20

NP-Completeness - 4 נושא

א) מטרה

בפרק זה נלמד על מדד שבאמצעותו נקבע האם בעיה היא קשה או קלה. הקריטריונים שיקבעו זאת: משך זמן הריצה וגודל הזיכרון הנדרשים לפתרון הבעיה. שניהם ביחד נקראים "סיבוכיות הבעיה". בעיה תיקרא "קלה" אם הסיבוכיות שלה פולינומית ו"קשה" אם לא פולינומית.

אמנם למדוד מדד כזה זה לא פשוט, שהרי ישנם סוגים שונים של בעיות. נחלק ל-3 סוגים:

(1) בעיות הכרעה - האם קיים משהו כלשהו. לדוגמא, האם קיים בגרף G מעגל המילטון

(2) בעיות חיפוש - לא רק להכריע אלא גם למצוא משהו. לדוגמא, מצא ב-G מעגל המילטון.

(3) בעיות אופטימיזציה - למצוא משהו אופטימלי. לדוגמא, מצא מעגל המילטון עם משקל מינימלי.

בפרק הבא ננסה להראות (באמצעות דוגמא) שכל סוגי הבעיות אלו שקולות מבחינת סיבוכיות לבעיות הכרעה. כלומר אם קיים פתרון לבעיית הכרעה בזמן פולינומי, אזי קיים פתרון פולינומי לשני הסוגים האחרים. כתוצאה מכך נוכל להתרכז בהם בלבד.

ב) רדוקציה עצמית

טענה: אם קיים אלגוריתם לפתרון בעיית הכרעה בזמן פולינומי, אזי גם קיים אלגוריתם לפתרון בעיית חיפוש מתאימה בזמן פולינומי.

הוכחה: באמצעות דוגמא.

בהינתן גרף G נסמן $\omega(G)$ את גודל התת גרף השלם הגדול ביותר ב-G. אם לדוגמא הקליקה המקסימלית ב-G היא K_4 אזי $\omega(G)=4$. נגדיר שפה $K\text{-Clique}:=\{(G,k):\omega(G)\geq k\}$

בעיית הכרעה: בהינתן גרף G ומספר k נרצה להכריע האם (G,k) שייך לשפה הזו.

בעיית חיפוש: בהינתן גרף G ומספר k נרצה להחזיר קליקה K_k שנמצאת ב-G.

אם ידוע כי קיימת פונקציה $A(G,k)$ שמחזירה 0 אם לא שייכת לשפה ו-1 אם כן, נוכל לפתור את בעיית החיפוש באלגוריתם הבא:

```
Search(G,k) {
    if A(G,k)=0 return no K_k
    while |V(G)|>k {
        find v∈V(G)
        if A(G-v,k)=1 , set G=G-v
    }
    return G
}
```

ניתן לראות שאם פתרון בעיית הכרעה בזמן פולינומי כך הוא גם עבור בעיית החיפוש. כך נוכל לעשות עבור כל בעיה.

6/4

21

ג) מכונת טיורינג דטרמיניסטית - DTM

מורכבת מחמישה חלקים:

(1) Σ - אוסף של א"ב כלשהו. בדרך כלל $\Sigma = \{0,1\}$.

(2) סרט נע - סרט חד-צדדי אינסופי שיש עליו תאים. ובכל תא אות אחת מ-Σ.

(3) ראש קורא - רכיב שיכול לקרוא מתא בסרט נע, ללכת ימינה, שמאלה או להישאר במקום.

(4) מערכת מצבים - מסומנים ב-Q. שני מצבים מיוחדים q_{start} ו-q_{halt}.

(5) פונקציית מעברים - פונקציה המקבלת מצב ואות מהראש קורא ומחזירה שלישייה של:

פעולה לראש קורא, מצב עבור אליו, ואות לכתיבה בסרט. $\delta: Q \times \Sigma \to \{Left, \overset{Stay}{\Delta}, Right\} \times Q \times \Sigma$.

בתחילת החישוב תמיד נמצאים במצב התחלתי ובתא הראשון בסרט נע. נגדיר "צעד חישוב" הפעלה אחת של δ.

ניגוד לזה פוטס המחשב מודרני. כל אריתמ שאנו יריק הוא DTM.

קבלת מילה ושפה

יש המון ניסוחים לעתי מילה מתקבלת. אנו נגדיר ש-DTM, אותו נסמן ב-M, יקבל מילה w

אם לאחר קריאת המילה הגיע למצב q_{halt} והסרט נע נכתב המצב "accept". אחרת,

המילה לא מתקבלת.

