המכללה האקדמית להנדסה בכרמיאל
בראודה

המחלקה למתמטיקה שימושית

חקירת משחק האורות

מאת:
ולדיסלב ברקנס

מנחה:
אלכס גולוורד

6 ביוני 2022

תוכן העניינים

1 הקדמה

פרויקט זה הינו פרויקט סוף של סטודנט במחלקה למתמטיקה שימושית. הפרויקט עוסק בחקירה מתמטית של משחק האורות. המטרה המקורית של הפרויקט הייתה למצוא אלגוריתם למציאת פתרונות למשחק, אך במהלך העבודה העלנו שאלות מעניינות נוספות ומצאנו תשובות להן.

נציג שתי שיטות למציאת פתרון של המשחק, ונראה שהן בעצם מתבססות על אותו הרעיון.

במהלך הפרויקט עסקנו גם במציאת פתרונות אופטימליים. נציין שהגדרה של פתרון אופטימלי, לפי מה שידוע לנו, לא מופיע באף מאמר שמוקדש לחקר מתמטי של משחק האורות.

עבודה סוף זו הייתה מהנה עבורי. אני מודה למחלקה למתמטיקה שימושית, במיוחד לאלכס גולוורד על הזדמנות לעשות עבודה מרתקת. עבודה זאת לימדה אותי המון ונתנה לי את האומץ להשתמש בכלים שלמדתי במהלך התואר.

2 תיאור המשחק

משחק האורות, בלועזית Lights Out , הוא בעצם חידה. במקור היא התפרסמה כמשחק אלקטרוני על לוח משבצות מלבני. המשחק יצא לשוק בשם זה בשנת 1995 על לוח 5×5. קיימים משחקים דומים שהופיעו בשוק לפני כן, כמו Merlin , שהופיע בשנת 1970 עבור לוח 3×3. במשחק האורות כל משבצת יכולה להיות באחד משני מצבים, נקרא להם דלוק וכבוי. כאשר משתמשים בשמות האלו מתכוונים שבכל משבצת יש נורה והיא יכולה להיות דלוקה או כבויה. במצב התחלתי כל הנורות כבויות. יש לנו לוח בקרה שמאפשר בכל שלב של המשחק ללחוץ על משבצת ולשנות את מצב הנורה, אם היא דלוקה אז ניתן לכבות אותה ואם היא כבויה אז ניתן להדליק אותה. לוח הבקרה בנוי בצורה כזאת שכאשר מתבצעת לחיצה על משבצת אז מצבה של הנורה משתנה ויחד איתה משתנים גם מצבם של הנורות הסמוכות לה. שתי נורות נקראות סמוכות אם הן נמצאות במשבצות בעלות צלע משותפת. המטרה של המשחק היא לעבור ממצב התחלתי שבו כל הנרות כבויות, למצב בו כל הנורות יהיו דולקות.

קראנו לשני המצבים בהם יכולה להיות כל משבצת נורה דלוקה ונורה כבויה. כמובן שניתן לתת לשני מצבים שמות אחרים, כגון משבצת בצבע צהוב ומשבצת בצבע שחור.

2.1 תיאור גרפי של המשחק

באיור הבאה נגדיר: מצב התחלתי הוא מצב בו כל הנורות צהובות. מצב סופי הוא מצב בו כל הנורות שחורות. לחיצה על משבצת תסומן על ידי צביעת שפה שלה בירוק.

איור 2.1: הסבר שינוי מצב הלוח לאחר לחיצה

פירוט: באיור 2.1א מתואר מצב התחלתי. באיור 2.1ב ניתן לראות את התוצאה של לחיצה על משבצת שמסומנת בירוק. באיור 2.1ג ניתן לראות את התוצאה של לחיצה נוספת.

לשם הבנה מומלץ לנסות את המשחק, כפי שנאמר "עדיף לראות פעם אחת, מאשר לשמוע מאה פעמים" או במקרה שלנו פשוט לשחק. את המשְחק אפשר לשחק בקישור הזה .

2.2 סוגיות בהן נעסוק בפרויקט

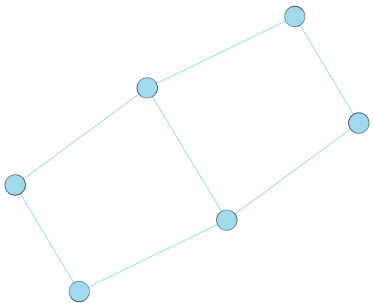
1. תיאור ודיון בשני אלגוריתמים למציאת פתרון המשחק.
2. הרחבה של משחק על לוח למשחק על גרף.
3. הוכחת קיום הפתרון לכל גרף.
4. מספר פתרונות עבור לוח $m \times n$. חסם עליון של המספר הזה.
5. חיפוש לוחות בהם קיים פתרון בו הנורות שינו את מצבם רק פעם אחת. נקרא לפתרונות האלה אופטימליים.

