# סיבוכיות

[בפרק זה נדבר על האפשרות לפתור בעיות, ועל סדר גודל זמן הריצה של הפתרון האפשרי.]

## דוגמאות לבעיות קלות / קשות

| קל | קשה (כנראה) |
|---|---|
| מסלול אוילר (עובר בכל הקשתות)<br>[ניתן לפתור בזמן פולינומי] | מסלול המילטון (עובר בכל הקדקודים)<br>[ניתן לפתור בזמן אקספוננציאלי] |
| חתך מינימלי<br>[מוצאים בעזרת מציאת זרימה מקסימלית] | חתך מקסימלי (אפילו בגרף לא ממושקל<br>[ניתן למצוא בעזרת מעבר על כל החתכים האפשריים (אקספוננציאלי)] |
| 2-צביעת מפות<br>4-צביעת מפות | 3-צביעת מפות |
| | שולה המוקשים<br>[נסביר בהמשך] |

[לגבי הדוגמאות הקשות מאמינים כיום שלא ניתן לפתור בפחות זמן מכך (אך זה לא הוכח).]

### בעיית k-צביעת מפות

- <u>מופע</u>: מפה [[ ניתן לייצג ע"י גרף ]]
- <u>פלט</u>: האם יש צביעה של כל המדינות לכל היותר ב- $k$ צבעים [[ כלומר עם $\geq k$ צבעים ]] כך שכל שתי מדינות שכנות צבועות בצבעים שונים.

<u>דוגמה</u>:

[ציור]

ניתן לראות כאן שכל שכנותיה של המדינה הצבועה בירוק חייבות להיות צבועות בצבע שאינו ירוק, וגם לא ניתן לצבוע שתיים מהן באותו הצבע. לכן חייבים להשתמש לפחות ב-4 צבעים.
את המדינה הנותרת כבר ניתן לצבוע בצבע קיים – אדום או ירוק.

#### עבור המקרה k=1

פשוט נבדוק אם יש שתי מדינות שכנות (אולי יכולים להיות איים שונים באותו הצבע).

#### עבור המקרה k=2

מבחינה תאורטית, הדבר אפשרי אם"ם אין מעגל של מדינות שכנות מאורך אי-זוגי.
אבל ע"מ לבצע אלגוריתם שבודק זאת, ניתן פשוט לבחור מדינה ולתת לה צבע כלשהו.
מאותו רגע, שאר הצביעה (לאותה יבשת) היא חד-משמעית (יש רק אפשרות אחת).
אז פשוט ממשיכים לצבוע מדינות עד שצובעים את כולן (ואז זה אפשרי) או שמגיעים לסתירה (ואז זה בלתי-אפשרי).

#### עבור המקרה k>=4

<u>משפט (Appel-Haken '76)</u>: **<u>כל מפה</u>** ניתן לצבוע ב-4 צבעים.

לכן אלגוריתם שפותר זאת צריך פשוט להחזיר את התשובה "כן," בלי תלות במפה.

#### עבור המקרה k=3
מסתבר שזו כבר בעיה קשה.

### שמות הקבוצות בטבלה והשאלה הגדולה של מדעי המחשב
לקבוצת הבעיות ה"קלות" קוראים $P$ , והבעיות הקשות נקראות $NP$ -שלמות.

אחת מבעיות המילניום של מכון קלֵי (Clay) למתמטיקה:
"האם אפשר לפתור בעיה כזו [ $NP$ -שלמה] בזמן פולינומי?"
יש עליה כיום פרס של $1,000,000.

## הגדרות לא פורמליות
$P$ = Polynomial time = מחלקת בעיות שניתן לפתור בזמן פולינומי [כלומר שניתן לפתור בזמן יעיל].

$NP$ = Non-deterministic polynomial time = מחלקת בעיות שבהינתן פתרון קל [=ניתן לבצע בזמן פולינומי] לבדוק שהוא פתרון נכון [אך אנחנו לאו-דווקא יודעים למצוא פתרון בעצמנו].

### דוגמה לבעיה ב-NP [שולה המוקשים]
[שולה המוקשים (Minesweeper) הוא משחק מפורסם של מערכת ההפעלה Microsoft Windows.
בכל משבצת יש או מוקש או מספר שמתאר את כמות המוקשים שנמצאים מסביב לאותה משבצת (מתוך 8 המשבצות שמקיפות אותה – או פחות, עבור משבצות שנמצאות בקצוות המפה).

[להשלים הסבר ותמונות]

#### בעיית עקביות שולה המוקשים
- מופע: לוח $n \times n$ שבחלק מתאיו יש או מספר או מוקש (או לא מסומן)
- פלט: האם יש קונפיגורציית מוקשים (שכוללת את המוקשים המסומנים) שמתאימה למספרים שמופיעים?

