אלגוריתמים הסתברותיים

MatrixID

האם $AB \neq C$?

(A,B,C מטריצות מגודל $n \times n$ מעל $\{0,1\}$.)

<u>פתרון טריוויאלי</u>: מכפילים ובודקים – $O\left(n^3\right)$ [[זכור לי מהפעם הקודמת שעשיתי קורס

באלגוריתמים (לפני 4 שנים) שיש איזשהו אלגוריתם שעושה את זה ב- $O\left(n^{2\frac{7}{8}}\right)$ או משהו כזה...]]

<u>טעות חד-צדדית</u>:

- עבור תשובה א', ההסתברות לטעות היא 0.
- עבור תשובה ב', ייתכן שנעשתה טעות.

נדרש אלגוריתם הסתברותי בעל טעות חד-צדדית בהסתברות $\frac{1}{2}$ לכל היותר.

נגדיר $D = AB - C$.

אזי $D = 0$ אם"ם $AB = C$.

לכל $\vec{v}$, אם $D = 0$ אזי $0 = D\vec{v} = AB\vec{v} - C\vec{v}$.

[ואם $D = 0$ אז לכל $\vec{v}$ נקבל $D\vec{v} = 0$.]

למה זה עוזר לנו?

חישוב של מכפלה מהצורה $X \cdot \vec{v}$ עולה $O\left(n^2\right)$ (n פעמים כפל של שורה בעמודה, שמבצע n הכפלות וסכימה). אז נוכל לחשב את $D\vec{v}$ במקום ע"י חישוב D (אשר לבד עולה n^3 כי יש לבצע את המכפלה AB), ע"י חישוב:

$$D\vec{v} = \overbrace{A\underbrace{\left(B\vec{v}\right)}_{n \times 1}}^{O\left(n^2\right)} - \overbrace{C\vec{v}}^{O\left(n^2\right)}$$

כלומר, חישוב D עולה $O\left(n^3\right)$, אך חישוב $D\vec{v}$ עולה $O\left(n^2\right)$.

<u>הבעיה</u>: קיימים וקטורים $\vec{v}$ כך ש- $D\vec{v} = 0$ גם אם $D \neq 0$! [[אלה הווקטורים בגרעין, $\ker D$]]

נבצע את האלגוריתם הבא:

- נגריל וקטור $\vec{v}$ בגודל $n \times 1$ מעל $\{0,1\}$

- נחשב את $D\vec{v}$
- אם $D\vec{v}=0$ נחזיר כי $AB=C$ – סיכוי לטעות!
- אחרת נחזיר $AB\neq C$ – אין סיכוי לטעות!

נניח $D\neq 0$, ובפרט $D_{ij}\neq 0$ [[עבור $i,j\in\{1,\dots,n\}$ כלשהם]].
על-מנת לטעות נדרש כי:

$$D_{i1}v_1+\cdots+D_{ij}v_j+\cdots+D_{in}v_n=0$$

[[אנו מניחים כי $\vec{v}=\begin{pmatrix}v_1\\ \vdots\\ v_n\end{pmatrix}$]]

לכן בהכרח $v_j=0$ [הערה: זה **לא נכון**; ראו הסבר בהמשך].

כמה וקטורים $\vec{v}$ יש בהם $v_j=0$?

תשובה: 2^{n-1} [[יש לבחור את ערכי $n-1$ הערכים הנותרים בוקטור]]

אז כמה וקטורים "שקרנים" יש כך ש-$D\vec{v}=0$?

<u>תשובה</u>: לכל היותר 2^{n-1} [[יכולים להיות וקטורים עם $v_j=0$ שלא יניבו $D\vec{v}=0$ (אם יש k כך ש-$(v_k, D_{ik}=1$]]

לכן ההסתברות לטעות הינה:

[להשלים]

כלומר היא $\geq\frac{1}{2}$, וסיבוכיות האלגוריתם היא $O(n^2)$.