ל גם להיות 1 בציורים כולאני כל סימן אחר שמילה התקבלה,

נסמן ב-$L(M)$ את שפת כל המילים ש-M מקבלת.

תיאור בעיה כפונקציה

נוכל להגדיר כל בעיה ופתרונה באמצעות פונקציה $F: \{0,1\}^* \to \{0,1\}^*$, כאשר F מקבל קלט

$x \in \{0,1\}^*$ כלשהו ומחזירה לו את הפלט הרצוי $F(x)$, שהוא פתרון הבעיה עבור אותו קלט.

נאמר שמכונת טיורינג דטרמיניסטית M "מחשבת" פונקציה $F: \{0,1\}^* \to \{0,1\}^*$ אם לכל $x \in \{0,1\}^*$

M עוצרת ב-q_{halt} וכותבת בסרט נע את $F(x)$, כלומר קיים תיאור של מכונת טיורינג

פותרת בעיה כלשהי.

נשים לב כי בעיות הכרעה ניתן לתאר באמצעות פונקציה $F: \{0,1\}^* \to \{0,1\}$.

ו. כל בעיית הכרעה ניתן לתאר כשפה L כל המילים פה הפלטות 1, וכל המילים אינן בשפה מקבלות 0.

זמן ריצה

נאמר ש-DTM M מחשבת פונקציה $F: \{0,1\}^* \to \{0,1\}^*$ בזמן ריצה של פונקציה $T: \mathbb{N} \to \mathbb{N}$,

אם לכל קלט $x \in \{0,1\}^n$ באורך n, כאשר $n \in \mathbb{N}$, M עוצרת על q_{halt} לאחר $T(n)$

צעדי חישוב וכותבת את $F(x)$ על הסרט נע.

במילים אחרות, זמן הריצה של M הוא פונקציה T אם עבור כל קלט באורך n M פותר

את הבעיה לאחר $T(n)$ צעדי חישוב.

6/4

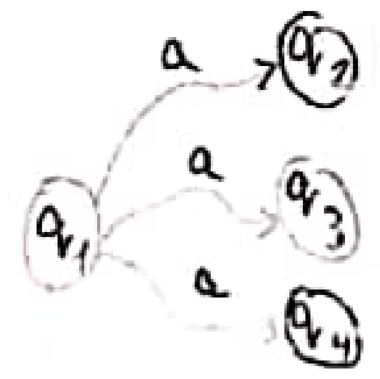
3

בס"ד

מחלקת בעיות P

* נסמן ב-$DTIME(T(n))$ קבוצת כל בעיות ההכרעה להם קיימת DTM שפותרת אותן בזמן ריצה של $T(n)$.
* נסמן ב-P את קבוצת כל בעיות ההכרעה להן קיימת DTM שפותרת אותן בזמן ריצה פולינומי. $\boxed{P := \bigcup_{c\geq 1} DTIME(n^c)}$

3) מכונת טיורינג אי-דטרמיניסטית - NDTM



NDTM זהה בהגדרה כמעט ל-DTM מלבד שינוי אחד. במקום פונקציית מעברים קיים בו יחס מעברים, שהוא בעצם תת-קבוצה של המכפלה הקרטזית $\delta \subseteq (Q\times\Sigma)\times(\{L,\Delta,R\}\times Q\times\Sigma)$.
שינוי זה מאפשר שלכל זוג סדור של מצב ואות $(q,a)\in Q\times\Sigma$ יכולים להיות מספר מעברים שהמכונה יכולה לבחור איזה מהם לבצע, בניגוד ל-DTM שיש לו רק אפשרות אחת לכל זוג סדור כזה.

קבלת מילה

NDTM מקבל מילה ω אם קיים איזשהו רצף של מעברים על המילה כך שהמכונה בסיום ω תגיע ל-q_{acc} והסרט נע תכתוב "accept".* במילים אחרות, נניח תמיד כי ה-NDTM "בוחר נכון" את המעבר שיגרום לו לקבל את המילה אם קיים כזה.

*יכול גם להיות 1 בשאריתם הבוליאני, כל סימן אחר אשר שהמילה תקבלה.

זמן ריצה

הגדרת זמן הריצה ב-NDTM זהה ל-NDTM. כלומר נאמר כי NDTM רץ בזמן ריצה של $T(n)$ אם לכל קלט x באורך n $x\in\{0,1\}^n$, המכונה עוצרת לאחר $T(n)$ צעדי חישוב.
בקורס אוטומטים למדנו שהמודל הדטרמיניסטי והאי-דטרמיניסטי שקולים. אמנם זה נכון רק כאשר מתעלמים מזמני הריצה. אנו מתחשבים בזמני ריצה ולכן מבחינתנו הם לא שקולים.

