מ.מ.למדמ"ח ~ עמית וינשטיין ~ קוד ברגר וחישוב נומרי

שחר פרץ

25 ליוני 2024

REMINDERS . . . .

1.1 Index Code

$$
\begin{aligned}
& & EC(x) = \bigoplus_{x_i=1} i & \\
IC: & \quad & x \circ EC(x) & \quad d=2 \\
IC_2: & \quad & x \circ EC(x) \circ EC(x) & \quad d=3 \\
IC_3: & \quad & x \circ EC(x) \circ EC(x) \circ par(x) & \quad d=4
\end{aligned}
$$

1.2 Hamming Code (743)

The messeage: $x_3x_5x_6x_7$ (2^4 options)

We adds:

$$
\begin{aligned}
x_1 &= x_3 \oplus x_5 \oplus x_7 \\
x_2 &= x_3 \oplus x_6 \oplus x_7 \\
x_4 &= x_5 \oplus x_6 \oplus x_7
\end{aligned}
$$

And in general, hamming code will acts as $(2^t-1, 2^t-t-1, 3)$

ניתן להסתכל על 2^4 מילות הקוד ולוודא שאכן מתקיים $d=3$. לחילופין, אפשר לפלג ל־3 מקרים עבור השתנות של כל מספר תווים, ולהראות שזה ישנה כמות נתונה של ביטי זוגיות. לכן, ניתן לתקן שששששגיאה בודדת. איך? נניח רצו לשלוח את ההודעה $x_1x_2x_3x_4x_5x_6x_7$. בפועל, קיבלנו את $y_1y_2y_3y_4y_5y_6y_7$. נניח שקיימת לכל היותר שגיאה בודדת. נחשב:

$$
\begin{aligned}
y_1' &= y_3 \oplus y_5 \oplus y_7 \\
y_2' &= y_3 \oplus y_6 \oplus y_7 \\
y_4' &= y_5 \oplus y_6 \oplus y_7
\end{aligned}
$$

נבדוק באילו מבין y_1, y_2, y_4 שונים מ־y_1', y_2', y_4'. נחשב את האינדקס:

$$i = 4 \cdot [y_4 \oplus y_4'] + 2 \cdot [y_2 \oplus y_2'] + 1 \cdot [y_1 \oplus y_1']$$

כאשר הכפל במספרים הוא כדי לייצד להתחשב בהיות המספר בינארי, וקיסור ייבדוק האם הם שונים. נתקן את ההודעה: נהפוך את y_i עבור ה־i שחישבנו, ונחזיר את הערכים באינדקסים $3, 5, 6, 7$.

טענה: עבור $i \neq j$, $x_j = y_j$, ובנוסף, $x_i \neq y_i$ (i משתנה קשור, j חופשי. לא מנוסח בבהירות).

אינטואיציה שאני כתבתי: אם הטעות בתיקון השגיאות, אז ישתנה איבר יחיד מבין y_1, y_2, y_3 ולכן הקיסור כחלק מציאת האינדקס יתחלף במיקום יחיד, ולא נשנה את y_3, y_5, y_6, y_7 (תבדקו ידנית). כחלק מהבנייה של i, נמצא יותר ביטי זוגיות יתהפכו, בנינו את הקוד בצורה כזו שהאינדקס יתאים (לדוגמא, טעות ב־$y-7$ תגרור שלוש שגיאות, ולכן נגיע לאינדקס האחרון; כן טעות באחרים יגררו שתי שגיאות שיובילו למיקום המדוייק גם כן).

אינטואיציה שהמורה כתב: כאשר ביו מתהפך, הוא משפיע על y_1', y_2', y_4' בהדיוק לפי היבטים הדוקים בייצוג הבינאי של האינדקס שלו. לכן, אך יש עוד שיבוש יחיד, נתקן בדיוק אותו.

BERGER CODE

נגדיר $k = 2^\ell - 1$. נגדיר:

$$\text{Berger}(x) = x \circ \underbrace{\text{bin}\left(\text{כמות ביטי האפסים ב־}x\right)}_{EC(x)}$$

דוגמה: $\ell = 3,\ x = 110001, EC(x) = 011$ (כי יש 3 ביטי אפסים ב־x)

אורך מילת קוד? $2^\ell - 1 + \ell$

מרחק מינימלי? $d = 2$.

לא מאפשר לתקן טעויות. אבל יש לו תכונה מעניינת. נניח, וידוע שרק אפסים יהפכו לאחדים ($1 \mapsto 0$). אינטואיציה: ה־EC יוכל רק לגדול, וכמות האפסים ב־x תוכל רק לקטון, ולכן, תחת ההנחה שהשינוי הוא $1 \mapsto 0$ בלבד, נוכל לזהות כל מספר של טעויות מהסוג הזה. נפרמל.

טענה: נוכל לזהות כל מהפס ר של טעויות חד־צצדיות שבהן 0 הופך ל־1 (וזה עובד גם הפוך).

