תורת הסיבוכיות – תרגול 0
חזרה על סיבוכיות זמן וזכרון

המודל הבסיסי

המודל העיקרי שבו נעסוק בקורס זה הוא **מכונת טיורינג רב־סרטית**, בעלת **סרט קלט** לקריאה בלבד (אנו מניחים שהקלט מופיע בו יחד עם שני סימנים מיוחדים המציינים את תחילת וסוף הקלט), **וסרט עבודה** הניתן לקריאה ולכתיבה.

הסיבה לשימוש במודל זה היא כדי לאפשר דיון עדין יותר בסיבוכיות זכרון–במודל הרגיל קשה לתת משמעות לסיבוכיות זכרון שקטנה מלינארית מכיוון שקריאת הקלט כולו כבר תוביל (לפי ההגדרה הקלאסית) לסיבוכיות זכרון לינארית. עם זאת, ישנם חישובים שבהם נדרשת כמות פחותה בהרבה של זכרון; כך למשל כל שפה רגולרית ניתנת לזיהוי עם $O\left(1\right)$ זכרון נוסף. בהמשך הקורס נראה כי חיפוש בגרף ניתן לביצוע עם כמות פולי־לוגריתמית של זכרון נוסף, וכדומה.

מושג מרכזי הוא מושג ה**קונפיגורציה** של מכונת טיורינג, המתאר את תמונת המצב הרגעית שלה בריצתה על קלט כלשהו. הקונפיגורציה מורכבת משלושה פרטי מידע:

1. מצב הבקרה שבו המכונה נמצאת.
2. מיקום הראשים על גבי סרטי המכונה.
3. תוכן סרט העבודה (סרט הקלט אינו משתנה ולכן אין צורך לשמור אותו במפורש כחלק מהקונפיגורציה).

ה**קונפיגורציה ההתחלתית** של המכונה בריצתה על קלט x היא זו שבה מצב הבקרה הוא q_0 (המצב ההתחלתי), סרט העבודה ריק, וכל הראשים בתחילת הסרטים (וסרט הקלט מכיל את x, אך כאמור זה אינו מידע שהכרחי לשמור עם הקונפיגורציה).

קונפיגורציה סופית של המכונה היא קונפיגורציה שבה מצב הבקרה הוא מצב סופי (בדרך כלל, במכונות לזיהוי שפות, זהו q_{acc} או q_{rej}).

במכונת טיורינג **דטרמיניסטית** לכל קונפיגורציה יש **קונפיגורציה עוקבת** חוקית יחידה (שתלויה בקונפיגורציה וב־x שעליו רצה המכונה). במכונה **אי־דטרמיניסטית** זה לא כך וישנן מספר קונפיגורציות המשך אפשריות.

מסלול חישוב של המכונה M על קלט x הוא סדרה של קונפיגורציות עוקבות המתחילות מהקונפיגורציה ההתחלתית של M על x (סדרה זו עשויה להסתיים בקונפיגורציה סופית אך היא יכולה גם להיות אינסופית).

המכונה M **מקבלת** את הקלט x אם יש לה מסלול חישוב על x שמסתיים בקונפיגורציה סופית מקבלת (שבה מצב הבקרה הוא q_{acc}).

סיבוכיות זמן

פונקציה $t:\mathbb{N}\to\mathbb{N}$ היא **חסם זמן** עבור מכונה M אם לכל קלט x, אורך כל מסלולי החישוב של M על x קטן או שווה ל־$t\left(\left|x\right|\right)$.

בהינתן שפה $L\subseteq\Sigma^{*}$, נאמר כי $L\in\mathsf{DTIME}\left(t\left(n\right)\right)$ אם קיימת מכונה <u>דטרמיניסטית</u> M בעלת חסם זמן $O\left(t\left(n\right)\right)$ המקבלת את L.

בדומה, נאמר כי $L\in\mathsf{NTIME}\left(t\left(n\right)\right)$ אם קיימת מכונה <u>אי־דטרמיניסטית</u> M בעלת חסם זמן $O\left(t\left(n\right)\right)$ המקבלת את L.

כעת ניתן להגדיר פורמלית מספר מחלקות מוכרות:

$$\mathsf{P}\triangleq\bigcup_{c>0}\mathsf{DTIME}\left(n^{c}\right)$$

$$\mathsf{NP}\triangleq\bigcup_{c>0}\mathsf{NTIME}\left(n^{c}\right)$$

$$\mathsf{EXP}\triangleq\bigcup_{c>0}\mathsf{DTIME}\left(2^{n^{c}}\right)$$

אבחנות בסיסיות:

1. לכל $t(n)$ מתקיים $\mathsf{DTIME}(t(n)) \subseteq \mathsf{NTIME}(t(n))$ (מכונה דטרמיניסטית ניתנת לתיאור כמקרה פרטי של מכונה אי־דטרמיניסטית).
2. $\mathsf{DTIME}(t(n))$ סגורה למשלים (היפוך תפקידי q_{acc} ו־q_{rej} עושה את העבודה).
3. לא ידוע אם $\mathsf{NTIME}(t(n))$ סגורה למשלים (חוסר הסימטריה בהגדרת הקבלה של מכונה אי־דטרמיניסטית מקלקל את הפתרון הנאיבי).
4. אם $t_1(n) \le t_2(n)$ לכל n אז $\mathsf{DTIME}(t_1(n)) \subseteq \mathsf{DTIME}(t_2(n))$, ובדומה עבור NTIME (תוספת משאבים לא פוגעת בכוח החישוב).
5. $\mathsf{NTIME}(t(n)) \subseteq \mathsf{DTIME}\left(2^{O(t(n))}\right)$ (סריקת DFS דטרמיניסטית של עץ החישוב האי־דטרמיניסטי).