מחלקת בעיות NP

* נסמן ב-$NTIME(T(n))$ את קבוצת בעיות ההכרעה להם קיימת NDTM שפותרת אותן ב-$T(n)$.
הגדרה נוספת היא קבוצת כל השפות L להם קיים NDTM M המקיים $L=L(M)$.
* נסמן ב-NP את קבוצת כל הבעיות להן קיימת NDTM הפותרת אותן בזמן ריצה פולינומי. $\boxed{NP := \bigcup_{c\geq 1} NTIME(n^c)}$
* נסמן ב-CONP את קבוצת כל הבעיות שאינן ב-NP, כלומר שלא קיים להם NDTM הפותר אותן בזמן ריצה פולינומי.

6/4

23

בס״ד

ה) הגדרה נוספת ל-NP - אלגוריתמי וידוא

כיוון שההגדרה של מ״טמא מאוד מופשטת, נרצה למצוא הגדרה אינטואיטיבית יותר ל-NP ולדעת מתי בעיה משתייכת לקבוצה זו באמצעות מכונת טיורינג דטרמיניסטית מ״טד.
אם ישנה שפה $L\subseteq\{0,1\}^*$ (המייצגת בעיה) השייכת ל-NP, אזי ישנו מ״טמא M שמקבל אותה בזמן ריצה פולינומי, לכן עבור כל $x\in\{0,1\}^*$ כך ש-$x\in L$ קיים רצף של מעברים, שנסמן אותו ב-y, שבהם עברנו ב-M כדי לקבל את x. אך אם $x\notin L$ משמע שאין רצף y כזה שאם נריץ עליו את x היא תתקבל. נשים לב שמכיוון ש-$L\in NP$, y ת״ב להיות בגודל פולינומי.
באמצעות תכונה זו נוכל לבנות DTM, הנקרא גם "אלגוריתם וידוא", שיבדוק האם שפה שייכת ל-NP.

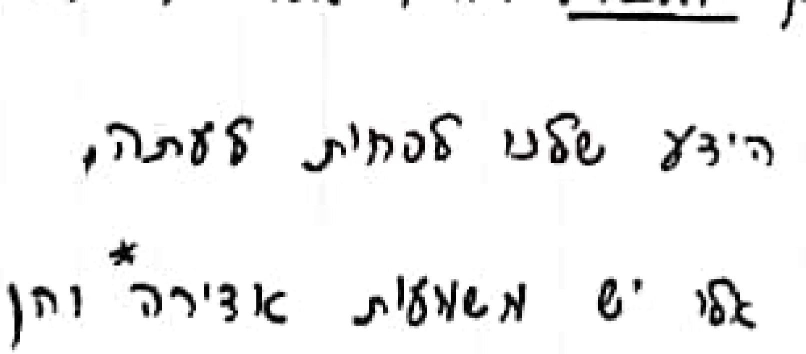
נאמר כי שפה $L\subseteq\{0,1\}^*$ שייכת ל-NP אם קיים איזשהו DTM M בזמן ריצה פולינומי T, כך שלכל קלט $x\in\{0,1\}^*$ מתקיים ש-$x\in L$ אם ורק אם יש ב-M איזשהו רצף מעברים y פולינומי שאם M יקבל כקלט את (x,y) אזי היא יקבל את המילה ב-T(|x|) צעדי חישוב.
במילים אחרות, בעיה שייכת ל-NP אם קיים DTM שאם יקבל מופע של הבעיה ואת הפתרון למופע שהוא פולינומי, ה-DTM ידע לבדוק האם זהו אכן פתרון למופע. אם אין מ״טד כזה אזי הבע״ה שייכת ל-coNP.

<u>דוגמא:</u> עבור בעיית ההכרעה "האם קיים בגרף מעגל המילטון", נוכל לבנות אלגוריתם (DTM) שיקבל גרף G ומעגל המילטון c בגודל פולינומי למספר הצמתים ב-G, ויוכל לבדוק האם c אכן המילטוני. לכן בע״ה זו שייכת ל-NP. אמנם הבע״ה ההפוכה לבדוק האם אין מעגל המילטון היא coNP, מפני שלא נוכל להכניס את כל תתי הקבוצות של V(G) ולוודא שהם אכן לא מעגל המילטון, כי מספרם לא פולינומי.