2.3 תיאור משחק על גרף

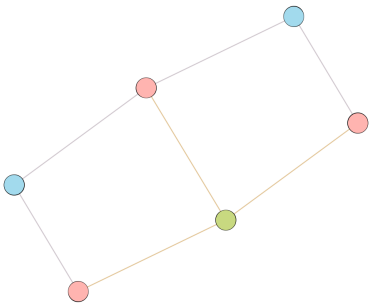
אחרי שתיארנו את המשחק על לוח, נתאר את המשחק על גרף. נזכיר שגרף זה מבנה המכיל קשתות וצמתים, קשתות מוגדרות כצירוף סדור של שני צמתים. כדי לתאר את משחק האורות על גרף נשתמש באותם כללים שהגדרנו. לחיצה על צומת משנה את מצבו ומצב שכניו. נגדיר שזוג צמתים יקראו שכנים אם קיימת קשת המחברת ביניהם. מטרת המשחק היא לעבור מגרף בו כל הצמתים נמצאים במצב אחד לגרף בו כל הצמתים נמצאים במצב שני.

איור 2.2: משחק על גרף לדוגמה

(א) מצב התחלתי



(ב) לחיצה על משבצת מסומנת



נמחיש זאת על דוגמה שבאיור 2.2. איור 2.2א מתאר את מצב התחלתי, נסמן את המצב התחלתי של צומת בצבע כחול. איור 2.2ב מתאר לחיצה על צומת שצבוע בירוק. הלחיצה הזאת שינתה את הצמתים השכנות למצבם הסופי שמסומן בצבע ורוד.

הערה 2.1: בפועל צומת ירוקה גם נצבעת בורוד. הצביעה בירוק נועדה רק לציין שנעשתה לחיצה על הצומת הזה.

בפרק 3.1 נראה קשר בין המשחק ולאחת משיטות של ייצוג גרפים , ייצוג בעזרת מטריצת שכנויות.

הגדרה 2.1: תהי גרף $G = (V, E)$, כאשר V קבוצת הצמתים ו - E קבוצת הקשתות של הגרף. נסמן ב - $|V| = n$ את מספר הצמתים. נמספר את הצמתים ב - $1, 2, \ldots, n$. הקשת שמחברת את הצמתים i, j נסמן ב - (i, j). נגדיר מטריצת שכנויות $A = (a_{i,j})$ כך:

$$a_{i,j} = \begin{cases} 1, & (i,j) \in E \\ 0, & (i,j) \notin E \\ 1, & i = j \end{cases}$$

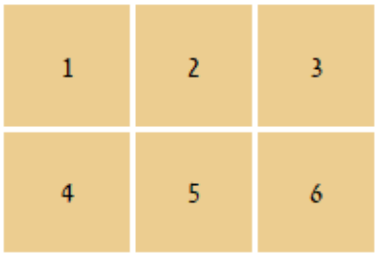
מטריצה A נקראת מטריצת שכנויות של הגרף.

משחק על כל לוח מלבני הוא בעצם משחק על גרף כאשר כל משבצת היא צומת וזוג של משבצות סמוכות הוא שני צמתים שמחוברים בקשת.

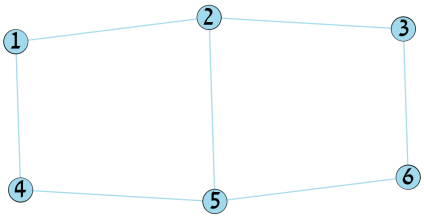
לדוגמה, ניקח לוח 2×3. נמספר את המשבצות כמו באיור 2.3א. הגרף המתקבל מתואר באיור 2.3ב.

איור 2.3: דוגמה למשחק על לוח שתורגם למשחק על גרף

(א) משחק על לוח 2×3 שמשבצותיו ממוספר



(ב) משחק על גרף שתורגם מלוח 2×3



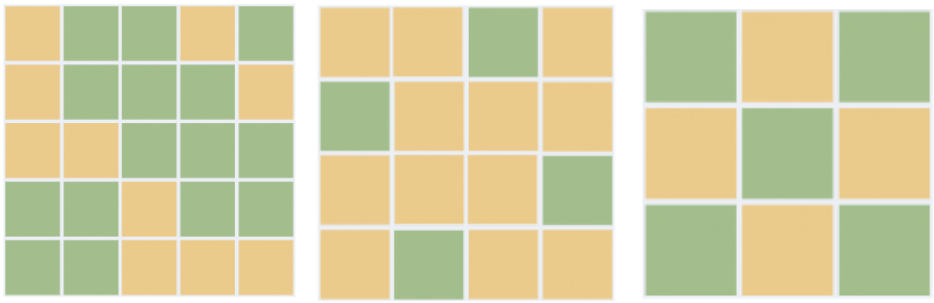
הערה 2.2: כל לוח הוא גרף אבל לא כל גרף הוא גרף של לוח. לדוגמה, גרף בו יש צומת אם יותר מ 4 שכנים לא ניתן לתאר כלוח מכיוון שלכל משבצת על לוח יש לכל היותר 4 משבצות סמוכות.