[להשלים הסבר]

שיפור הסתברות

- נבצע את התהליך k פעמים
- אם במהלך שלב כלשהו חוזר $AB\neq C$ נסיים ונחזיר כך
- אחרת נענה $AB=C$ – סיכוי לטעות של $\frac{1}{2^k}=2^{-k}$

סיבוכיות:

$$O(k\cdot n^2)$$

תקשורת: בעיית Equality

לאליס ובוב [[Alice & Bob; נקראים כך על-שם A וְ-B]] יש מחרוזות מעל $\{0,1\}^n$.

המחרוזת של אליס: $A=(a_0,\dots,a_n)$

המחרוזת של בוב: $B=(b_0,\dots,b_n)$

כיצד בוב ואליס יאמתו כי יש בידיהם את אותה מחרוזת במינימום תקשורת?

<u>פתרון:</u>

- אליס בונה פולינום: $R_A(x)=\sum_{0}^{n-1}a_ix^i$ $\quad(O(n))$
- אליס מגרילה ראשוני: $q>\alpha n$ [מיד נדבר על מהו α] $\quad(O(n\log(\alpha n)))$
- אליס מגרילה t, $0\le t\le q$ $\quad(O(n\log(\alpha n)))$
- בוב, בדומה, מחשב פולינום $R_B(x)$ $\quad(O(n))$
- בוב בודק אם $R_B(t)\bmod q=R_A(t)\bmod q$ $\quad(O(n))$
- אם שווים, מחזיר "כן"; אחרת "לא"

[שימו לב: המקדמים בפולינום כולם 0 או 1, לכן הפולינום הוא מהצורה $x^0+x^9+x^{48}$ וכו'.]

אם $A=B$ אזי $R_A(x)=R_B(x)$ ולכן ללא קשר לערכי t,q נקבל $R_B(t)\bmod q=R_A(t)\bmod q$ והאלגוריתם לעולם לא יטעה [זוהי טעות חד-צדדית].

ואם $A\neq B$?
אם $R_B(t)\bmod q\neq R_A(t)\bmod q$ אז האלגוריתם יחזיר תשובה שלילית, כנדרש.
מה ההסתברות ש- $R_A(x)\neq R_B(x)$ אך $R_B(t)\bmod q=R_A(t)\bmod q$?

נגדיר $D(x)=R_A(x)-R_B(x)$.
הנ"ל מתקיים אם"ם $D(t)\bmod q=0$.
$D(x)$ הוא פולינום מדרגה $n-1$.
אזי ל- $D(x)$ יש לכל היותר n שורשים [$n-1$ ליתר דיוק, אבל n מספיק טוב לנו, ומפשט את החישובים].

כמה שורשים יש ל- $D(x)\bmod q$?
– לכל היותר n [לא נכונס בדיוק לסיבה לכך, אבל זה נובע מהעובדה שאם $D(x)\bmod q=0$ אז גם $D(x\bmod q)\bmod q=0$].

לכן הסיכוי לטעות הוא:

$$\text{(הסתברות לשגיאה)}=\frac{\text{מס' מספרים "רעים"}}{\text{סה"כ טווח}}\le\frac{n}{q}\le\frac{\cancel{n}}{\alpha\cancel{n}}=\frac{1}{\alpha}$$

סיבוכיות כוללת: $O(n\log n)$

<u>הערות לגבי זמני הריצה</u>:

- במקרה הגרוע ביותר, יש לחשב בהערכת הפולינום את $(\alpha n)^n$ (או מספר קרוב), כי החזקה הגדולה ביותר בפולינום היא $n-1$ (שזה כמעט n).
רק האחסון של התוצאה לוקח $O(n\log(\alpha n))$, וביצוע פעולות כפל עליו זה בכלל זוועתי.
ע"מ להימנע מבעיה זו, ניתן אחרי כל פעולת כפל לבצע $\bmod q$ ועדיין מקבלים את אותה תוצאה מודולו q בסוף.
- איך אנו מוודאים שאנו יכולים למצוא מס' ראשוני אשר $<\alpha n$ בזמן $O(n\log(\alpha n))$?
אנחנו די מרמים: אנחנו מנסים לנחש מספרים ראשוניים במשך $O(n\log(\alpha n))$ זמן – ואם אנחנו לא מוצאים, אנחנו עוצרים את האלגוריתם כולו ומחזירים תשובה: "כן" (הרצפים זהים).
כך אנו חוסמים את זמן הריצה, וכל מה שעלינו לעשות זה לוודא שההסתברות שלא נצליח למצוא ראשוני בזמן זה קטנה מספיק (בכך אנחנו מכניסים את ההסתברות לטעות כאן לתוך הטעות החד-צדדית שלנו).
[[אגב, **חשוב** שהמספר שאנו לוקחים יהיה ראשוני; אחרת הטענות שלא פירטנו לגבי השורשים של הפולינום לא עובדות כי הקבוצה מודולו q איננה חבורה (נראה לי)]]

בעיית התאמת המחרוזות

נתון:

$$P=[P[1],\dots,P[n]]$$
$$T=[T[1],\dots,T[m]]$$
$$(n\ll m)$$

בכמה נקודות אפשר "להציב" את P כך שהיא חופפת ל- T?

<u>פורמלית</u>: יש למצוא קבוצה $S\subseteq\{1,\dots,m-n\}$ כך שלכל $j\in S$ מתקיים:

$$P[1],\dots,P[n]=T[j],\dots,T[j+n]$$

[[קיצר: יש למצוא אלגוריתם שמוצא את כל המיקומים של מחרוזת אחת כתת-מחרוזת של מחרוזת אחרת.]]

[זו שאלה נפוצה בביואינפורמטיקה, שכן עושים זאת הרבה בחיפושים ב-DNA.]

<u>פתרון</u>:

1. $S\leftarrow\varnothing$ (בתרגול $O(1)$)

2. Find prime $q > 4mn$ ($O(n\log n)$)
3. Randomly select $0 \le t \le q$ ($O(n\log n)$)
4. Build $R(x) = \sum_{1}^{n} P[i+1]x^i$ ($O(n)$)
5. Calculate $\sigma = R(t) \bmod q$ ($O(n)$)
6. For $0 \le j \le m-n$:
   6.1. Build $Q_j(x) = \sum_{0}^{n-1} T[j+i+1]x^i$
   6.2. Calculate $\tau_j = Q_j(t) \bmod q$
   6.3. If $\tau_j = \sigma$:
      6.3.1. $S \leftarrow S \cup \{j\}$

   פעם אחת $O(n)$
   ובכל פעם נוספת
   $O(1)$
7. Return S

טענה [ייעול החישוב]

$$Q_j(x) = xQ_{j+1}(x) - T[j+n+1]x^n + T[j+1]$$

[זה מעניין אותנו כי זה אומר שבמקום לחשב את $Q_{j+1}(x)$ בזמן ארוך ($O(n)$) ניתן להשתמש בתוצאת השלב הקודם, $Q_j(x)$, בזמן קצר ($O(1)$).]

<u>הוכחה</u>:

$$Q_j(x) = \sum_{0}^{n-1} T[j+i+1] = T[j+1]x^0 + \sum_{1}^{n-1} T[j+i+1]x^i = \cdots$$

$$\vdots$$

מתמטיקה כלשהי

$$\vdots$$

מש"ל

[הפסקנו משיקולי זמנים (יש דברים יותר חשובים שאנחנו רוצים להספיק בשיעור).]

בהינתן τ_{j+1}, חשב τ_j [כן, אנחנו עושים את זה מהסוף להתחלה כדי שזה יתאים לנוסחה הנ"ל]:

$$\tau_j = \left(t\tau_{j+1} - T[j+n+1]t^n - T[j+1]\right) \bmod q$$

<u>סיבוכיות כוללת</u>: $O(m + n\log n)$

<u>סיכוי לשגיאה:</u>

$$\frac{mn}{q} \le \frac{\cancel{mn}}{4\cancel{mn}} = \frac{1}{4}$$