

סיבוכיות

© ארזים

16 במאי 2017

1 אי דטרמיניזם ומחלקות משלימות

1.1 $NP \cap co-NP$

נרצה לומר משהו על איזה שפות נמצאות בתוך $NP \cap co-NP$. בבירור למשל $P \subseteq NP \cap co-NP$.

ניתן שפה לדוגמה במחלקה הזו: השפה PRIMES, שהיא אוסף כל המחרוזות הבינאריות המייצגות מספרים ראשוניים. קל לראות $PRIMES \in co-NP$ - העד יהיה הפירוק של המספר. יותר מסובך להראות $PRIMES \in NP$ אבל גם זה נכון. במשך הרבה זמן זה כל מה שידעו לומר, אבל לפני 15 שנה בערך הוכח $PRIMES \in P$.

שפה נוספת: Integer Factorization - כל הזוגות $\langle N, M \rangle$ בייצוג בינארי כך שיש גורם $1 \neq t \leq M$ ראשוני המקיים $t | N$. ברור ששפה זו היא בתוך NP - העד יהיה הגורם. לא קשה להראות גם שהיא בתוך $co-NP$ - העד יהיה אוסף כל הגורמים של N (כולל ריבוי). בעזרת PRIMES מוודאים את הראשוניות של כל הגורמים הללו, ומוודאים חלוקה.

שאלה ראינו $P = coP$. למה אותו טיעון לא יעבוד כדי להראות $NP = co-NP$? אם קלט בשפה שבתוך NP , ייתכן שיש מסלול דוחה בנוסף למסלול מקבל וזה לא ישתנה אם נחליף *reject, accept*.

1.2 $NL = co-NL$

משפט 1.1 (אימרמן-שלפצני - Immerman-Szelepesenyi) $NL = co-NL$, או במילים אחרות $STCON \in NL$.

הוכחה: נתאר ראשית את הרעיון. ההוכחה הולכת להראות לנו קיום עד לאי קשירות של גרף. נגדיר

$$C_k = \{v \mid \text{dist}(s, v) \leq k\}$$
$$C_0 = \{s\}, |C_0| = 1$$

נצטרך לדעת לאמר משהו על $|C_k|$ בהנתן מידע על $|C_{k-1}|$. נראה כיצד לוודא $v \notin C_k$ כשידוע $|C_k|$, ואפילו איך לוודא $v \notin C_k$ כשידוע $|C_{k-1}|$. ראשית, נראה עד לוודא $v \notin C_k$ בהינתן $|C_k|$. כלומר, ידוע לנו $|C_k|$ ונתון לנו קודקוד v , ואנחנו נבדוק האם $v \notin C_k$. העד שלנו יהיה רשימה (מסודרת מהקטן לגדול) של הקודקודים מתוך C_k , יחד עם המסלולים מהקודקוד s אליהם (הוכחה לכך שהם בתוך C_k).

נעבור עליהם אחד אחד, נוודא שהעד תקין, שיש מספיק קודקודים, ונבדוק האם v אחד מהם.

כעת ניתן עד לוודא $v \notin C_k$ בהינתן $|C_{k-1}|$. העד שלנו יהיה, בדומה לקודם, רשימה ממויינת של הקודקודים מתוך C_{k-1} ועדים שלהם, ולכל קודקוד כזה, נוודא שאין קשת ממנו אל v .

אם כן, בהינתן $|C_{k-1}|$ ניתן, לכל קודקוד $v \in V$, לספק עד להאם הוא בתוך C_k או לא. בעצם, זה נותן לנו הוכחה של $|C_k|$ בהינתן $|C_{k-1}|$. העד, אם כך, יכיל הוכחות לגדלים של C_k , בהינתן הגודל של C_{k-1} , עבור $1 \leq k \leq n$. בסוף הוא יכיל עד שמראה $t \notin C_n$. לכן סיימנו. ■

2 אלגוריתמים הסתברותיים

דוגמא בהנתן שלוש מטריצות $n \times n$, A, B, C , רוצים לוודא שמתקיים $A \cdot B = C$. דרך ראשונה - לחשב את $A \cdot B$ ולבדוק שוויון עם C . זמן ריצה - הטוב ביותר שידוע הוא n^ω כאשר $\omega = 2.3\dots$ נראה דרך אחרת.

