תורת החישוביות – תרגול מספר 2
התזה של צ'רץ' וטיורינג ומודל ה־RAM

מבוא

הסכנה הגדולה שבמחקר מתמטי של מושג ה"חישוב" המעורפל היא בהתמקדות במודל מוגבל, שאינו תופס את מושג החישוב במשמעותו הכללית. בצורה זו תוצאות קושי אמנם יראו כי המודל חלש במובן מסויים, אך לא יהיו תקפות עבור סוגים כלליים יותר של חישוב. התזה של צ'רץ' וטיורינג היא ההשערה (שאינה ניתנת להוכחה מתמטית, שכן הוכחה כזו תדרוש מלכתחילה קיום פורמליזה מתמטית של מושג החישוב) שכל המודלים הסבירים והכלליים עבור חישוב שקולים זה לזה בכוחם החישובי, כלומר מסוגלים לחשב בדיוק את אותה מחלקה של פונקציות. לצורך העניין מודל "כללי" הוא כזה שמסוגל לעשות לפחות מה שמכונת טיורינג מסוגלת לעשות, ו"סביר" פירושו שזהו מודל שניתן להעלות על הדעת מימוש שלו בעולם האמיתי (קל לממש מכונת טיורינג, אם כי לא בטוח שניתן יהיה לספק לה במהלך ריצתה די סרט עבודה – אך במגבלות מסוג זה עוסק חציו השני של הקורס). כאשר מנסים לתת תיאור קונקרטי יותר של ה"סבירות" דורשים בדרך כלל כי המודל יהיה ניתן לתיאור סופי וכי כל צעד חישובי בסיסי שהוא מבצע יהיה פשוט דיו עד שניתן לבצע אותו באופן מכני וללא מחשבה. מכונת טיורינג בבירור עונה לדרישות אלו.

החל משנות ה־30 של המאה ה־20, מודלים מתמטיים לחישוב החלו לצוץ בעבודותיהם של מתמטיקאים, החל מקורט גדל במשפטי אי־השלמות שלו (שבהם לדרישה שהוכחות בתורה המתמטית שעליה מפעילים את המשפטים יהיו ניתנות לבדיקה אלגוריתמית היא קריטית). גדל הציע (באופן עקיף, שכן זו לא הייתה מטרתו) את המודל של פונקציות רקורסיביות. צ'רץ' הציע את המודל של תחשיב λ. טיורינג הציע את מכונת טיורינג. שלושת המודלים הללו פותחו בנפרד זה מזה ובמטרות שונות, אך שלושתם התגלו לבסוף כשקולים מבחינת כוחם החישובי. זה הביא את מדעני המחשב להעלות את ההשערה כי מודלים אלו אכן מצליחים לתאר בדיוק את הפונקציות שניתנות לחישוב באופן "אפקטיבי".

בשנים שחלפו מאז הועלתה התזה הוצעו עוד מודלים סבירים וכלליים רבים לחישוב, וגם הם התגלו כשקולים למודלים הקיימים. מודל שזוכה לפרסום בספרות המדע הפופולרית כ"חזק יותר" ממכונת טיורינג הוא המודל של חישוב קוונטי; אולם גם חישוב קוונטי שקול, מבחינת מחלקת הפונקציות שניתן לחשב בעזרתו, לחישוב קלאסי. ההבדל הוא בסיבוכיות החישובים; יש סיבות להאמין (אך אין לבינתיים הוכחה) כי בעזרת חישובים קוונטיים ניתן לחשב ביעילות פונקציות שלא ניתנות לחישוב ביעילות המודל הקלאסי (על משמעות המושג "חישוב יעיל" נדון בחלקו השני של הקורס).

