מודלים חישוביים - הרצאה 9

שרון מלטר, אתגר 17

3 ביוני 2024

היום נתחיל בחלק השני של הקורס, יעילות. או כפי שרונן מעדיף לקרוא לכך, סיבוכיות חישובית.
(כזכור נתייחס ליעילות כיעילות בזמן)

1 הגדרות

עבור מ"ט M דטרמיניסטית ומילה x; $Time(M, x)$ = מספר הצעדים ש- M הריצה על x.
כמובן שמספר זה יכול להיות אינסופי.
זה היה קל. מה לגבי מ"ט לא דטרמיניסטית?

כפי שראינו, ניתן לייצג את מסלולי החישוב השונים של מילה במ"ט לא דטרמיניסטי בעזרת עץ. עם אותו עץ, נגדיר גם את הזמן של הרצת מ"ט לא דטרמיניסטית על מילה.

עבור פונקציה $t : N \to N$ נאמר שמ"ט דטרמיניסטית / לא דטרמיניסטית רצה בזמן $t(\cdot)$ אם לכל מילה x, $Time(M, x) \leqslant t(|x|)$.
נאמר שפונקציה $t : N \to N$ היא פולינומית אם קיים קבוע c כך ש- $t(n) = O(n^c)$. כמו בקורסים קודמים, נחשוב על יעילות כערך ששקול לפולינומיות.

<u>הערה:</u> בהגדרה זו עשינו שתי בחירות:

1. למדוד סיבוכיות כפונקציה של אורך קלט
2. $worst\ case\ complexcity$

ניתן לעשות בחירות אחרות, אך לא נראה זאת בקורס זה :(

ועכשיו להגדרות שכולכם חיכיתם להן-

$DTime(t(\cdot)) = \{L$ כך שיש מ"ט דטרמיניסטית שרצה בזמן $O(t(\cdot))$ שמקבלת את L.$\}$
$NTime(t(\cdot)) = \{L$ כך ש- L מתקבלת על ידי מ"ט לא דטרמיניסטית.$\}$
$P = \cup_{c=1}^{\infty} DTime(n^c)$ כלומר קבוצת המילים שמתקבלות ע"י מ"ט דטרמיניסטית.
$NP = \cup_{c=1}^{\infty} NTime(n^c)$ כלומר קבוצת השפות שמתקבלות ע"י מ"ט לא דטרמיניסטי שרץ בזמן פולינומי.

לפי הבחירות שלנו, P היא קבוצת השפות שניתן לקבל ביעילות. אבל מדוע דווקא פולינומיות = יעילות?
ברור שאלגוריתם שזמן הריצה שלו הוא n^{666} יכול להיות מאוד בעייתי.
התשובה לכך היא שאנו לא מעוניינים בהגדרה ששקולה לחלוטין ליעילות, אלא יותר אכפת לנו מכך שמה לא ייחשב "יעיל", באמת לא יהיה כך. למשל זמן אקספוננציאלי.

1.1 התזה המעודכנת של צרץ' וטיורינג

- כל מה שניתן לחישוב בזמן פולינומי על איזשהו מחשב, ניתן לחישוב בזמן פולינומי במ"ט.

מכיוון שאנו יודעים שניתן לסמלץ כל מחשב שאנחנו מכירים ע"י מ"ט ו- t צעדים במחשב המקורי ייקחו $t^{O(1)}$ צעדים במ"ט.

אם זאת, ישנם שני תחומים מתפתחים שמאיימים על התזה המעודכנת; חישוב הסתברותי וקוונטי.
הדעה הרווחת היום שחישוב הסתברותי כבר לא יפריח את התזה, אך שחישוב קוונטי דווקא כן יעשה זאת.
מדוע? מכיוון שחישוב קוונטי פועל על פי חוקים אחרים משל חישוב הסתברותי.
במחשוב קוונטי, כל ביט נמצא ב'קופסא' מבודדת, בה הוא שרוי במצב סופרפוזיציה. כלומר, לא ניתן לדעת את ערכו עד שפותחים את הקופסא. לכן ישנם מספר אפשרויות לערכי הביטים. מעבר לכך, ישנה התאבכות של אפשרויות; ישנן מספר אפשרויות שונות גם בחישוב הסתברותי, אך רק בחישוב קוונטי ההתאבכות יכולה גם **לבטל** אפשרויות. לכן, אם מצליחים לבטל אפשרויות שאנו לא רוצים, ניתן להאיץ את החישוב. למשל; כאשר רוצים לפרק מספר ניתן לבטל את בדיקת האפשרויות שאינן מחלקות אותו.
כך ניתן להתייחס לחישוב קוונטי כעל חישוב אי־דטרמיניסטי שיכול לא לקחת בחשבון מסלולים שאינם רלוונטיים.

2 אלגוריתמים פולינומיים

עשינו קורס שלם על אלגוריתמים פולינומיים־ תכנון וניתוח אלגוריתמים. לא נצטרך לחשוב הרבה בשביל למצוא דוגמאות לאלג' פולינומיים.
אחד האלגוריתמים הראשונים שלמדנו בקורס הוא BFS, אך הוא גם מאוד מחוכם. הוא רץ בזמן לינארי על מחשב ובזמן לינארי על מ"ט. הוא מחשב את השפה הבאה;

$$PATH = \{< G, s, t >\}$$

כאשר G הוא גרף, s, t הם קודקודים בו וקיים מסלול מ־ s ל־ t. בזכות BFS מתקיים $PATH \in P$.