3 אלגוריתם למציאת פתרון

לפני שנציג את שיטות למציאת פתרון, נרצה להמחיש את האתגר במשחק על ידי הצגה מספר תופעות שמתקיימות במשחק. באיור 3.1 מוצגים מספר פתרונות אפשריים ללוחות שונים, לחיצה על הלחצנים הירוקים בסדר כלשהו תוביל לפתרון המשחק. ניתן לראות שמספר הלחיצות הנדרשות לפתרון לוח 4×4 קטן ממספר הלחיצות הנדרשות לפתרון לוח 3×3. אפשר היה לחשוב שככל שהלוח גדול יותר נדרשות יותר לחיצות כדי להגיע לפתרון, אך, ניתן לראות באיור 3.1 שזה לא נכון. תופעה נוספת המתקיימת במשחק היא שמספר הפתרונות

עבור לוחות שונים משתנה. עבור לוח 3×3 קיים פתרון יחיד, אולם ללוח 4×4 קיימים 16 פתרונות. באופן מפתיע, ללוח 5×5 קיימים רק 4 פתרונות. תופעה זאת מפתיעה משום שאפשר היה לצפות שככל שהלוח גדול יותר , כך מספר הפתרונות יגדל. על מנת לחדד תופעה זו, נסתכל על לוחות $n \times n$. נשאל מהו המימד של הלוח אם הכי הרבה פתרונות ומהו מספר פתרונות ללוח זה כאשר $n \in \{1, 2, \ldots, 20\}$. התוצאה המתקבלת היא שמספר הפתרונות הגדול ביותר הוא כאשר $n = 19$ ומספר הפתרונות הוא $65,536$. בנוסף $n = 19$ הוא הלוח היחיד ב - $\{1, 2, \ldots, 20\}$ המקבל את מספר פתרונות זה. מספר הפתרונות השני בגודלו הוא 256 והוא מתקבל עבור $n = 9, 16$.

איור 3.1 : פתרונות של משחק על לוחות שונים



שתי גישות למציאת פתרון שנציג בעבודה מבוססות על מידול הבעיה ע״י מערכת משוואות שפתרונה יוביל לפתרון המשחק.

3.1 אלגוריתם שמבוסס על מטריצת שכנויות

משום שמשחק על לוח מלבני הוא מקרה פרטי של משחק על גרף , נעסוק במידול מתמטי של משחק על גרף. לחיצה על צומת משנה את מצב הצומת ומצב שכנותיו. נסמן את הצמתים ב - i. נתאר את המשחק בצורה אלגברית:

1. כל צומת יכול להיות בשני מצבים, את המצבים נסמן: $\{0,1\}$.
2. מצב של צומת i נסמן ב - n_i.
3. בתחילת המשחק מצבו של כל צומת הוא $n_i = 0$, $i = 1, \ldots, n$.
4. משחק מסתיים כאשר מצבם של כל הצמתים הוא $n_i = 1$, $i = 1, \ldots, n$.

הערה 3.1: פעולת לחיצה על לחצן משנה את מצב הנורה, שינוי מצב הזה ניתן לתאר בעזרת חיבור בשדה $\mathbb{Z}_2$. נורה שמצבה הוא n_i לאחר לחיצה תעבור למצב $n_i + 1$.

למה 3.1: נניח שלוח נמצא במצב X . לחיצה על משבצת i ואחר כך על משבצת j מעבירות את הלוח ממצב X למצב Y. לחיצה על משבצת j ואחר כך על משבצת i מעבירות את הלוח ממצב X למצב Z . נוכיח ש $Y = Z$.

הוכחה. אם למשבצות i, j אין שכנים משותפים אז מובן ש - $Y = Z$. נניח ש - k משבצת שכנה ל - i ול - j . לכן מצב של k משתנה פעמיים, גם כאשר לוחצים קודם על i ואחר כך על j וגם כאשר לוחצים קודם על j . □

מסקנה 3.1: התכונה הזאת מאפשרת למדל את סדרת הלחיצות על ידי חיבור ב - $\mathbb{Z}_2$ כי הוכחנו שהרכבה של שתי לחיצות היא פעולה קומוטטיבית וחיבור היא פעולה קומוטטיבית.

הערה 3.2: מספר זוגי של לחיצות על אותו צומת יחידה אינו משנה את מצב הלוח.

הוכחה. כאשר מספר הלחיצות הוא זוגי מספר השינויים של צומת ושל שכנותיו הוא זוגי כלומר, מצבן לא ישתנה בכלל אחרי שתי לחיצות על אותו צומת. □

ממה שכתבנו נובע שפתרון המשחק הוא סדרה לחיצות על צמתים מסוימים בגרף , כאשר סדר הלחיצות לא משנה. עבור גרף עם k צמתים ניתן לבנות 2^k סדרות שונות של לחיצות. לוח משבצות $m \times n$ הוא גרף עם $m \cdot n$ צמתים ולכן יש 2^{mn} סדרות שונות של לחיצות עבור לוח $m \times n$.

