{2^1} + \frac{1}{2^2} = \frac{5}{4} > 1$ ולכן לפי אי שיוון קרפט לא ניתן לבנות קוד כזה

מטרה: עבור קבוצת הסתברויות נתונה $P = [p_1, p_2, \ldots p_n]$ נרצה לבנות מילות קוד $C = [c_1, c_2, \ldots, c_n]$ כך ש:
1. $K(C) \le 1$
2. $E(C,P)$ **מינימלי**

Fixed Length **–** קוד שבו כל מילת קוד היא באורך קבוע, כלומר כל מילות הקוד באותו האורך לדוגמא *Ascii*
Variable Length **–** קוד שבו המילות קוד הן באורך משתנה

הרצאה 2 – מודלים והוכחת משפט קראפט

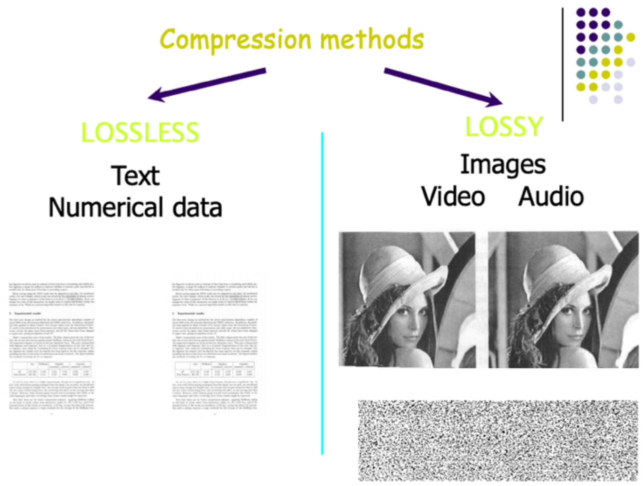
מטרות דחיסה:
1. לשמור מקום
2. להוריד את פעולות הקלט/פלט אשר גורמות לחיסכון בזמן
3. שיפור זמני התקשורת ע"י העברת קובץ קטן יותר
4. קריפטוגרפיה – להגן על מידע מגורם שלישי, להגן על המידע באמצעות הורדת **היתירות**
הערה: קודם כל יש לדחוס ואז להצפין, כי אם היינו עושים הפוך היינו מקבלים קובץ רנדומלי
5. נרצה להוריד מידע מיותר, כלומר להוריד את היתירות – *Redundancy*
הגדרה: יתירות הוא ביטוי כללי המתאר מצב או תכונה של כפילויות, תוספת מעבר לנדרש או הנורמלי.

קורס דחיסת נתונים א – סמסטר א' 2020 מדעי המחשב

נכתב ע"י צבי מינץ

מרצה: פרופ' דנה שפירא

סוגי דחיסות:



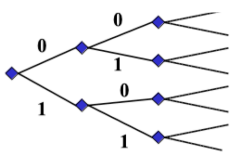
תזכורת (אי-שיוון קרפט):

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

$K(C)\le 1 \Leftrightarrow$ **קיים** קוד חסר רישות עם האורכים $|C|$

הוכחה: (נניח כי $l_i \le l_j$ עבור כל $i<j$)

<u>כיוון 1: נניח כי קיים קוד חסר רישות ונראה כי $K(C)\le 1$</u>



נסתכל על עץ D-ארי מלא (ראה שרטוט עבור D=2).

הקשתות מסומנות מ-0 עד D-1

כל צומת בעץ מייצג מילת קוד אפשרית.

לפי הגדרת קוד רגעי - מילת קוד אינה יכולה להיות על צומת שהוא צאצא של מילת קוד אחרת.