לכ $(\underset{x}{G},\underset{y}{c})$

ו) P=NP ?

למדנו על שלושה קבוצות בעיות P, NP, coNP. מקבוצה P עבור כל מופע של בע״ה אנו יודעים בשוטף לפתור ולא להכניס את הפתרון לאלגוריתם וידוא, לכן <u>$P\subseteq NP$</u>. אמנם אנו גם יודעים גם להכריע מתי אין פתרון, לכן גם <u>$P\subseteq coNP$</u>*. בה נאמר הידע שלנו לפחות לעתה, איננו יודעים האם P=NP או האם NP=coNP. לשאלות אלו יש משמעות אדירה* והן נשארו חידה גדולה במדעי המחשב עד היום.



דוגמא: אנו יודעים להכריע מתי גרף קשיר אך אנו גם יודעים מתי גרף אינו קשיר.

*דוגמא: כל תחום ההצפנה מובנה על פקטוריזציה השייכת ל-NP.

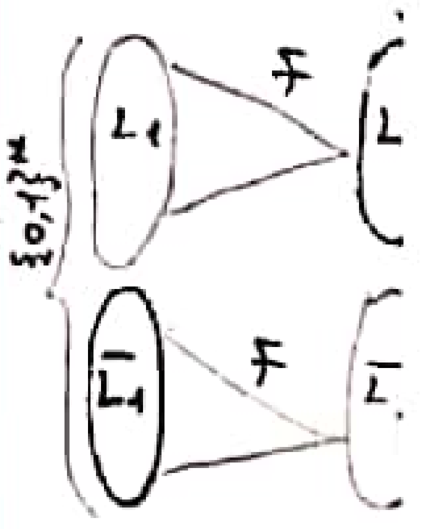
ג) <u>רדוקציה בזמן פולינומי</u>

בהינתן שתי שפות $L_1,L_2\subseteq\{0,1\}^*$ המתארות שתי בעיות הכרעה, נאמר כי ניתן לעשות רדוקציה מ-L_1 ל-L_2, בסימון $L_1\leq_p L_2$; אם קיימת פונקציה $f:\{0,1\}^*\to\{0,1\}^*$ המקיימת את התנאים הבאים:

(1) $x\in L_1$ אם ורק אם $f(x)\in L_2$. כלומר אם x מתקבל ב-L_1 אזי $f(x)$ יתקבל ב-L_2, וכן ההיפך.

(2) לכל $x\in\{0,1\}^*$, $f(x)$ מחושב בזמן פולינומי לפי אורכו של x.

אם קיימת f כזו נקרא לה "רדוקציה בזמן פולינומי".



המשמעות של רדוקציה כזו בין שתי שפות המקיימות $L_1\leq_p L_2$, היא שאם נמצא אלגוריתם הפותר את L_2 נוכל בקלות למצוא פתרון גם ל-L_1 בדרך הבאה: עבור כל $x\in\{0,1\}^*$ המייצג מופע של הבעיה, אם $f(x)$ מתקבל ב-L_2 אזי גם יתקבל ב-L_1 ואם לא מתקבל ב-L_2 גם לא יתקבל ב-L_1. כלומר ניתן להסיק את הבעיה של L_1 ל-L_2.

ד) <u>NP-HARD ו- NP-COMPLETE</u>

<u>NP-HARD</u> - קבוצת כל הבעיות שעבור כל אחת מהן ניתן לבצע אליהן רדוקציה מכל אחת מהשפות ב-NP. במילים אחרות, אם קיימת שפה $L\subseteq\{0,1\}^*$ המקיימת $L'\leq_p L, \forall L'\in NP$, אזי $L\in NP\text{-}HARD$. לא כל השפות ב-NP-HARD הן גם ב-NP. נסמן בקיצור NPH.

<u>NP-COMPLETE</u> - קבוצת כל הבעיות שהן גם ב-NP וגם ב-NPH, $L\in NP\cap NPH$. נסמן בקיצור NPC.

בעיות אלו חשובות מכיוון שאם נמצא אחת מהן שיש לה פתרון פולינומי, כלומר שהיא גם ב-P, אז הוכחנו ש-$P=NP$.

ה) <u>תיאוריית COOK-LEVIN - CNF-SAT</u>

הם שני האנשים הראשונים שהצליחו למצוא ולהוכיח בעיה מ-NPC. מבעיה זו הצליחו למצוא עוד המון בעיות ב-NPC.