כדי להבין כמה גדול $2^{m \cdot n}$ נסתכל על לוח 6×6. כמות האפשרויות ללחיצה גדולה מכמות המספרים שמציגים מספרים שלמים במחשב (4 בתים). המספר הגדול ביותר שאפשר להציג בעזרת 4 בתים הוא $2^{32} - 1$. המטרה של המחשה זו היא להדגיש כמה לא פרקטי לנסות למצוא פתרון למשחק בעזרת מעבר על כל האפשרויות.

נראה עכשיו איך לתרגם תהליך חיפוש פתרון של משחק לחיפוש פתרון של מערכת משוואות לינאריות מעל שדה $\mathbb{Z}_2$. ניקח לדוגמה משחק על לוח 2×2. נייצג את הלוח הזה בעזרת מטריצה $a_{i,j} = \mathbb{Z}_2$, $\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}$.

המטריצה שמייצגת את הלוח במצב התחלתי של משחק היא $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$. לחיצה משבצת $a_{1,1}$ נתאר כך

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \xrightarrow{a_{1,1}} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \tag{1}$$

לחיצה על משבצת $a_{1,2}$ כאשר הלוח נמצא במצב $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ מתוארת כך

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \xrightarrow{a_{1,2}} \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \tag{2}$$

נשים לב שמעבר 1 ניתן להציג כחיבור מטריצות

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

ומעבר 2 כחיבור מטירצות

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$$

מתקבלת מסקנה שלחיצה על משבצת ניתן לייצג כחיבור מטירצות. מטריצה ראשונה היא מטריצה שמייצגת מצב הלוח לפני לחיצה. מטריצה שנייה מייצגת את הלחיצה עצמה והיא בנויה באופן הבא. אם נלחצת משבצת $a_{i,j}$ אז במטריצה של לחיצה הזאת יש לכתוב 1 באיבר הזה וגם באיברים השכנים לאיבר הזה לפי שורה ועמודה.
עבור לוח 2×2 יש 4 מטריצות ללחיצות האפשריות

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

אז בעית מציאת משבצות עליהן יש ללחוץ על מנת להעביר לוח ממצב $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ למצב $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ ניתן לנסח בעזרת חיבור מטריצות כך

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + x_1 \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} + x_2 \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} + x_3 \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} + x_4 \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{3}$$

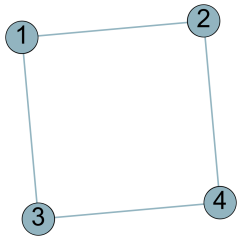
הנעלמים x_1, x_2, x_3, x_4 שייכים לשדה $\mathbb{Z}_2$. הטענה שהוכחנו קודם שסדר הלחיצות לא משנה את התוצאה מוצאת בו ביטוי בכך שפעולה חיבור של מטריצות היא פעולה קומטטיבית. נרשום עכשיו שוויון 3 בצורה וקטורית. לשם כך נכתוב וקטור קואורינטות של כל מטריצה בשוויון הזה בבסיס סטנדרטי של מרחב וקטורי של מטריצות $\mathbb{Z}_2^{2\times 2}$.

$$\begin{bmatrix}0\\0\\0\\0\end{bmatrix} + x_1\begin{bmatrix}1\\1\\1\\0\end{bmatrix} + x_2\begin{bmatrix}1\\1\\0\\1\end{bmatrix} + x_3\begin{bmatrix}1\\0\\1\\1\end{bmatrix} + x_4\begin{bmatrix}0\\1\\1\\1\end{bmatrix} = \begin{bmatrix}1\\1\\1\\1\end{bmatrix} \tag{4}$$

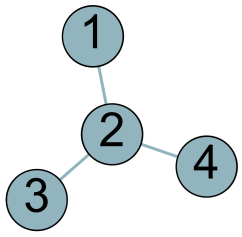
שוויון 4 הוא מערכת משוואות $\vec{0} + A\vec{x} = \vec{1}$ או $A\vec{x} = \vec{1}$ כאשר

$$A = \begin{bmatrix}1&1&1&0\\1&1&0&1\\1&0&1&1\\0&1&1&1\end{bmatrix}, \vec{x} = \begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix}, \vec{1} = \begin{bmatrix}1\\1\\1\\1\end{bmatrix}$$

מטריצה A היא מטריצת שכנויות של גרף שמייצג לוח 2×2:



לכן תרגום בעיית מציאת פתרון של משחק אורות על לוח לבעיית מציאת פתרון של מערכת משוואות לינאריות מעל שדה $\mathbb{Z}_2$ תקף עבור משחק על גרף. נדגים עבור גרף הבאה.



מטריצת שכנויות שלו היא

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

אז על מנת למצוא פתרון למשחק אורות על הגרף הזה יש לפתור מערכת $A\vec{x} = \vec{1}$

הגדרה 3.1: שיטת פתרון הנעזרת ביצירת מערכת משוואות ומציאת וקטור פתרון תקראה אלגוריתם מבוסס מטריצת שכנויות.