2.1 מציאת מסלולים המילטוניים

עכשיו נתייחס לשפה יותר מסובכת;

$$HAM-PATH = \{< G, s, t >\}$$

כך ש־ G הוא גרף עם הקודקודים s, t וקיים מסלול המילטוני מ־ s ל־ t. (מסלול בו עוברים בכל קודקוד בדיוק פעם אחת)
פיתרון ה־ $brute-force$ עבור הבעיה הוא פשוט לעבור על כל מסלולי הגרף, כך שישנו פתרון אקספוננציאלי.
האם $HAM-PATH \in P$? ־ אנו לא יודעים. אך כעת נוכיח ש־ $HAM-PATH \in EXP$ כאשר

$$EXP = \cup_{c=1}^{\infty} DTime(t(2^{n^c}))$$

בהינתן גרף G נסמן ב־ n את מספר הקודקודים וב־ m את מספר הקשתות.
נסמן ב־ $< G >$ את קידוד הגרף במטריצת שכנויות, כך ש־ $|< G >| = n^2$. כמובן שריבוע ה־ n לא משנה, מכיוון שפולינומיות ב־ n שקולה לפולינומיות ב־ n^2.

מסלול מאורך $t = r$ בגרף אפשר לקודד ע"י מחרוזת y מאורך $t(logn)$. בהינתן גרף G וקודקודים s, t עבור על המחרוזות y מאורך $n(logn)$ ולכל y כזה הרץ פונקציה $Verify(< G, s, t >, y)$ שבודקת האם y הוא קידוד של מסלול המילטוני מ־ s ל־ t.
אם כן, קבל. אם סיימנו לעבור על כל ה־ y־ים, דחה.

$Verify(< G, s, t >, y)$ תבדוק האם:

1. הקודקוד הראשון ב־ y הוא s
2. הקודקוד האחרון ב־ y הוא t
3. כל הקודקודים ב־ y שונים
4. לכל קודקוד ב־ y יש קשת לקודקוד שאחריו

אם כל הבדיקות התקבלו, קבל.

<u>הסיבוכיות של $Verify$:</u>

1. $O(n)$

2. $O(n)$
3. $O(n^2)$
4. $O(n^4)$

סיבוכיות הזמן פולינומית ל־ n, כך שהיא פולינומית ל־ n^2 ולכן פולינומית לקלט.
נסמן את התכנית שיצרנו ב־ M.
ההוכחה ש־ $L(M) = HAM-PATH$ הינה טכנית וקלה.

<u>טענה:</u> $HAM-PATH \in NP$

<u>הוכחה:</u> נבנה מ"ט לא דטרמיניסטית M שמקבלת את $HAM-PATH$.

<u>M:</u> בהינתן $<G,s,t>$ נבצע "ניחוש" לא דטרמינסיטי ואז נבצע "בדיקה" דטרמיניסטית. נציג כיצד מבצעים את שני השלבים.

<u>ניחוש:</u>
ניכנס למצב אחר בכל פעם ונבצע איתו $nlogn$ פעמים את המסלול y. ניעזר בספירה עם מ"ט כדי לדעת מתי ביצענו $|y|$ צעדים ונצטרך להחזיר את השליטה.

<u>בדיקה:</u>
אם הקלט התקבל, נריץ את $verify$.
כפי שראינו $verify$ הינה פולינומית.

בסך הכל המ"ט הנתונה מנחשת את כל האפשרויות ואז בודקת אותן. לכן ההוכחה טכנית וקלה :)

3 מוודא של שפה מ־ NP

<u>משפט:</u>
$L \in N \Leftrightarrow$ ל־ L יש מוודא פולינומי.

3.1 דוגמאות

הדוגמה הראשונה שראינו היא השפה $HAM-PATH$. נעבור לדוגמה הבאה;

$$CLIQUE = \{<G,k>\}$$

כך ש־ G הינו גרף וב־ G יש קליקה מגודל k.

<u>טענה:</u> $CLIQUE \in NP$

<u>הוכחה:</u>
כפי שנוחש, נבנה מ"ט M לא דטרמיניסטית שמקבלת את $CLIQUE$ ורצה בזמן פולינומי.
האינטואציה היא שנבדוק כל קבוצת צמתים אפשרית בגודל k. באופן כללי, הפרדיגמה היא תמיד לבדוק את כל האפשרויות־ $Exhaustive\ Search$.
בהינתן גרף על n קודקודים וקבוצת קודקודים S מגודל k, נקודד אותה ע"י מחרוזת מעל הא"ב $0,1$ מאורך $klogn$. (כמובן ש־ $logn$ זהו אורך קידוד n הצמתים)

כדי להרחיב על כך, ניזכר מעט באלגברה בוליאנית:
להלן נוסחה:

$$(x_1 \vee x_2) \wedge ((x_3 \vee (x_3 \vee (x_1 \vee x_4)))$$

נוסחה על n משתנים $x_1, ..., x_n$ מורכבת ע"י צירוף המשתנים באמצעות הפעולות $\hat{}, \wedge, \vee$.
הנוסחה מקודדת ע"י מחרוזת
*

טענה: קיימת שפה NP שלמה.

4 בפרקים הבאים

נצטרך להוכיח את משפט קוק־לוין ושקיימת שפה NP שלמה.