- יהי $l_{\max}$ אורך מילת קוד הגדול ביותר בקידוד רגעי כלשהו.
- ברמה $l_{\max}$ בעץ יש $D^{l_{\max}}$ צמתים.
- לכל מילת קוד באורך l_i יש $D^{l_{\max}-l_i}$ צאצאים ברמה $l_{\max}$
- כל קבוצות הצאצאים הנ"ל זרות (מתכונות עץ).
- סה"כ צאצאים של מילות קוד ברמה $l_{\max}$: $\sum_{i=1}^{m}D^{l_{\max}-l_i}$
- $\sum_{i=1}^{m}D^{l_{\max}-l_i}\le D^{l_{\max}} \Rightarrow \sum_{i=1}^{m}D^{-l_i}\le 1$

15

הערה: קוד רגעי הינו קוד מיידי כלומר קוד חסר רישות

<u>כיוון 2: נניח כי $K(C)\le 1$ ונוכיח כי קיים קוד חסר רישות</u>

ניתן לבנות קידוד רגעי עם האורכים הנ"ל ע"י מעבר על אותו עץ מלא:

Start from an empty code.

for $i=1$ to m do:

- Scan the tree left - most to the first node of depth l_i.
- Add the code - word corresponding to the node to the code.
- Delete the node and all its descendants.

בגלל המחיקה - מובטח לנו כי תנאי הרישא מתקיים.

נותר רק להראות כי הבניה בהכרח לא תיכשל. היא עלולה להיכשל רק אם עבור i כלשהו, לא קיים צומת ברמה l_i . נניח בשלילה כי זה קרה.

כל צומת שנבחר ברמה l_j $(j = 1,\dots,i-1)$ גורם למחיקת $D^{l_i-l_j}$
צמתים ברמה l_i . בסה"כ נמחקו D^{l_i} ברמה l_i .

$$\sum_{j=1}^{i-1} D^{l_i-l_j} = D^{l_i} \quad \Rightarrow \quad \sum_{j=1}^{i-1} D^{-l_j} = 1$$

$$\Rightarrow \quad \sum_{j=1}^{i} D^{-l_j} > 1$$

בסתירה לקיום אי-השוויון.

□

מודלים:

דחיסה סטטית:

נקבע פעם אחת לפני התחלת הקידוד, ולא משתנה במהלכו.

לדוגמא: מייצגים כל תו לפי קידוד *Ascii* ולכן אין צורך לתאר את המודל ולכן אין *Overhead*

נקבל כי $p_i = \frac{1}{256}$ $\forall i:$ ולכן האנטרופיה $H(C) = -\sum_{i=1}^{256} \frac{1}{256} \cdot \log_2 \frac{1}{256} = 8.0$

דחיסה סטטית למחצה:

בשיטה זו נסתכל על הקובץ כדי לראות כמה תווים שונים יש לנו ואז מניחים שההסתברויות בינהם היא אחידה, במקרה זה יש *prelude* שאומר למפענח אילו תווים יש בטקסט, ב*prelude* נשתמש ב- *Ascii* וגם צריך להגיד למפענח כמה תווים שונים מעבירים לו.

לדוגמא: עבור הקלט:

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!
```

נקבל כי $p_i = \frac{1}{25}$ כי יש 25 אותיות שונות ולכן

$$H(C) = -\sum_{i=1}^{25} \frac{1}{25} \cdot \log_2 \frac{1}{25} = 4.64$$

אולם, בגלל שהערוץ צריך לדעת את הקידוד יש צורך להעביר קודם כל את הקידוד כדי שהפענח יוכל לדעת לפענח ולכן יש Overhead

כל מילת קוד תצטרך 8 ביט (מעבירים בקידוד Ascii)

ולכן סה"כ נצטרך $8 \cdot 25$ ביטים של הסבר למפענח, ולכן סה"כ נקבל כי:

$$4.64 + \frac{8 \cdot 25 + 8}{128} = 4.64 + \frac{208}{128} = 6.27$$

כאשר 128 זה כמות האותיות הכוללת בהודעה

כאשר ה8 הביטים במונה זה בשביל לדעת שיש 25 תווים **שונים** $(\log_2(256) = 8)$ היות ויכול להיות שנקבל קובץ אחר עם יותר מ25 תווים, אבל המקסימום זה 256 תווים **שונים**), במילים אחרות זהו $|n|$ התווים השונים

מודל סטטי למחצה עם הסתברויות עצמאיות:

בנוי על סטטיסטיקות. כאשר $p_i = \frac{\text{כמה פעמיים } s_i \text{ הופעה}}{\text{גודל ההודעה}} = \frac{v_i}{m}$

לדוגמא: בבניית קוד נסתמך על נתונים סטטיסטיים המורים כי 'י' היא האות השימושית ביותר, אחריה- 'ה', וכו'...

מודל זה מורכב היות וצריך להעביר טבלה של הסתברויות למפענח באופן הבא:

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		

ולכן נקבל:

8 bits - alphabet size (why?)
25*8 - symbol descriptions
25*4 - symbol frequencies

ב-*prelude* נתאר:

- גודל הא"ב $|n| = 8bits$
- התווים עצמם ב-*Ascii* – $8 \cdot n\ bits$
- תיאור השכיחויות לכל תו - תיאור השכיחות $n \cdot$

קבלנו כי התוחלת הינה $6.63 = 4.22 + \frac{308}{128}$ שזה יותר **גרוע** ממודל סטטי למחצה

מודל סדר ראשון:

במודל זה מניחים תלוי בין התווים, אם אנו יודעים שתו מסויים מופיע ואחריו יש לנו הסתברויות גבוהה יותר לקודד תויים מסויימים, אנו מעילים את ההסתברויות של אותו תו ואז כמות האינפורמציה יורדת, ולכן גם האנטרופיה יורדת.

חישוב סדר התווים, כלומר לפי תו מסויים מה התו הבא שנראה

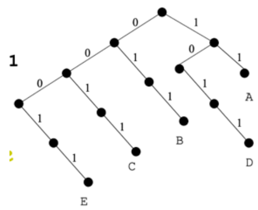
'i' is followed by 'n' with probability 5/8
'i' is followed by 'o' with probability 1/8
'i' is followed by 'r' with probability 2/8

סיכום משפטים והגדרות עד כה:

קוד חסר רישות

אף מילת קוד אינה רישא של מילת קוד אחרת

משפט: אם ניתן לייצג קוד בעזרת עץ שכל מילות הקוד בעלים אז הוא חסר רישות



קוד UD

קוד אשר ניתן לפענוח בצורה יחידה

$\boldsymbol{Prefix - free \Rightarrow UD}$
$\boldsymbol{UD \nRightarrow Prefix - free}$

משפט: $K(C) \leq 1$ אם"ם קיים קוד ייחודי (UD)

קורס דחיסת נתונים א – סמסטר א' 2020 מדעי המחשב
נכתב ע"י צבי מינץ
מרצה: פרופ' דנה שפירא

אורך מילת הקוד הממוצעת

$$E(C,P)=\sum_{i=1}^{n}p_i\cdot|c_i|$$

קוד מיידי Instantaneous