בתרגול זה נתאר מודל חישובי הקרוב באופיו למחשב כפי שאנו מכירים אותו, ונתאר את האופן שבו מוכיחים את השקילות שלו למכונת טיורינג. בכך "נשתכנע" שבהמשך הקורס, כאשר רוצים לבצע חישוב מסויים עם מכונת טיורינג, די במתן פסאודו־קוד או תיאור של אלגוריתם שבבירור ניתן לממש במחשב, והדבר יבטיח כי קיימת מכונת טיורינג שמממשת את האלגוריתם.

מודל ה־RAM – מה זה?

במה מכונת טיורינג נבדלת ממחשב רגיל? כמובן, למכונת טיורינג יש כמות זכרון בלתי חסומה ("סרט אינסופי") אבל באותה מידה אפשר לחשוב על מחשב בעל כמות זכרון בלתי חסומה (בכל פעם שבה יתמלא התקן זכרון אחד נחבר לו אחר), כך שההבדלים הם בעיקר לרעת מכונת טיורינג:

1. מודל הזכרון של מכונת טיורינג הוא **סדרתי** - כדי להגיע לתא זכרון מסויים, הראש צריך לעבור על פני כל התאים שבין המיקום הנוכחי שלו, והתא שאליו הוא רוצה להגיע. זכרון של מחשבים אמתיים הוא (בנפנוף ידיים כלשהו) מבוסס על **גישה אקראית** (Random Access - מכאן ה־RAM) - כדי להגיע לתא זכרון כלשהו פונים ישירות אליו.

2. למעבד ישנם **רגיסטרים** שבהם ניתן לשמור מידע זמני, בעוד שמכונת טיורינג נאלצת להסתפק רק בתאי הסרט הראשי.

3. מעבד מסוגל לבצע **פקודות אריתמטיות־לוגיות** בעוד שמכונת טיורינג יודעת רק לקרוא ולכתוב תא בודד בסרט ותו לא.

4. תוכנית המחשב שמריץ המעבד שמורה בזכרון, והמחשב יכול לקרוא אותה ולבצע עליה פעולות ואף לשנות אותה, בעוד שבמכונת טיורינג התוכנית קבועה ומקודדת בפונקציית המעברים, ולא ניתן לשנותה.

כפי שנראה בקרוב, כל ההבדלים הללו הם **חסרי כל חשיבות**.

המודל שנציע "תופס" את כל ההבדלים הללו. כל מכונה במודל בנויה מהמרכיבים הבאים.

1. זכרון אינסופי **בשני הכיוונים**. כל תא זכרון הוא בעל כתובת שהיא מספר שלם (חיובי, אפס או שלילי). כל תא זכרון יכול להכיל כל מספר שלם אפשרי (אין מגבלות על גודל המילה כמו במחשבים אמיתיים).

2. n רגיסטרים, שכל אחד מהם יכול להכיל מספר שלם כלשהו. הרגיסטרים מסומנים ב־$r_1, r_2, \ldots, r_n$. בנוסף ישנם שלושה רגיסטרים מיוחדים: רגיסטר קלט IN, רגיסטר פלט OUT ורגיסטר בקרת התוכנית PC.

3. סט פקודות שהמכונה יכולה לבצע:

   (א) $r_i \leftarrow (r_j)$: Load – מכניס לרגיסטר r_i את תוכן תא הזכרון שכתובתו היא הערך שב־r_j.

   (ב) $r_i \rightarrow (r_j)$: Store – מכניס מכניס לתא הזכרון שכתובתו היא הערך שב־r_j את תוכן הרגיסטר r_i.

   (ג) $r_i \leftarrow r_i + r_j$: Add – מוסיף לרגיסטר r_i את הערך שברגיסטר r_j.

   (ד) $r_i \leftarrow r_i - r_j$: Sub – מפחית מהרגיסטר r_i את הערך שברגיסטר r_j.

   (ה) $r_i \leftarrow r_i + 1$: Inc – מוסיף 1 לערך שברגיסטר r_i.

   (ו) $r_i \leftarrow r_i - 1$: Dec – מפחית 1 מהערך שברגיסטר r_i.