לפי מה שידוע לנו השיטה שקראנו לה בפרויקט שיטת מטריצת השכנויות לראשונה הופיעה במאמר של K. Sutner [2]. נציין שבמאמר של K. Sutner לא רק הוצגה שיטת זו אלה גם ניתנה הוכחה לקיום פתרון. מרגע שהצלחנו לתאר את הבעיה בצורה משוואות לינארית על שדה $\mathbb{Z}_2$, נוכל להיעזר בכלים של אלגברה לינארית כדי למצוא פתרון, כמו מציאת פתרון בעזרת דירוג או מציאת מטריצה פסאודו הפוכה וכולי.

הערה 3.3: עבור משחק על לוח $m \times n$ גודל מערכת המשוואות המתקבל משיטה מבוססת מטריצת שכנויות הוא $m \cdot n$ משתנים ומשוואות.

עבור לוח $[m \times n]$ מטריצת השכנות בגודל $[m \cdot n \times m \cdot n]$. השאלה שנרצה לענות בפרק הבאה היא: האם קיימות שיטות לפתור את המשחק, בעזרת מערכת המשוואות יותר מצומצמת?

3.2 אלגוריתם שמבוסס על מילוי עקבי של שורות

בפרק 3.1 תיארנו איך מתקבלת מערכת משוואות לינאריות מעל שדה $\mathbb{Z}_2$. מטריצה של מערכת הזאת היא מטריצת שכנויות. עכשיו נראה איך אפשר להגיע לאותה מערכת משוואות על בסיס של שיקולים שונים .נראה גם שבמקום לפתור את המערכת הזאת ניתן לפתור מערכת הרבה יותר קטנה. השיטה שנציג מבוססת על המאמר [1]. לגישה החדשה נקרא "אלגוריתם שמבוסס על מילוי עקבי של שרות" או בקצרה מילוי עקבי.

ללא מגבלת הכלליות נסביר את הגישה החדשה עבור לוח משבצות 3×3. נשייך לכל משבצת משתנה באופן הבא.

x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9

כל משתנה מסמן האם משבצת שלו נלחצה או לא נלחצה כאשר עברנו ממצב הלוח בו כל המשבצות כבויות למצב בו כל המשבצות דלוקות. אז $x_i = 1$ כאשר משבצת הזאת נלחצה ו - $x_i = 0$ כאשר משבצת הזאת לא נלחצה. נציין שמשבצת: מסוימת תהיה דלוקה אחרי סדרת לחיצות אם סכום לחיצות עליה ושכנות שלה הוא מספר

אי-זוגי. כאשר ננסח את התנאי הזה עבור כל משבצת נקבל מערכת משוואות מעל שדה $\mathbb{Z}_2$. ממה שכתבנו קודם נובע שמטריצת מקדמים שלה היא מטריצת שכנויות. מערכת משוואות עבור לוח 3×3 ניתן לראות באיור 5.

נדגים עבור משבצת עם משתנה x_1. על מנת שהיא תהיה דלוקה אחרי סדרת לחיצות סכום לחיצות עליה ועל שכנותיה חייב להיות אי-זוגי, כלומר, בשדה $\mathbb{Z}_2$ מתקיים השוויון

$$x_1 + x_2 + x_4 = 1$$

עבור משבצת x_2 מתקבל השוויון

$$x_1 + x_2 + x_3 + x_5 = 1$$

עכשיו נראה איך ניתן במקום מערכת עם מטריצת שכנויות לבנות מערכת הרבה יותר קטנה.

לשם כך נחלץ את x_4 ממשוואה ראשונה:

$$x_4 = 1 + x_1 + x_2$$

נחלץ x_5 ממשוואה שנייה:

$$x_5 = 1 + x_1 + x_2 + x_3$$

נחלץ את x_6 ממשוואה שלישית ונקבל:

$$x_6 = 1 + x_2 + x_3$$

נחלץ את x_7 ממשוואה רביעית, כאשר נעזר בערכי המשתנים שכבר חולצו:

$$x_1 + x_4 + x_5 + x_7 = 1 \Rightarrow x_1 + (1 + x_1 + x_2) + (1 + x_1 + x_2 + x_3) + x_7 = 1$$

לכן מתקבל:

$$x_7 = 1 + x_1 + x_3$$

נחלץ את x_8 ממשוואה חמישית:

$$x_8 = 0$$

נחלץ את x_9 ממשוואה שישית: נקבל:

$$x_9 = 1 + x_1 + x_3$$

נציב ביטויים שקיבלנו עבור המשתנים $x_4, \ldots, x_9$ בשלושת המשוואות האחרונות של מערכת משוואות ונקבל מערכת:

$$\begin{cases} x_2 + x_3 = 1 \\ x_1 + x_2 + x_3 = 0 \\ x_1 + x_2 = 1 \end{cases}$$

למערכת המשוואות יש פתרון יחיד והוא $x_1 = 1, x_2 = 0, x_3 = 1$.
מפה נובע ש $x_4 = 0, x_5 = 1, x_6 = 0, x_7 = 1, x_8 = 0, x_9 = 1$.