הבעיה נקראת CNF-SAT והיא מתוארת כך: בהינתן n משתנים בוליאניים, $x_1,x_2,\dots,x_n$, נגדיר לכל משתנה x_i ליטרל שאם ערכו כערך המשתנה נסמנו x_i ואם השלילה שלו $\bar{x}_i$.*

נגדיר נוסחה בוליאנית φ, המורכבת מליטרלים אלו, ונקראת CNF אם מורכבת מפסוקיות* שביניהם יש $\cap$ אך בתוכן יש רק $\cup$. לדוגמא, $\varphi=(x_1\cup\bar{x}_5\cup x_7\cup x_2)\cap x_3\cap(\bar{x}_4\cup x_6)$.

הבעיה היא בהינתן נוסחת CNF תכריע האם יש השמה כלשהי של המשתנים $x_1,x_2,\dots,x_n$ כך שהנוסחה כולה תקבל ערך "true" אחיד. כל נוסחאות CNF הספיקות נקראות CNF-SAT.

התיאוריה COOK-LEVIN מוכיחה שבעיה זו שייכת ל-NPC. נציג הוכחה בקורס חישוביות.

אם x_1 = false
כי הליטרל $x_1=F$
ו- $\bar{x}_1=T$. וכן ההפך.

*כלומר שאם יש יותר מליטרל אחד יהיו סוגריים.

*אם יש השמה כזו אז שהנוסחה ספיקה.

1) 3-CNF-SAT

נגדיר נוסחה בוליאנית 3-CNF להיות נוסחת CNF שבה כל פסוקית בדיוק באורך של 3 ליטרלים. לדוגמא, $\varphi=(x_1\cup\bar{x}_4\cup x_3)\cap(\bar{x}_5\cup\bar{x}_2\cup x_4)$. נגדיר 3-CNF-SAT להיות כל הנוסחאות 3-CNF הספיקות. נראה כי בעיה זו גם היא NPC. לשם כך יש להוכיח 2 דברים:

(1) $3\text{-}CNF\text{-}SAT \in NP$

נוכיח באמצעות אלגוריתם וידוא. נבנה אלגוריתם שמקבל כקלט נוסחת 3-CNF-SAT φ, והשמה של n משתנים בוליאניים, $a:\{x_1,x_2,\dots,x_n\}\to\{0,1\}$. האלגוריתם יבדוק האם a מספקת את φ. אם כן יקבל ואם לא ידחה. מכיוון שניתן לבנות אלגוריתם וידוא כזה בזמן פולינומי $3\text{-}CNF\text{-}SAT\in NP$.

(2) $3\text{-}CNF\text{-}SAT \in NPH$

נוכיח זאת ע"י רדוקציה פולינומית מ-CNF-SAT אל 3-CNF-SAT, $CNF\text{-}SAT \le_p 3\text{-}CNF\text{-}SAT$.

כדי לעשות זאת נגדיר פונקציה F שמקבלת כקלט נוסחת CNF, φ, והפלט שלה $F(\varphi)$ היא נוסחת 3-CNF. עבור כל פסוקית $c=l_1\cup l_2\cup\dots\cup l_m$ ב-φ, F תבצע את הפעולה הבאה בהתאם לאורך c:

- אם $m=3$ - תעתיק את c כמו שהיא.
- אם $m<3$ - תחבר את אחד מהליטרלים שבה ב-c מספר פעמים עד שיהיו 3 ליטרלים. $c=\begin{cases} l_1 \Rightarrow l_1\cup l_1\cup l_1 \\ l_1\cup l_2 \Rightarrow l_1\cup l_2\cup l_2\end{cases}$
- אם $m>3$ - אז תבנה $m-3$ משתנים חדשים $y_1,y_2,\dots,y_{m-3}$ שיופיעו אך ורק בפסוקית c, ויחלקו אותה בדרך הבאה: $(l_1\cup l_2\cup y_1)\cap(\bar{y}_1\cup l_3\cup y_2)\cap(\bar{y}_2\cup l_4\cup y_3)\cap\dots\cap(\bar{y}_{m-3}\cup l_{m-1}\cup l_m)$.

כדי להוכיח שזוהי אכן רדוקציה פולינומית יש להוכיח שתי תכונות:

① $\varphi\in CNF\text{-}SAT \iff F(\varphi)\in 3\text{-}CNF\text{-}SAT$: כיוון 1- נניח $\varphi\in CNF\text{-}SAT$ עם השמה $a:\{x_1,x_2,\dots,x_n\}\to\{0,1\}$ של כל המשתנים ב-φ. נגדיר השמה a' שעבור כל המשתנים המקוריים $x_1,\dots,x_n$, a' מעתיק את הערכים מ-a.