   (ז) $\mathrm{PC} \leftarrow r_j$ if $r_i > 0$: BGT – מעביר את התוכנית לתא r_j בזכרון אם $r_i > 0$.

   (ח) HALT – עוצר את התוכנית.

   אפשר להוסיף עוד פקודות כיד הדמיון הטובה – ככל שנדרש כדי להשתכנע שמודל ה־RAM מסוגל לסמלץ מעבד של מחשב אמיתי. הפקודות עצמן מקודדות בעזרת מספרים טבעיים.

4. "תוכנית מחשב" סופית שהמכונה מבצעת. כאשר המכונה מתחילה את ריצתה, התוכנית כתובה בזכרון החל מתא 0 והלאה.

על קלט x, המכונה מתחילה את ריצתה כאשר ברגיסטר IN כתוב x, בשאר הרגיסטרים כתוב 0 וכך גם בזכרון כולו למעט האיזור בו שבו נכתבה התוכנית של המכונה. בכל צעד חישוב שלה המכונה קוראת את תוכן התא (PC) (התא שבכתובת שמכיל PC), מגדילה את PC ב־1 ומבצעת את הפקודה. אם בשלב כלשהו התוכנית מבצעת את הפקודה HALT, הפלט שלה הוא תוכן הרגיסטר OUT.

שקילות למכונת טיורינג

הנחת היסוד שלנו היא שמודל ה־RAM חזק דיו כדי להיות מסוגל לסמלץ כל מחשב שקיים כיום, ולכן גם ניתן להריץ עליו כל שפת תכנות קיימת. כדי להראות שהמודל חזק לפחות כמו מכונת טיורינג, אם כן, די להציג קוד של תוכנית מחשב שמסמלצת מכונת טיורינג, וזהו תרגיל נחמד למשועממים (אתגר: ניתן לסמלץ מכונת טיורינג בתוך מנגנון ה־Templates של C++ כך שהמכונה תרוץ בשלב ה־Preprocessing).

הכיוון השני הוא המאתגר יותר. השלב הראשון הוא להרחיב את המודל הפשוט של מכונת טיורינג למודל חזק מעט יותר שיקל על סימולציה של מכונת RAM – המודל הרב סרטי. במודל זה (שהשקילות אליו מתוארת בהרצאה) למכונה יש מספר **קבוע** וגדול מאחד של סרטים כך שלכל סרט ראש קורא וכותב משל עצמו, ופעולת המכונה בכל סיבוב נקבעת על בסיס מה שכל הראשים רואים (מספר אינסופי של סרטים יהפוך את המודל לחזק מדי – הוא יכול לחשב כל פונקציה שהיא – ולבלתי סביר).

כל רגיסטר יסומלץ בידי סרט משל עצמו. את המספרים ניתן לייצג בבסיס בינארי, ומספרים שליליים יסומנו בתו " $-$ " בתחילתם. ראינו איך לבצע את פקודת Inc בתרגול הקודם, ופקודות אריתמטיות אחרות ניתנות לביצוע בצורה דומה.

הזכרון, לעומת זאת, מציב אתגר גדול יותר – עלינו לתאר אינסוף תאים שכל אחד מהם יכול להכיל מספר שאינו חסום בגודלו, ולאפשר גישה ישירה אליהם. נציג דרך "עצלנית" במיוחד לעשות זאת.

הזכרון כולו יסומלץ באמצעות סרט בודד, שתוכנו הוא מהצורה $@A_1\#C_1\$A_2\#C_2\$\ldots$, כאשר $@, \#, \$$ הם סימנים מיוחדים, A_i, C_i הם מספרים שלמים. A_i מייצג כתובת, ו־C_i מייצג תוכן, והזוג $A_i\#C_i$ פירושו "בתא שכתובתו A_i מאוחסן המספר C_i".

כמו כן נאפשר לשנות את $\#$ ל־X; הסימון A_iXC_i פירושו יהיה "תשכחו מכל מה שכתוב כאן, זה כבר לא רלוונטי".