Instantaneous ⟷ Prefix code

המפענח יודע את הפענוח ברגע שמילת הקוד מסתיימת (משמאל לימין), קוד כזה נקרא קוד מיידי
משפט: לא כל קוד בעל פענוח יחיד הוא קוד מיידי.

משפט: אם בהינתן קידוד כלשהו, נהפוך אותו ונקבל קוד חסר רישות אז הוא **קוד שלם** כי הוא בפרט מיידי
לדוגמה הקוד הבא הוא קוד מיידי, בפרט קוד יחודי

$$C(x)=\begin{cases}0, & \text{if } x=1\\ 01, & \text{if } x=2\\ 11, & \text{if } x=3.\end{cases}$$

כי אם נהפוך את מילות הקוד נקבל את $C'(x)=\begin{cases}0, & \text{if } x=1\\ 10, & \text{if } x=2\\ 11, & \text{if } x=3\end{cases}$ אשר נותן קוד מיידי (חסר רישות) ולכן הוא בפרט קוד יחודי, בנוסף התירגום של כל $w_c=w_{c'}^R$ הוא גם כן יחודי ולכן הקוד המקורי הוא קוד יחודי

קוד שלם
קוד עבורו כל מחרוזת אינסופית למחצה, ניתנת לפענוח בצורה ייחודית.
איך נדע שקוד הוא לא שלם? מספיק להראות מחרוזת **שאינה** ניתנת לפענוח עם הקוד הזה.

מבחן זיהוי קוד על פענוח יחיד (UD)
יש אלגוריתם אשר בודק אם בהינתן קלט קידוד האם הוא UD:
אלגוריתם Sardinas–Patterson

אי שיוון קרפט (בדיקה אם תכונת '**חסרת הרישות**' מתקיימת)

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

משפט: $K(C)\le 1 \Leftrightarrow$ **קיים** קוד חסר רישות עם האורכים |C|

אנטרופיה
זהו ממוצע משוקלל של האינפורמציה אשר מהווה **חסם תחתון**, כלומר עבור כל קידוד C $H(P)\le E(C,P)$

$$H(P)=-\sum_{i=1}^{n}p_i\cdot\log_2 p_i$$

קוד בעל יתירות מינימלי
משפט הקידוד שנון:

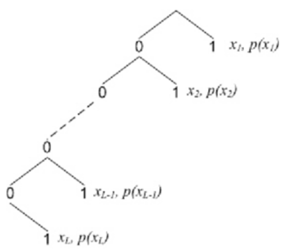
$$H(P)=-\sum_{i=1}^{n}p_i\cdot\log_2 p_i$$
$$\forall C: H(P)\le E(C,P)$$

קוד בעל יתירות, קוד מיותר (Redundant Code)
קוד מיותר תמיד יכול להיות טוב יותר ע"י הורדת אורך הקידוד למילת קוד כלשהי
משפט: אם $K(C)<1$ אזי הוא **קוד מיותר** (קוד עם יתירות) **(קיים קודקוד עם בן יחיד)**
אם $K(C)=1$ והקוד חסר רישות אז הוא **קוד שלם** **(אפשר לייצג בעץ מלא)**

<u>ויזואלציה מבוססת עץ</u>
אם יוצרים קוד פרפיקסי, אז ניתן להסתכל על העץ ומילות הקוד יהיו בעלים.
יהיה מסלול יחיד מהשורש עד לעלים וזה תיהיה מילת הקוד, הקידוד והפענוח יהיה באמצעות מעבר על העץ, המעבר על העץ הוא דיי איטי.
עץ אופטמלי – עץ השייך לקוד אופטימליd

<u>הרצאה 3:</u>
תזכורת: קוד Unary
קוד אונרי הוא קוד קבוע מראש, עבור א"ב סופי או אינסופי מילות הקוד באינדקס i תיהיה $x_i = 1^{i-1}0$
כאשר הסיבית 0 בסוף משתמשת כהפרדה בין מילות הקוד
כלומר:
מילות הקוד ה-1 – $x_1 =' 0'$
מילות הקוד ה-2 – $x_2 =' 10'$
וכך הלאה
אם ידוע לנו שא"ב סופי נוכל להוריד את ה0 במילת הקוד האחרונה ואז נקבל עץ מלא – קוד שלם
זה לא יפריע לנו משום שאם יודעים שישנם n מילות קוד אזי כאשר נראה 1^{n-1} נעבור לתו הבא, אם הוא 0 אזי מילת הקוד הוא x_{n-1} אם 1 אז זוהי מילת הקוד x_n