עבור כל פסוקית ב-φ, $l_1,l_2,\dots,l_m$, ת"פ להיות לפחות $i\in[m]$ אחד עבורו l_i היא true, אחרת φ לא הייתה ספיקה. לכן בכל פסוקית c ב-φ, שאותה פיצלנו לתת פסוקיות ב-$F(\varphi)$, לכל $j<i-1$ נגדיר $a'(y_j)=1$, ולכל $j\geq i-1$ נגדיר $a'(y_j)=0$. $(\overset{0}{l_1}\cup l_2\cup \overset{1}{y_1})\cap(\overset{0}{\bar{y}_1}\cup l_3\cup \overset{1}{y_2})\cap\dots\cap(\overset{0}{\bar{y}_{i-2}}, \overset{1}{l_i}, \overset{0}{y_{i-1}})\cap\dots\cap(\overset{1}{\bar{y}_{m-3}}\cup l_{m-1}\cup l_m)$.

נשים לב שבכל פסוקית ב-φ כל התת פסוקיות החדשות שנוצרו ב-$F(\varphi)$ יש בהן 1. לכן עבור ההשמה a' $F(\varphi)$ תקבל true, מכאן ש-$F(\varphi)$ ספיקה.

כיוון 2 - נניח ש-$F(\varphi)$ ספיקה עם השמה a'. נגדיר השמה a שעושה כל מה שהמשתנים המקוריים ב-a' עושים. נניח בשלילה ש-φ לא ספיקה. אז קיימת ב-φ פסוקית c עם יותר מ-3 ליטרלים שכולם 0. אם כך בפסוקית המקבילה ב-$F(\varphi)$ נקבל סתירה. $(\overset{0}{l_1}\cup \overset{0}{l_2}\cup \overset{1}{y_1})\cap(\overset{0}{\bar{y}_1}\cup \overset{0}{l_3}\cup \overset{1}{y_2})\cap\dots\cap(\cancel{\overset{0}{\bar{y}_{m-3}}}\cup \overset{0}{l_{m-1}}, \overset{0}{l_m})$

הפסוקית המקבילה ב-$F(\varphi)$ צריכה להתקבל ואנו אין שום דרך לסדר את a' בהתאם ל-a כך שפסוקית זו תתקבל ב-$F(\varphi)$ אך תתקבל ב-φ. סתירה!

② לכל $x\in\{0,1\}^*$ F מחשבת את $F(x)$ בזמן פולינומי: טריוויאלי, מספיק מעבר אחד על כל φ ולשנות בהתאם.

20/4

26

בס"ד

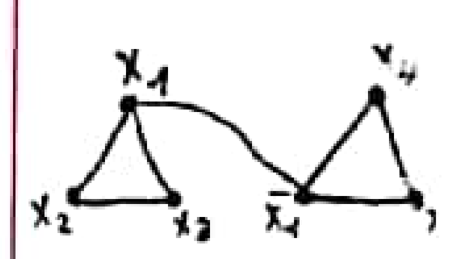
יא) קבוצה בלתי-תלויה

היא קבוצה של צמתים שאין אף צלע בין כל שניים מהם. בגרף G מספר הצמתים בקבוצה הבלתי תלויה המקסימלית מסומן ב- $\alpha(G)$. נגדיר את השפה IS להיות כל הזוגות הסדורים (G,k) כך שבגרף G יש קבוצה בלתי תלויה עם לפחות k צמתים. $IS=\{(G,k) : \alpha(G) \geq k\}$.

שפה זו מייצגת את בעיית ההכרעה האם ב-G יש קבוצה ב"ת בגודל k. נוכיח ש-$IS \in NPC$.

<u>$IS \in NP$:</u> די טריוויאלי שכן בקלות ניתן לבנות אלגוריתם ודא שמקבל גרף וקבוצה ב"ת בגודל k ובודק האם הוא אכן ב"ת.