איור 3.2 : מערכת משוואות עבור לוח 3×3

$$\left[\begin{array}{ccccccccc|c} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right] \quad (5)$$

נבחין שכל הפעולות שנעשו בגישה מילוי עקבי ניתן לבצע ע״י פעולות אלמנטריות של דירוג במערכת 5. לכן מימוש השיטה בקוד נעשה ע״י שימוש בפעולות אלמנטריות של גאוס אבל סדר של פעולות האלה שונה מסדר של פעולות אלמנטריות בשיטת הדירוג של גאוס. סדר פעולות היה לפי סדר חילוץ המשתנים שהצגנו בגישה מילוי עקבי. נציג את סדר פעולות הדירוג על אותה דוגמה, על לוח 3×3. נציין שפעולות הדירוג כמובן מתבצעות על מטריצת המורחבת כלומר מטריצת השכנויות עם וקטור הלוח הפתור $\vec{1}$.

כאשר תיארנו את האלגוריתם מילוי עקבי עבור שלושת המשוואות הראשונות רק חילצנו את המשתנים ולכן עבור שלושת השורות הראשונות במטריצה לא נבצע אף פעולה אלמנטרית.

עבור משוואה רביעית הצבנו את המשתנים המחולצים של x_4, x_5, לכן עבור משוואה רביעית ננסה להיפטר ממשתנים עלו בעזרת פעולות שורות :

$$r_4 \leftarrow r_4 + r_1$$

$$r_4 \leftarrow r_4 + r_2$$

עבור משוואה חמישית הצבנו את המשתנים המחולצים של x_4, x_5, x_6, לכן עבור משוואה חמישית ננסה להיפטר ממשתנים עלו בעזרת פעולות שורות :

$$r_5 \leftarrow r_5 + r_1$$

$$r_5 \leftarrow r_5 + r_2$$

$$r_5 \leftarrow r_5 + r_3$$

עבור משוואה שישית הצבנו את המשתנים המחולצים של x_5, x_6, לכן עבור משוואה שישית ננסה להיפטר ממשתנים עלו בעזרת פעולות שורות:

$$r_6 \leftarrow r_6 + r_2$$

$$r_6 \leftarrow r_6 + r_3$$

עבור משוואה שביעית הצבנו את המשתנים המחולצים של x_7, x_8, לכן עבור משוואה שביעית ננסה להיפטר ממשתנים עלו בעזרת פעולות שורות:

$$r_7 \leftarrow r_7 + r_4$$

$$r_7 \leftarrow r_7 + r_5$$

עבור משוואה שמינית הצבנו את המשתנים המחולצים של x_7, x_8, x_9, לכן עבור משוואה שמינית ננסה להיפטר ממשתנים עלו בעזרת פעולות שורות:

$$r_8 \leftarrow r_8 + r_4$$

$$r_8 \leftarrow r_8 + r_5$$

$$r_8 \leftarrow r_8 + r_6$$

עבור משוואה תשיעית הצבנו את המשתנים המחולצים של x_7, x_8, לכן עבור משוואה תשיעית ננסה להיפטר ממשתנים עלו בעזרת פעולות שורות:

$$r_9 \leftarrow r_9 + r_5$$

$$r_9 \leftarrow r_9 + r_6$$

לאחר פעולות דירוג מערכת המשוואות המתקבלת בשורות שבע שמונה ותשעה היא:

$$x_1 + x_2 + x_3 = 1$$

$$x_2 + x_3 = 0$$

$$x_1 + x_2 = 1$$

4 קיום פתרון ומספר הפתרונות עבור משחק על גרף

בפרק 4.1 נוכיח שעבור משחק על כל גרף קיים לפחות פתרון אחד. לראשונה הוכחת קיום הפתרון לכל גרף ניתנה במאמר [3]. מעניין לציין שעד היום קיימת בעצם רק הוכחה אחת שמבוססת על כלים של אלגברה לינארית למרות שמדובר בעצם על תכונה מסוימת של גרף.

4.1 הוכחת קיום פתרון עבור משחק על גרף

הגדרה 4.1: לכל שני וקטורים $\vec{x},\vec{y}\in\mathbb{Z}_2{}^n$ נגדיר פעולה הבאה:

$$\vec{x}\cdot\vec{y}=\begin{bmatrix}x_1\\x_2\\\dots\\x_n\end{bmatrix}\cdot\begin{bmatrix}y_1\\y_2\\\dots\\y_n\end{bmatrix}=x_1y_1+x_2y_2+\dots x_ny_n=\vec{x}^T\vec{y} \tag{6}$$

כאשר $\vec{x}^T=[x_1,x_2,\dots,x_n]$ ו- $\vec{x}^T\vec{y}$ היא מכפלת מטריצות. לפעולה הזאת בין שני וקטורים ב $\mathbb{Z}_2{}^n$ נקרא מכפלה סקלרית.