קריאה מהתא A_i תבוצע באופן הבא: סורקים את הסרט עד שמתגלה רצף התווים $A_i\#$ (מתעלמים מ־A_iX), ואז מעתיקים את התוכן של מה שמופיע אחרי ה־$\#$ אבל לפני ה־$\$$ אל היעד המתאים. אם לא נתגלה $A_i\#$, זה אומר שטרם נכתב ל־A_i מאום ולכן מחזירים 0.

כתיבה לתא A_i תבוצע באופן הבא: סורקים את הסרט, ואם מתגלה $A_i\#$ משנים את ה־$\#$ ל־X. כשמגיעים לקצה הסרט כותבים $A_i\#C_i\$$, כאשר C_i הוא התוכן המיועד.

נותר לטפל בבעיות של אתחול וסיום הריצה. כאשר מכונת הטיורינג מזהה שמכונת ה־RAM הפעילה פקודת HALT, היא עצמה מעתיקה את תוכן סרט ה־OUT אל הסרט הראשון שלה ומעבירה את הראש אל מימין לתוכן הזה (שכן כך מוגדר הפלט של המכונה).

האתחול מציב אתגר גדול מעט יותר. ראשית, יש להעביר את תוכן הסרט הראשון (שהוא הקלט למכונת הטיורינג) אל סרט ה־IN. שנית, המכונה צריכה לכתוב את קוד תוכנית המחשב שמודל ה־RAM שהיא מסמלצת מריץ אל תוך סרט הזכרון (כזכור, הנחנו שהתוכנית הזו "מופיעה באופן פלאי" בתוך סרט הזכרון של מכונת ה־RAM). לשם כך אנו מסתמכים על כך שתוכנית המחשב של מכונת ה־RAM היא מחרוזת סופית באורכה, ומכונת טיורינג יכולה להכיל כחלק מהקידוד שלה עצמה מספר סופי של מחרוזות מאורך סופי ולעשות בהן כרצונה.

מודל ה־RAM ותפקידו בחקר סיבוכיות של אלגוריתמים

קורסים קודמים, כגון אלגוריתמים, ואפילו מבוא למדעי המחשב, מסתמכים למעשה על מודל ה־RAM למרות שאינם מגדירים אותו במפורש. כאשר מנתחים אלגוריתם וקובעים שהסיבוכיות שלו היא (למשל) $O(n)$, אין משמעות לטענה הזו אם לא נתון מודל חישובי מסויים. ניתוחים כאלו מתבססים לרוב על מודל ה־RAM, או על גרסה דומה שלו (בדרך־כלל מאפשרים פעולות אריתמטיות "עשירות" יותר ממה שהנחנו, ולעתים יש צורך לשים מגבלה על גודל הערך שאפשר לאחסן בתא בודד בזיכרון).

הסימולציה של מודל ה־RAM בידי מכונת טיורינג כפי שראינו אינה משמרת את הסיבוכיות של האלגוריתם. בחלקו הראשון של הקורס, כאשר אנו מתעסקים בשאלה מה ניתן ומה לא ניתן לחשב, אין לכך חשיבות, ודי לנו בטענה שהמודלים שקולים. בחלקו השני של הקורס נתעניין גם בשאלה מה ניתן לחשב באופן יעיל; בעיקר נתרכז בשאלה מה ניתן לחישוב בזמן פולינומיאלי. למרות שהסימולציה שראינו איננה משמרת את הסיבוכיות המדויקת של אלגוריתמים, היא כן משמרת את ה"פולינומיות" שלהם. כלומר, אלגוריתם שניתן לחישוב בזמן פולינומיאלי במכונת טיורינג, ניתן לחישוב בזמן פולינומיאלי גם במודל ה־RAM, ולהיפך. מסיבה זו, המסקנות שנסיק בחלקו השני של הקורס לגבי חישוב יעיל במכונת טיורינג, תקיפות גם עבור מודל ה־RAM, וגם עבור מחשב מודרני.