<u>$IS \in NPH$:</u> באמצעות רדוקציה מ- 3-CNF-SAT. נגדיר פונקציה $f: 3\text{-}CNF\text{-}SAT \to IS$. תהי נוסחת 3-CNF φ, f תיצור מ-φ גרף G_φ בדרך הבאה: עבור כל פסוקית ב-φ תיצור בגרף G_φ משולש K_3 שהצמתים בו הם תוויות עבור כל ליטרל בפסוקית. הקשתות בין המשולשים יהיו רק בין שני צמתים שהם תוויות של הליטרלים x_i ו-$\bar{x}_i$. כל פסוקית מייצגת 3 צמתים נפרדים אפילו אם הליטרל מופיע בפסוקיות שונות. גודל הקבוצה הב"ת שמחפשים ב-G_φ הוא m מספר הפסוקיות ב-φ.



$(x_1 \cup x_2 \cup x_3) \cap (\bar{x}_1 \cup x_4 \cup \ldots$

(1) $\varphi \in 3\text{-}CNF\text{-}SAT \iff (G,m) \in IS$: כיוון 1 - נניח ש- $\varphi \in 3\text{-}CNF\text{-}SAT$ כך שיש השמה עבורה φ מקבל ערך true. מכל פסוקית ב-φ ניקח ליטרל אחד שבהשמה זו מקבל true. ת"ב להיות לפחות אחד כזה מכל פסוקית אחרת φ לא הייתה מתקבלת, קיבלנו m ליטרלים שבגרף G_φ מהווים קבוצה בלתי תלויה, שכן לקחנו צומת מכל משולש, ואם היה קשת בין שני צמתים מקבוצה זו זה היה בין x_i ל-$\bar{x}_i$ ולכן לא יכול להיות ששניהם true ב-φ.

אם φ ספיקה זוהי גם הקבוצה הב"ת המקסימלית, כי G_φ מורכבת מ-m רכיבים של משולשים.

כיוון 2 - נניח שב-G_φ יש קבוצה I בלתי תלויה בגודל m. אזי כל צומת בקבוצה זו חייב להיות ממשולש שונה. $\forall \Delta\ |I \cap \Delta| = 1$. נגדיר השמה a שעבור כל צומת ב-I הליטרל המתאים ב-φ יקבל true. השמה כזו בוודאי קיימת שכן החשש היחיד הוא ששני ליטרלים x_i ו-$\bar{x}_i$ יקבלו true, אולם זה לא יכול להיות שכן אז היה בין הצמתים קשת והם לא היו ב-I. מכיוון שבהשמה זו כל הפסוקיות מקבלות true φ מתקבלת.

(2) לכל $\varphi \in 3\text{-}CNF$ f תחשב את G_φ בזמן פולינומי. טריוויאלי. מסקנה: $3\text{-}CNF\text{-}SAT \leq_p IS$

יב) צביעה ב-3 צבעים - 3-col

הקבוצה 3-col מכילה את כל הגרפים שניתן לצבוע אותם בלכל היותר 3 צבעים. $3\text{-}col = \{G : \chi(G) \leq 3\}$.

3-col מייצגת את בעיית ההכרעה האם ניתן לצבוע גרף במקסימום 3 צבעים. נוכיח $3\text{-}col \in NPC$.

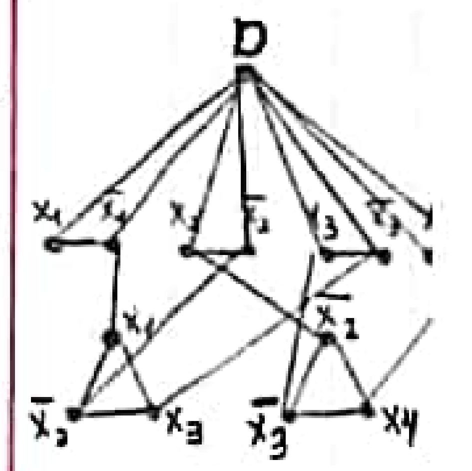
<u>$3\text{-}col \in NP$:</u> נבנה אלגוריתם וידוא שמקבל גרף וצביעה ב-3 צבעים, ונבדוק האם יש קשת בין כל הצמתים שהם מאותו צבע.

בס"ד

$3\text{-Col} \in NPH$: באמצעות רדוקציה מ-NAE-3-CNF-SAT, שהי[א] עם 3 ליטרלים בכל פסוקית, כך שבכל פסוקית יש לפחות א[חד] ... אך $NAE\text{-}3\text{-}CNF\text{-}SAT \in NPC$*. נגדיר פונקציה $CNF\text{-}SAT \to 3\text{-}col$ ... f תיצור גרף G_φ. בגרף זה יש צומת D (Dont care), ... ב-φ, ב-G_φ יהיו שני צמתים x_i ו-$\bar{x}_i$ וביניהם קשת, וכולם ... בנוסף, עבור כל פסוקית ב-φ נידור משולש בפני עצמו, ומכל צומ[ת] ... הנגדית לה בנגנדות.