שאלה: אם הא"ב סופי ו-n (כמות התווים השונים) ידוע למפענח, האם נוכל לחסוך באורך הקוד האונרי?



פתרון: כן, הקוד שנקבל הוא קוד עם יתירות היות ונוכל לקצץ את הקודקוד האחרון ולהוריד ביט

שאלה: מתי קוד אונרי הוא קוד ללא יתירות עבור הסתברות אינסופית (אינסוף תווים)? עבור הסתברות סופית?
פתרון:
עבור אינסופי:
נרצה ליצור קוד שבו ההסתברויות של התווים הן חזקות של ½ ואז כמות האינפורמציה של כל תו תיהיה שווה בדיוק לאורך של מילת הקוד.
ולכן אם נרצה לקודד תו עם מילת הקוד 01 אז נרצה שההסתברות שלו תיהיה רבע $\left(\frac{1}{2}\right)^2$ ולכן:
שאיפה היא ככמות האינפורמציה (אבל זה לא תמיד אפשרי כי זה שברי ביטים)

$$p_1 = \frac{1}{2}, \quad p_2 = \frac{1}{4}, \quad I(S_i) = -\log p_i = \log\left(\frac{1}{p_i}\right)$$

כדי ש-$I(p_i) = i$
ולכן נרצה שההסתברויות לתווים יהיו חזקות של 2 כי אז

$$E(C,P) = \sum_{i=1}^{n} p_i \cdot |c_i| = H(P) = -\sum_{i=1}^{n} p_i \cdot \log_2 p_i$$

עבור סופי:
לקצץ את הביט האחרון כדי ש- $I(p_i) = i$

<u>קוד בינארי</u>
קוד בינארי "פשוט" – כל תו מותאם למילת קוד בגודל $\lceil \log_2 n \rceil$, כלומר עושים קוד בעל אורך קבוע, כמו ב-Ascii. נשים לב שאם יש לנו $n = 2^k$ (חזקה של 2) אז נקבל שאורך מילת הקוד הוא k

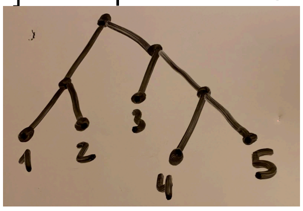
אוניברסיטת אריאל בשומרון

קורס דחיסת נתונים א – סמסטר א' 2020 מדעי המחשב
נכתב ע"י צבי מינץ
מרצה: פרופ' דנה שפירא

קוד בינארי "מינמלי" – לא מחייבים קוד בעל אורך קבוע, נעשה שחלק ממילות הקוד יהיו באורך $[\log_2 n]$ והשאר יהיו באורך $[\log_2 n] - 1$, כלומר נאפשר מקסימום הפרש של סיבית אחת בין מילות הקוד הקצרות למילות הקוד הארוכות.

עבור א"ב עם n תווים, אזי קוד בינארי מינמלי מכיל $2^{[\log_2 n]} - n$ מילות קוד שהם באורך $[\log_2 n]$ ושאר ה- $2n - 2^{[\log_2 n]}$ הם באורך $[\log_2 n]$
אבחנה: . נשים לב שאם יש לנו $n = 2^k$ (חזקה של 2) אז נקבל שאורך מילת הקוד הוא k

לדוגמה, עבור $n = 5$ וא"ב $S = [1,2,3,4,5]$ אזי מילות הקוד הינם $C = [00,01,10,110,111]$



שאלה: נניח כי n הוא חזקה של 2, מתי קוד בינארי מינימלי הוא ללא יתירות?
פתרון: $\forall i: p_i = \left(\frac{1}{2}\right)^{|s_i|}$ ולכן האנטרופיה $H(P) = E(C,P)$

__**Elias Codes**__
זה מעין מיזוג של קוד הבינארי והקוד האונרי
הקידוד עבור x הוא סדר גודל $O(logx)$ ביטים
נראה שני קודים - C_γ ו- C_δ

- **C_γ: (גמה C)**
  - __**החלק הראשון:**__
    - כותבים בקוד אונרי את מספר הסיביות, ביצוג הבינארי של x יקח לנו $1 + [\log_2 x]$
  - __**החלק השני:**__
    - קוד בינארי עבור x עם טווח אשר נקבע ע"י הקוד האונרי
    - כלומר, כותבים את היצוג הבינארי ללא ה-1 המוביל
    - אשר לוקח $[log_2 x]$ ביטים