[להשלים דוגמאות]

#### *טענה [טענה זו נמצאת ב-NP]*
צ"ל: קיים אלגוריתם [יעיל] שבודק אם פתרון הוא חוקי.

אלגוריתם: (מקבל גם קלט וגם פתרון)

- נבדוק שהפתרון הוא אכן טבלה של קונפיגורציית מוקשים [כלומר נבדוק שה-syntax תקין; זה קל לביצוע]
- נעבור על כל התאים בטבלה המקורית:
  - אם בתא המקורי יש מוקש, נבדוק שבפתרון יש מוקש באותו מקום

- אם בתא מופיע מספר $k$, נעבור על התאים שמסביבו, נספור את מספר המוקשים בפתרון ונוודא שיש $k$ כאלה.

זמן ריצה: $O(1)$ לכל תא $\Leftarrow O(n^2)$ (לינארי בגודל הקלט [הקלט הוא טבלה בגודל $n^2$]).

## חידה

מה הספרה ה-1000 אחרי הנקודה העשרונית במספר $\left(1+\sqrt{2}\right)^{4000}$?

<u>תשובה</u>: 9

<u>הסבר</u>:
נתבונן בביטויים:

$$\left(1+\sqrt{2}\right)^n = 1+n\sqrt{2}+\binom{n}{2}\left(\sqrt{2}\right)^2+\binom{n}{3}\left(\sqrt{2}\right)^3+\cdots$$
$$\left(1-\sqrt{2}\right)^n = 1-n\sqrt{2}+\binom{n}{2}\left(\sqrt{2}\right)^2-\binom{n}{3}\left(\sqrt{2}\right)^3+\cdots$$
$$\Rightarrow x=\left(1+\sqrt{2}\right)^n+\left(1-\sqrt{2}\right)^n\in\mathbb{N}$$

כעת:

$$\left(1+\sqrt{2}\right)^{4000}=x-\left(1-\sqrt{2}\right)^{4000}=x-\left(\sqrt{2}-1\right)^{4000}$$

לכן:

$$\underbrace{\left(\sqrt{2}-1\right)}_{\approx 0.41\ldots<\frac{1}{2}}^{4000}<\frac{1}{2^{4000}}=\frac{1}{\left(\underbrace{2^{10}}_{=1024}\right)^{400}}<\frac{1}{\left(\underbrace{10^3}_{=1000}\right)^{400}}=10^{-1200}$$

כלומר אנו מחסרים מ-$x$ שלנו ערך ש-1200 הספרות הראשונות שלו אחרי הנקודה הן 0.
לכן כיוון ש-$x$ שלם, בתוצאת החיסור ב-1200 הספרות הראשונות אחרי הנקודה תהיה הספרה 9 – בפרט בספרה ה-1000.

$$\underbrace{****\cdots*}_{\in\mathbb{N}}.\underbrace{999\ldots\overset{\overset{1000}{\downarrow}}{9}\ldots 99}_{1200}***\ldots$$

מכון Clay הציע $10^6$ דולר עבור פתרון לשאלה: האם $P=NP$?
[מדוע זה כ"כ מעניין?
מתישהו בעבר קבוצת אנשים רצתה לבנות אלגוריתם שיוכיח באופן שיטתי את כל השאלות הפתוחות במתמטיקה. אז בא גדל (Gödel) והראה שיש משפטים נכונים שלעולם לא נוכל להוכיח.

אז שינו את הגישה, ואמרו "בסדר, אז נבנה אלגוריתם שיוכיח את כל מה **שאפשר** להוכיח."
אז בא Alan Turing והציג את מכונת טיורינג ע"מ להראות שיש בעיות שניתן להוכיח אך לא ניתן לבנות מכונה שתוכיח אותן.
אז עלתה השאלה הבאה: מה לגבי בדיקת האם הוכחה נתונה היא נכונה?
כאן נכנס העניין של $P$ ו- $NP$.]

## בעיות חיפוש והכרעה

- <u>בעיית חיפוש</u>: בהינתן קלט, למצוא פתרון חוקי (אם קיים).
- <u>בעיית הכרעה</u>: בהינתן קלט, האם קיים פתרון חוקי? (הפלט: כן / לא)

<u>דוגמה</u>:

- בעיית חיפוש: מציאת צביעה של מפה ב-3 צבעים (אם קיימת).
- בעיית הכרעה: בהינתן מפה, האם קיימת 3-צביעה חוקית? (כן / לא)

<u>אבחנה</u>: תמיד בעיית החיפוש תהיה קשה לפחות כמו בעיית ההכרעה המתאימה [כי תשובה לבעיית החיפוש גוררת מיד תשובה לבעיית ההכרעה].