הערה 4.1: פעולה שהגדרנו בהגדרה 4.1 נקראת מכפלה סקלרית למרות שהיא מקיימת רק 3 מתוך 4 תכונות של מכפלה סקלרית ב- R^n. תכונה $\vec{u}\cdot\vec{u}=0\Leftrightarrow\vec{u}=\vec{0}$ לא מתקיימת.

למשל,

$$\begin{bmatrix}1\\1\end{bmatrix}\cdot\begin{bmatrix}1\\1\end{bmatrix}=1+1=0$$

הערה 4.2: וקטורים $\vec{x},\vec{y}\in\mathbb{Z}_2{}^n$ יקראו מאונכים אחד לשני אם, תוצאת מכפלה הסקלרית ביניהם שווה ל 0 כלומר, $\vec{x}\cdot\vec{y}=0$. וקטורים מאונכים זה לזה יסומנו ב- $\vec{x}\perp\vec{y}$.

משפט 4.1: תהי מטריצה $A\in\mathbb{Z}_2{}^{m\times n}$. אז מתקיים $\mathrm{Col}A\perp\mathrm{Nul}A^T$ ו $\mathrm{Col}A^T\perp\mathrm{Nul}A$

הוכחה. תהי מטריצה $A\in\mathbb{Z}_2^{m\times n}$ ניקח $\vec{x}\in\mathrm{Nul}A$. לכן $A\vec{x}=\vec{0}$. אז,

$$\vec{x}\perp\mathrm{Row}A=\mathrm{Col}A^T\Rightarrow\mathrm{Nul}A\perp\mathrm{Col}A^T$$

נציב במקום A את A^T ונקבל ש- $\mathrm{Nul}A^T\perp\mathrm{Col}A$ □

הערה 4.3: עבור המכפלה הסקלרית שהגדרנו על השדה הוקטורי $\mathbb{Z}_2{}^n$ לא לכל מטריצה A מתקיים:

$$\mathrm{Col}A\cap\mathrm{Nul}A^T=\{\vec{0}\}$$

דוגמה 4.1: המחשב להערה 4.3:

עבור מטריצה:

$$A=\begin{bmatrix}1&1\\1&1\end{bmatrix}$$

מתקיים:

$$\begin{bmatrix}1\\1\end{bmatrix} \in \mathrm{Col}A \cap \mathrm{Nul}A^T$$

משפט 4.2: למשחק על כל גרף קיים פתרון.

הוכחה. יהי $G=(V,E)$ גרף עם n צמתים, $|V|=n$. נסמן ב- $A \in \mathbb{Z}_2{}^{n}$ מטריצת שכנויות שלו. נזכיר לקורא ש- A מטריצה סימטרית וכל האיברים באלכסון הראשי שלה שווים 1.

כדי להראות שלמשחק יש פתרון צריך להראות שקיים פתרון למערכת

$$A\vec{x}=\vec{1}$$

אם A מטריצה הפיכה אז קיים פתרון והוא יחיד. נניח עכשיו ש- A לא הפיכה, כלומר $\mathrm{Nul}A \neq \{\vec{0}\}$. ניקח $\vec{x} \in \mathrm{Nul}A$ אז מתקיים $A\vec{x}=\vec{0}$ לכן

$$\vec{x}^T A\vec{x} = \vec{x}^T\vec{0} = 0$$

$$\begin{aligned}\vec{x}^T A\vec{x} = a_{1,1}x_1^2 + 2(a_{1,2}+a_{2,1})x_1x_2 + \cdots + 2(a_{1,n}+a_{n,1})x_1x_n + \\ + a_{2,2}x_2^2 + 2(a_{2,3}+a_{3,2})x_2x_3 + \cdots + 2(a_{2,n}+a_{n,2})x_2x_n + \cdots\end{aligned} \tag{7}$$

היות ומטריצה סימטריות $a_{i,j}=a_{j,i}$ לכן מתקבל:

$$a_{i,j} - a_{j,i} = a_{i,j} + a_{j,i} = 0$$

לכן את המשוואה 7 אפשר לפשט כך:

$$\vec{x}^T A\vec{x} = a_{1,1}x_1^2 + a_{2,2}x_2^2 + \ldots + a_{n,n}x_n^2$$

היות ומתקיים $x_i^2 = x_i$ כי $x_i \in \mathbb{Z}_2$ לכן, ניתן לפשט את משוואה 7:

$$\vec{x}^T A\vec{x} = a_{1,1}x_1 + a_{2,2}x_2 + \ldots + a_{n,n}x_n = 0 \tag{8}$$

$a_{1,1}=a_{2,2}=\ldots=a_{n,n}=1$ לכן משוויון 8 נובע ש $\vec{x} \perp \vec{1}$ כאשר $\vec{x} \in \mathrm{Nul}A$. לפי משפט 4.1 $\vec{1} \in \mathrm{Col}A^T$. מטריצה A סימטרית, לכן $A^T=A$ ומתקיים $\vec{1} \in \mathrm{Col}A$. מפה נובע שלמערכת $A\vec{x}=\vec{1}$ יש פתרון. □