הוכחה. כל טעות בחלק של x מקטינה את מס' ה־0 ב־x שאמורים להיות שווים ב־$EC(x)$. מצד שני, כל טעות כזו ב־$EC(x)$ מגדילה את מס' האפסים שאנחנו מצפים לראות ב־x. לכן, לא משנה כמה טעויות כאלה תהינה, נראה אי־התאמה בין x המשודר ו־$EC(x)$ הישודר. ■

הבחנה: אותה ההוכחה תעבוד גם עבור שגיאות בכיוון ההפוך.

NUMERIC CALCULATIONS

מיועד לביצוע חישובים בעזרת המחשב, בין עם מתמטיים ובין אם לאו.

3.1 מציאת שורש

נתונה לנו פונקציה $f\colon \mathbb{R} \to \mathbb{R}$ (עד לכדי דיוק של float). רוצים למצוא את שורש של הפונ', כלומר ערך a כך ש־$f(a) = 0$, או לפחות $|f(a)| < \varepsilon$ עבור ε קטן שהוגדר מראש. בעיות:

1. בעיה ראשונה: האם יש שורש?
2. בעיה שנייה: האם הפונקציה רציפה?

דוגמה:

```
lambda x: 0 if x==0.123456 else 1
```

קשה למצוא שורש כי זה די אקראי, לפונקציה כזו.

הנחות מקלות: נניח שהפונקציה רציפה ובהכרח נתונים לנו u, l כך ש־$f(u) \geq 0, f(l) \leq 0$. תחת ההנחה הזו, ממשפט ערך הביניים, בין שתי הנקודות הללו יהיה שורש (כלומר קיימת נק' $L < a < u$ כך ש־$f(a) = 0$, בה"כ $L < u$).

אלגו': נביע מעין חיפוש בינאי המרחב "רציף". נחשב בכל צעד את הערך $x = \frac{L+u}{2}$ ו־$f(x)$. נפלג למקרים.

- אם $|f(x)| < \varepsilon$ אז סיימנו ונחזיר את x.
- אחרת:
  - אם $f(x) > 0$ נחליף בין u ל־x.
  - אם $f(x) < 0$ נחליף בין L ל־x.

פונקציות שמקבלות פונקציות אחרות כפרמטר, או מחזירות פונקציות בערך ההחזרה – נקראות high-order functions. בפייתון זה מאוד טבעי. חתימה:

```
def find_root(f: Callable, L: float, u: float, epsi: float) -> float: ...
```

כדי לחשב, לדוגמה, את $\sqrt{2}$, נוכל להעביר לפונקציה `f = lambda x: x * x - 2` עם הקצוות 1 ו־2 שכן $1 < \sqrt{2} < 2$.

3.2 חישוב π

נתבונן במעגל היחידה, וברביוע החוסם אותו. ברביע הראשון, היחס בין שטח הריבוע ($1^2 = 1$) לבין רבע המעגל ($\pi \cdot 1 \cdot 0.25 = \frac{\pi}{4}$) וסה"כ נחלק ונמצא את יחס השטחים.

הרעיון: נגריל ערכים ברביעו, ונבדוק כמה מהם נמצאים בתוך רבע העיגול – בהינתן נק' (x, y) נבדוק האם $x^2 + y^2 = 1$. נדגום n נקודות ונחשב כמה מתוכן נופלות בתוך רבע העיגול. נניח k נפלו ברבע העיגול, נחזיר $\frac{4k}{n}$. ההתפלגות של הערכים הנכונים, תהיה התפלגות בינומית. ניסויים מהסוג הזה נקראים Monte Carlo.

3.3 נגזרת ואינטגרל נומריים

3.3.1 נגזרת

הגדרת נגזרת:

$$f'(x) = \lim_{x \to 0} \frac{f(x+h) - f(x)}{h}$$

נרצה פונקציה בשם `diff`שמקבלת את f כקלט (ונניח גם את h) ומחזירה פונקציה חדשה f'. דוגמה:

```
def diff(f: Callable, h:float=.001) -> Callable:
    return lambda x: (f(x + h) - f(x)) / h
```

במקום לקחת h קבוע, אפשר לנסות לזהות מתי הנגזרת משתגעת ובתהאם לכך לבחור h יותר קטן.

3.3.2 אינטגרל

הגדרה של אינטגרל (מסויים): (ציור של שטח מתחת לפונקציה). סוכמים את שטח הפונ' מעל ציר ה־x פחות השטח שמתחת לציר במיקום במקטע.

ניתן לחלק את השטח למקטעים קטנים, ולסכום מלבנים ברוחב הקטן לפי ערך הפונקציה. עבור הקטע a, b מחשבים את הסכום:

$$h \cdot f(h) + h \cdot f(a+h) + \cdots + h \cdot f(a+ih) + \cdots + h \cdot f\left(a + \left\lfloor \frac{b-a}{h} \right\rfloor h\right)$$

כלומר את:

$$h \cdot \sum_{i=0}^{\left\lfloor \frac{b-a}{h} \right\rfloor} f(a+ih)$$

קוד:

```
def integral(f: Callable, h: float) -> Callable:
    return lambda a, b: h * sum(f(a + i * h) for i in range(1 + int((a - b) // h)))
```