[הערה: בד"כ זה אם"ם – כלומר בעיית ההכרעה קשה באותה מידה. למשל בבעיית הצביעה, אם יש אלגוריתם שרק עונה כן / לא לגבי האם קיימת צביעה, ניתן להשתמש בו בשביל למצוא צביעה חוקית שכזו (לא נוכיח זאת כאן).]

## הגדרות [שפה, P, NP, אלגוריתם אימות, עד]

- בהינתן בעיית הכרעה, שפה היא קבוצת המחרוזות $\{\, x \mid$ התשובה לקלט $x$ היא "כן" $\}$.
- $P=$ מחלקת כל השפות שמתאימות לבעיית הכרעה פתירה בזמן פולינומי (בגודל הקלט, ע"י מכונת טיורינג).
[[ אמרנו בעבר שמה שניתן לעשות בזמן פולינומי במכונת טיורינג ניתן גם לעשות בעזרת מכונת RAM או כל מודל חישובי שקול אחר (ולהיפך). בקיצור, הקטע הזה לא כ"כ מהותי; הוא פשוט נוח יותר לעבודה תאורטית. ]]
[[ פולינומי בגודל הקלט = אם הקלט הוא מגודל $n$, אז זמן הריצה הוא $O\left(p(n)\right)$ כאשר $p$ פולינום כלשהו. אם $n=2^m$, וזמן הריצה הוא $2^{8m+3}=2^3\cdot\left(2^m\right)^8$, זהו זמן ריצה פולינומי. ]]
- $NP=$ כל השפות $L$ שקיים עבורן אלגוריתם אימות.
- אלגוריתם עימות $A$ עבור $L$ מקבל:
  - $x$ – קלט לבעיה
  - $w$ – עֵד [[ witness ]] (באורך פולינומי ב- $|x|$)

  [זהו פתרון לדוגמה. למשל בבעיית המפות, המפה היא הקלט $x$, ופתרון (צביעה) לדוגמה $w$ הינה עד.]

  מתקיים:

- אם $x \in L$ (התשובה אמורה להיות "כן") אזי קיים עד (פתרון לבעיית החיפוש) $w$ כך ש- $A(x,w)$ מחזיר "כן".
- אם $x \notin L$ אזי $A(x,w)$ **תמיד** יחזיר "לא" (לכל $w$).

אינטואיציה:

- $x =$ מופע
- $w =$ פתרון לבעיית חיפוש
- $A =$ אלגוריתם שבודק האם $w$ פתרון חוקי למופע $x$

## שפת הקבוצה הבלתי תלויה IS = (Independent Set)

- מופע: גרף $G=(V,E)$, ומספר $k$
- המופע שייך ל- $IS$ אם קיימת קבוצת קדקודים $U \subseteq V$ בגודל $k \le |U|$ כך שאף קשת לא מוכלת ב-[[ תת הגרף המושרה ע"י ]] $U$ (אם $(u,v) \in E$ אזי או ש- $u \notin U$, או $v \notin U$, או שניהם).

### טענה [היא נמצאת ב-NP]

$$IS \in NP$$

הוכחה:

צ"ל שקיים אלגוריתם פולינומי $A$ כך שלכל מופע $G,k$:

- אם קיימת קבוצה בלתי-תלויה ב- $G$ בגודל $k \le$ אזי קיים עד (פתרון) $w$ בגודל פולינומי ב- $|G,k|$ כך ש- $A((G,k),w)$ מחזיר "כן".
- אחרת – $A((G,k),w)$ תמיד יחזיר "לא" לכל $w$.

האלגוריתם:

- $A$ מקבל עד $w$, בודק ש- $w = U \subseteq V$ (קבוצת קדקודים) ובודק ש- $k \le |U|$.
- עובר על כל קשת $(u,v) \in E$ ובודק האם $u \notin U$ או $v \notin U$.

גודל העד (הפתרון): $|V| \ge |U|$ – פולינומי (אפילו קטן-שווה) בגודל הקלט, $(G=(V,E),k)$.

[[ נכונות האלגוריתם: ]]

- אם $(G,k) \in IS$ אזי קיימת קבוצה $U$ לפי הדרישות, והאלגוריתם יוודא שזה פתרון חוקי.

- אם $(G,k)\notin IS$, אזי אין פתרון חוקי $U$, ולכן האלגוריתם **תמיד** יזהה שהעד אינו פתרון כזה.

<u>זמן ריצת האלגוריתם</u>:

- $O(|V|)$ לבדוק ש- $U\subseteq V$ וְ- $|U|\geq k$.
- $O(|E|)$ לבדוק את כל הקשתות.
- סה"כ:

$$O(|V|+|E|) = \text{פולינומי בגודל הקלט והעד}$$

□