הרעיון נניח שאנחנו עובדים מעל שדה סופי. אם $AB - C \neq 0$, אזי $\text{rk}(AB - C) \geq 1$, ולכן $\dim \ker(AB - C) \leq n - 1$. בשדה בגודל 2, למשל, גודל של תת מרחב ממימד r הוא 2^r . לכן, אם נבחר V באקראי, הסיכוי ליפול בגרעין הוא

$$\mathbb{P}(v \in \ker(AB - C)) \leq \frac{2^{n-1}}{2^n} = \frac{1}{2}$$

זמן הריצה הוא זמן החישוב של Cv , כלומר n^2 , וזמן החישוב של $ABv = A(Bv)$ כלומר עוד n^2 . בסך הכל - n^2 .

נכונות - אם $AB = C$ תמיד נקבל. אם $AB \neq C$ ההסתברות שנגריל V בגרעין של $AB - C$ היא לכל היותר $\frac{1}{2}$. אם נחזור על האלגוריתם 7 פעמים, ונקבל רק אם קיבלנו בכל אחת מהן, אז ההסתברות לטעות תרד להיות לכל היותר 0.01. היינו יכולים להגדיל את מספר האיטרציות, וכך לכל ε יש מספר איטרציות שמבטיח טעות שהיא לכל היותר בהסתברות ε . זמן הריצה יגדל רק בפקטור של $\log\left(\frac{1}{\varepsilon}\right)$. ראינו כאן מצב שבו בעזרת אקראיות הצלחנו לשפר מעט זמן ריצה. נראה כעת בעיה שבלי אקראיות אין לה פתרון יעיל בכלל.

בעיה מציאת מספר ראשוני בין n ביטים שגודל מאשר 2^{n-1} (עובדה - בין m לבין $2m$ יש מספר ראשוני).

אלגוריתם ננחש מספר בטווח הזה, ונבדוק האם הוא ראשוני (בזמן פולינומיאלי). אם הוא לא, נחזור על התהליך.

ברור שלא נטעה, אבל לא ברור מה ניתן לומר על זמן הריצה.

עובדה מספר הראשוניים בין m לבין $2m$ הוא $\theta\left(\frac{m}{\log m}\right)$. אצלנו זה יאמר שההסתברות לנחש מספר ראשוני היא בערך $\frac{c}{n}$, כאשר c קבוע. ההסתברות שנוץ t צעדים עד שנצליח היא

$$\left(1 - \frac{c}{n}\right)^{t-1} \cdot \frac{c}{n}$$

לכן תוחלת זמן הריצה היא

$$\sum_{t=1}^{\infty} t \cdot \left(1 - \frac{c}{n}\right)^{t-1} \frac{c}{n} = \frac{n}{c} = O(n)$$

מכאן, ומאי שוויון מרקוב, נוכל להקטין את ההסתברות לטעות כמה שרצה במספר קבוע של איטרציות. בעיה פתוחה ומעניינת היא למצוא אלגוריתם שבהינתן n מחזיר מספר ראשוני $2^{n-1} < p < 2^n$.

אלגוריתמים מהסוג של זה שראינו לכפל מטריצות (זמן ריצה פולינומי וטעות חסומה) נקראים אלגוריתמי מונטה קרלו, ואם הטעות חד צדדית (כלומר למשל אם לא בשפה בטוח צודקים), אז הם נקראים אלגוריתמי מונטה קרלו חד צדדיים. הסוג השני (אף פעם לא טועים, אבל בתוחלת זמן הריצה פולינומיאלי) נקרא אלגוריתמי לאס וגאס (או לאס וגאס חד צדדי, כמו קודם).