NAE: Not All Eq[ual]

...י בתיאל או ...סלות.



$(x_1 \cup \bar{x}_2 \cup x_3) \cap (\bar{x}_2 \cup \bar{x}_3 \cup$ ...

(1) $\varphi \in NAE\text{-}3\text{-}CNF\text{-}SAT \Leftrightarrow G_\varphi \in 3\text{-}col$: כיוון 1 - נניח ש-$CNF\text{-}SAT$ ... [יש] השמה מספקת. נצבע את $V(G_\varphi)$ בשלושה צבעים 1,2,3 (true, false, dont care) בדרך ה... בכל הנגנדות כל צומת שהליטרל המקביל שלו מקבל שערך true בהש[מה] ... אין שהליטרל המקביל שלו מקבל false יצבע בצבע 2. על כל משול[ש] ... לצומת שהליטרל המקביל בהשמה a היא true נצבע צומת זו ב... בוודאי יש לפחות אחת כזו כיוון ש-φ הוא NAE. את הצומת ... שצביעה כי לא תקית הוא שיש קשת אמשלש לנגנדה בין 2 ... מסמלים ליטרלים עגודים לא יכול להיות כזה.

כיוון 2 - נניח שב-G_φ יש צביעה ב-3 צבעים. כיוון שבכ[ל] ה... חייבת להכיל את כל שלושת הצבעים ומאחר שכולם מחוברים ל-D, ... מכילה את הצבעים 1 ו-2. נגדיר השמה פשוטה ל-φ, אם x_i בנג[נדה] ... יקבל false. בכל משולש, יש את כל שלושת הצבעים וכל צומת מ[שולש] מחוברת לנגנדה שמספרה 2 ולכן היא בהכרח true בהשמה, לצומת 1 ולכן בהשמה היא false. כיוון שבכל פסוקית יש ש... ספיקה ו-NAE.

(2) לכל $\varphi \in NAE\text{-}3\text{-}CNF$ f תחשב את G_φ בזמן פולינומי!

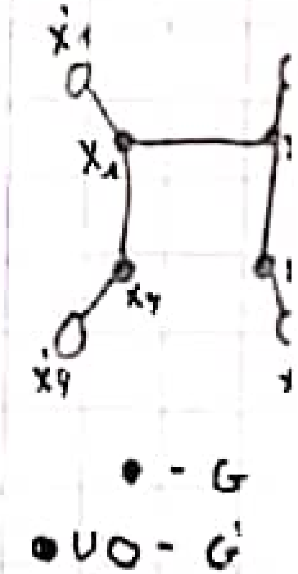
אלגוריתם לצביעת גרף 3-צביע ב-$\sqrt{n}$ צבעים

- כל עוד קיים $u \in V(G)$ עם לפחות $\sqrt{n}$ שכנים, צבע את u בצבע ח[דש] ...
- נפעיל את האלגוריתם הרגיל החמדני כדי לצבוע את הצמתים ...

הסבר: בשלב הראשון כל איטרציה נמחקים לפחות $\sqrt{n}$ צמתים, לכן לכל הדבשים - $3\sqrt{n}$. בשלב השני בגרף G' שנותר, $\sqrt{n} > \Delta(G)$, ולכן האלגוריתם ...

בס"ד

נגדיר פונקציה $F: HAM \to BST$ המקבלת גרף G ובונה גרף G' שבו מוסיפים לכל $x \in V(G)$ צומת נוספת x', כלומר מכפילים את כמות הצמתים ב-2, וכן מוסיפים קשת (x,x'). נבחר $k=3$.



<u>נוכיח: $G \in HAM \iff (G',k) \in BST$.</u>

כיוון 1: נניח שב-G יש מסלול המילטון, אזי אם נוסיף ב-G' לכל הקשתות במסלול את הקשתות (x,x') לכל $x \in V(G)$ נקבל עץ פורש שדרגתו לכל היותר 3.

כיוון 2: נניח שב-G' יש עץ פורש עם דרגה לכל היותר 3. אזי העץ כולל את כל הקשתות (x,x') לכל $x \in V(G)$. נוריד את כל הקשתות האלו, ונקבל עץ פורש ב-G עם דרגה לכל היותר 2. נשתמש במשפט האומר שכל עץ פורש עם דרגה 2 הוא מסלול המילטון. לכן מצאנו מסלול המילטון.