סיבוכיות זכרון

פונקציה $s:\mathbb{N}\to\mathbb{N}$ היא **חסם זכרון** עבור מכונה M אם לכל קלט x, מספר התאים בסרט העבודה בהם M קטן או שווה ל־$s(|x|)$.

כזכור, סרט הקלט הינו לקריאה בלבד ולכן ההגדרה שלנו לסיבוכיות זכרון ממדלת את הרעיון של זכרון נוסף שבו משתמשים במהלך החישוב בנוסף לזכרון שדורש הקלט.

הגדרת המחלקות עבור סיבוכיות זכרון אנלוגית לזו שעבור סיבוכיות זמן:

בהינתן שפה $L\subseteq\Sigma^*$, נאמר כי $L\in\mathsf{DSPACE}(s(n))$ אם קיימת מכונה דטרמיניסטית M בעלת חסם זכרון $O(s(n))$ המקבלת את L.

בדומה, נאמר כי $L\in\mathsf{NSPACE}(s(n))$ אם קיימת מכונה אי־דטרמיניסטית M בעלת חסם זכרון $O(s(n))$ המקבלת את L.

כעת ניתן להגדיר פורמלית שתי מחלקות מוכרות:

$\mathsf{PSPACE} \triangleq \bigcup_{c>0}\mathsf{DSPACE}(n^c)$

$\mathsf{NPSPACE} \triangleq \bigcup_{c>0}\mathsf{NSPACE}(n^c)$

אבחנות בסיסיות:

1. $\mathsf{DSPACE}(s(n)) \subseteq \mathsf{NSPACE}(s(n))$ (בהמשך הקורס, משפט $\mathrm{Savitch}$ יראה קשר חזק המתקיים בכיוון השני ומוכיח בין היתר כי $\mathsf{PSPACE}=\mathsf{NPSPACE}$).
2. $\mathsf{DTIME}(t(n)) \subseteq \mathsf{DSPACE}(t(n))$ ו־$\mathsf{NTIME}(t(n)) \subseteq (t(n))$ (לכל תא זכרון חדש שמנוצל, נדרש לפחות צעד אחד כדי להגיע אליו).
3. $\mathsf{DSPACE}(s(n)) \subseteq \mathsf{DTIME}\left(2^{O(s(n))}\right)$ עבור $s(n)\ge\log n$ (על ידי זיהוי קונפיגורציה שמופיעה פעמיים במהלך הריצה; עבור $s(n)<\log n$ הנוסחה נראית מעט שונה).

ספירת קונפיגורציות

נרחיב על סעיף 3. למכונה בעלת $k=|Q|$ מצבי בקרה, עם אלפבית סרט בגודל $t=|\Gamma|$, ובעלת חסם זכרון $s(n)$, יש לכל היותר מספר של $k\cdot(n+2)\cdot s(n)\cdot t^{s(n)}$ קונפיגורציות אפשריות בריצתה על קלט מאורך $n=|x|$ (שלשות של מצב בקרה + מיקום הראשים + תוכן סרט העבודה). אם $s(n)\ge\log n$ אז $t^{s(n)}\ge n$, ולכן מספר הקונפיגורציות האפשרי הוא לכל היותר $c^{s(n)}$ עבור קבוע $c>0$ מסויים.

רדוקציות ובעיות שלמות

רדוקציה היא תרגום של בעיה אחת למושגים של בעיה אחרת. פורמלית, אם $L_1,L_2\subseteq\Sigma^*$ הן שפות, אז רדוקציה מ־L_1 אל L_2 היא פונקציה מלאה $f:\Sigma^*\to\Sigma^*$ המקיימת $f(x)\in L_2 \iff x\in L_1$.

האינטואיציה שמאחורי רדוקציות היא זו: אם אנו עוסקים במחלקה C כלשהי, וידוע כי $L_2\in\mathsf{C}$ וכמו כן L_1 ניתנת לרדוקציה אל L_2, אז גם $L_1\in\mathsf{C}$. לצורך כך יש להחיל מגבלות כלשהן על הרדוקציה שנובעות מאופי המחלקה C. בחישוביות ראינו כי כאשר $\mathsf{C}=\mathsf{R}$ היה עלינו לדרוש כי f תהיה ניתנת לחישוב, וכאשר $\mathsf{C}=\mathsf{P}$ היה עלינו לדרוש כי f תהיה ניתנת לחישוב בזמן פולינומי.

אם C' היא מחלקה שמכילה את C ואנו תוהים לגבי שוויון או אי־שוויון ביניהן, יש עניין מיוחד בבעיות שהן C'־שלמות: בעיות ב־C' שכל בעיה אחרת ב־C' ניתנת לרדוקציה אליהן (רדוקציה העונה למגבלות הרגילות שאנו משיתים על רדוקציה ביחס למחלקה C). לבעיות אלו התכונה שאם הן שייכות ל־C אז $\mathsf{C}=\mathsf{C}'$ (וכמובן, אם הן אינן ב־C אז $\mathsf{C}\ne\mathsf{C}'$).

עבור המקרה של P ו־NP ראינו בחישוביות את קיומן של שפות NP־שלמות רבות, ובראשן SAT.

עבור המקרה של P ו־PSPACE נרמז בחישוביות על קיום שפה PSPACE־שלמה בשם TQBF. בקורס זה נוכיח כי היא אכן PSPACE־שלמה (שימו לב: ביחס לרדוקציות בזמן פולינומי ולא במקום פולינומי - מדוע?)