2.1 חישובים אלגבריים

בהרבה בעיות טבעיות, למשל חישוב דטרמיננטה, הפלט הוא פולינום בקלט. מודל החישוב הטבעי לחישוב פולינומים נקרא מעגל אלגברי. במעגל אלגברי יש שערי $+$, \times .

שאלה בהינתן שני מעגלים כאלו, רוצים להכריע האם הם מחשבים את אותו הפולינום.

אלגוריתם אם ידוע חסם עליון d על דרגת הפולינומים, נבחר נקודה $\bar{x} \in \{1, \dots, 2d\}^n$ באקראי ונבדוק האם הפולינומים מסכימים בנקודה.

למה 2.1 אם $f \neq 0, \deg(f) \leq d$ אזי

$$\mathbb{P}_{\bar{x} \in S^n} (f(\bar{x}) = 0) \leq \frac{d}{|S|}$$

בעיה פתוחה (וקשה): למצוא אלגוריתם דטרמיניסטי יעיל. הבעיה הזו נקראת Polynomial Identity Testing.

הגדרה 2.2 המחלקה $BPP(\alpha, \beta)$ מכילה את כל השפות $A \subseteq \{0, 1\}^*$ עבורן יש פולינום $r: \mathbb{N} \rightarrow \mathbb{N}$ ומכונת טיורינג $M(x, y)$ כך שמתקיים:

1. $|y| \leq r(|x|)$.

2. על כל קלט x, y , $M(x, y)$ רצה בזמן לכל היותר $r(|x|)$.

3. אם $x \in A$ אזי

$$\mathbb{P}_{y \in \{0,1\}^{r(|x|)}} (M(x, y) = 1) \geq 1 - \beta$$

ואם $x \notin A$ אזי

$$\mathbb{P}_y (M(x, y) = 1) \leq \alpha$$

מגדירים בנוסף:

$$\begin{aligned} \text{BPP} &= \text{BPP} \left(\frac{1}{3}, \frac{1}{3} \right) \\ \text{RP} &= \text{BPP} \left(0, \frac{1}{2} \right) \\ \text{coRP} &= \text{BPP} \left(\frac{1}{2}, 0 \right) \end{aligned}$$

ננסה לסווג את הבעיות שראינו. עבור בעיית $AB = C$, ראינו שאנחנו לא טועים אף פעם, כלומר $\beta = 0$, אבל ראינו שאנחנו צודקים בהסתברות לפחות $\frac{1}{2}$. לכן

$$"AB = C" \in \text{BPP} \left(\frac{1}{2}, 0 \right) = \text{coRP}$$

ברור (ותיכף נוכיח) למשל שמתקיים

$$\begin{aligned} \text{RP} &\subseteq \text{BPP} \\ \text{RP} &\subseteq \text{NP} \end{aligned}$$

טענה 2.3 מתקיים

$$\begin{aligned} \text{RP} &= \text{BPP} \left(0, \frac{1}{2^n} \right) \\ \text{BPP} &= \text{BPP} \left(\frac{1}{2^{n^c}}, \frac{1}{2^{n^c}} \right) \end{aligned}$$

הוכחה: לגבי RP - פשוט נחזור על האלגוריתם n פעמים, ונקבל אם קיבלנו באחת הריצות. לגבי BPP - נריץ את האלגוריתם n פעמים, ונקבל אם רוב הריצות קיבלנו. לפי אי שוויון צ'רנוף, שראינו בשיעורי הבית, אם X_1, \dots, X_n משתנים מקריים בלתי תלויים ושווי התפלגות המקיים

$$X_i \sim \begin{cases} 0 & 1-p \\ 1 & p \end{cases}$$

אזי מתקיים

$$\mathbb{P} \left(\sum_{i=1}^n X_i > (1 + \delta) pn \right) \leq \exp(-\delta^2 pn)$$

במקרה שלנו נקבל שההסתברות שיותר מחצי מהפעמים נקבל תשובה לא נכונה היא $\exp(-n)$ לכל היותר. ■