- ולכן נקבל שצריך $1 + [log_2 x]$ עבור החלק הראשון ו- $2[log_2 x]$ עבור הקוד בינארי סה"כ:
  $1 + 2[\log_2 x]$

לדוגמה:

x	דרך חישוב	תוצאה
1	יצוג בינארי: 1 $\underbrace{0}_{\text{אונרי}}$ $\underbrace{-}_{\text{בינארי}}$	0
2	יצוג בינארי: 10 $\underbrace{10}_{\text{אונרי}}$ $\underbrace{0}_{\text{בינארי}}$	10 0
3	יצוג בינארי: 11 $\underbrace{10}_{\text{אונרי}}$ $\underbrace{1}_{\text{בינארי}}$	10 1
4	יצוג בינארי: 100 $\underbrace{100}_{\text{אונרי}}$ $\underbrace{00}_{\text{בינארי}}$	100 00
5	יצוג בינארי:101 $\underbrace{110}_{\text{אונרי}}$ $\underbrace{01}_{\text{בינארי}}$	110 01
6	יצוג בינארי: 110 $\underbrace{110}_{\text{אונרי}}$ $\underbrace{10}_{\text{בינארי}}$	110 10

7	יצוג בינארי: 111 $\underbrace{110}_{\text{אונרי}}\ \underbrace{11}_{\text{בינארי}}$	110 11
8	יצוג בינארי: 1000 $\underbrace{1110}_{\text{אונרי}}\ \underbrace{000}_{\text{בינארי}}$	1110 000

- **C_δ: (C דלתא)**
  - **החלק הראשון:**
    - בחלק הראשון כותבים את מספר הביטים ביצוג הבינארי של x אבל הפעם לא בונארי אלא ב- C_γ
    - קוד C_γ עבור **כמות הביטים** ב-x אשר זהו $1+2\lfloor\log_2\log_2 2x\rfloor$
  - **החלק השני:**
    - קוד בינארי עבור x עם טווח אשר נקבע ע"י הקוד החלק C_γ כלומר ללא 1 המוביל
- סה"כ $1+2\lfloor\log_2\log_2 2x\rfloor+\lfloor\log_2 x\rfloor$ ביטים
- **לדוגמה:**

x	דרך חישוב	תוצאה
1	יצוג בינארי: 1 $\underbrace{0}_{C_\gamma}\ \underbrace{-}_{\text{בינארי}}$	0
2	יצוג בינארי: 10 $\underbrace{100}_{C_\gamma}\ \underbrace{0}_{\text{בינארי}}$	100 0
3	יצוג בינארי: 11 $\underbrace{100}_{C_\gamma}\ \underbrace{1}_{\text{בינארי}}$	100 1
4	יצוג בינארי: 100 $\underbrace{101}_{C_\gamma}\ \underbrace{00}_{\text{בינארי}}$	101 00
5	יצוג בינארי:101 $\underbrace{101}_{C_\gamma}\ \underbrace{01}_{\text{בינארי}}$	101 01
6	יצוג בינארי: 110 $\underbrace{101}_{C_\gamma}\ \underbrace{10}_{\text{בינארי}}$	101 10
7	יצוג בינארי: 111 $\underbrace{101}_{C_\gamma}\ \underbrace{11}_{\text{בינארי}}$	101 11
8	יצוג בינארי: 1000 $\underbrace{(C_\gamma\ of\ 4\ since\ its\ \ 4\ bit)\ 11000}_{C_\gamma}\ \underbrace{000}_{\text{בינארי}}$	11000 000

השוואה בין הקודים:

x	Unary	C_γ	C_δ
1	0	0	0
2	10	10 0	100 0
3	110	10 1	100 1
4	1110	100 00	101 00
5	11110	110 01	101 01
6	111110	110 10	101 10

7	1111110	110 11	101 11
8	11111110	**1110 000**	**11000 000**

נקבל כי:

- C_γ longer than Unary code only for x=2, x=4
- C_δ longer than C_γ only for x=2,3,8…15
- For large values Alias Codes are exponentially better than Unary codes.