4.2 קיום פתרון עבור משחק עם מצב התחלתי וסופי כלליים

במשחק המקורי על גרף כל הצמתים שלו כבוים, נמצאים במצב 0 והמטרה היא להגיע למצב סופי בו כל הצמתים דלוקים, נמצאים במצב 1. הוכחנו שלמשחק הזה יש פתרון לכל גרף. נניח עכשיו שבמצב התחלתי חלק מצמתי הגרף כבוים וחלק מהם דלוקים. נסמן את המצב הזה כווקטור $\vec{S}_0$. אם לגרף יש n צמתים אז $\vec{S}_0 \in \mathbb{Z}_2{}^n$. נשנה גם את מטרת המשחק והיא תהיה להגיע ממצב $\vec{S}_0$ למצב בו גם חלק מצמתי הגרף יהיה כבוי וחלק דלוק. נסמן את המצב הזה בווקטור $\vec{S}_e \in \mathbb{Z}_2{}^n$. אם $x_1, x_2, \ldots, x_n$ היא סדרת לחיצות שמעבירה את הגרף ממצב $\vec{S}_0$ למצב $\vec{S}_e$ אז מתקיים השוויון $\vec{S}_0 + A\vec{x} = \vec{S}_e$ כאשר A היא מטריצת שכנויות של הגרף. אז למשחק עליו אנחנו מדברים בפרק הזה יהיה פתרון אם ורק אם למערכת משוואות $\vec{S}_0 + A\vec{x} = \vec{S}_e$ יהיה פתרון. מדיון בפרק הקודם נובע שלמשחק יהיה פתרון אם ורק אם $\vec{S}_e - \vec{S}_0 \perp \mathrm{Nul}A$ נדגים את המסקנה עבור לוח 1×2 כאשר

$$\vec{S}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \ , \ \vec{S}_e = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

בלי לנסח מערכת משוואות ובירור האם יש לה פתרון, מובן שלמשחק הזה אין פתרון כי לחיצה על משבצת שמאלית או על משבצת ימינית מעבירה את הלוח ממצב $\vec{S}_0$ למצב $\vec{S} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ ולחיצה על משבצת שמאלית או ימינית בלוח שנמצא במצב $\vec{S}$ מעבירה אותו למצב $\vec{S}_0$. ננסח עכשיו מערכת משוואות $A\vec{x} = \vec{S}_e - \vec{S}_0$. מתקבלת מערכת

$$\left[\begin{array}{cc|c} 1 & 1 & 1 \\ 1 & 1 & 0 \end{array}\right]$$

ואחרי דירוג

$$\left[\begin{array}{cc|c} 1 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right]$$

ולכן אין לה פתרון. כמו כן, $\mathrm{Nul}A = \mathrm{Span}\left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}$ ווקטור $\vec{S}_e - \vec{S}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ לא אורתוגנלי ל - $\mathrm{Nul}A$.

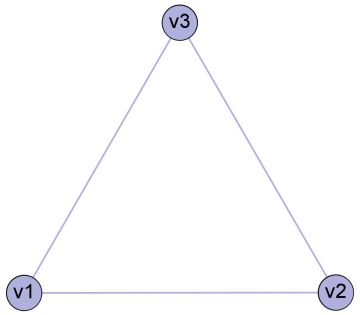
4.3 מספר הפתרונות

בפרק הזה נחזור למשחק המקורי כאשר בהתחלה כל הצמתים במצב 0 והמטרה היא ע"י לחיצות על צמתים מסוימים להגיע למצב בו כל הצמתים במצב 1. למערכת משוואות $A\vec{x} = \vec{1}$ כאשר A מטריצת שכנויות של גרף קיים לפחות פתרון אחד. אם A מטריצה לא הפיכה אז למערכת יש יותר מפתרון אחד. למשל, עבור לוח 1×2 יש 2 פתרונות, ללחוץ על משבצת שמאלית או על משבצת ימינית. בפרק הזה נעסוק בספירת מספר הפתרונות.

על מנת להבין איך לספור את מספר הפתרונות נתחיל מדוגמה פשוטה שתסביר את הרעיון.

דוגמה 4.2: נדון במשחק על גרף המתואר באיור 4.1.

איור 4.1: משחק על גרף



קל לראות שלמשחק עבור גרף הזה יש 4 פתרונות והם 4 קבוצות של לחיצות $\{v_1\},\{v_2\},\{v_3\},\{v_1,v_2,v_3\}$. בקבוצת לחיצות $\{v_1,v_2,v_3\}$ ניתן ללחוץ בכל סדר שנרצה, הרי כבר הסברנו שזה לא משנה את התוצאה. נספור עכשיו את מספר הפתרונות בעזרת מערכת משוואות $A\vec{x} = \vec{1}$. מטריצת שכנויות של הגרף היא $A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$. המערכת $A\vec{x} = \vec{1}$ לא הומוגנית. פתרון כללי שלה שווה לסכום פתרון כללי של מערכת הומגנית $A\vec{x} = \vec{0}$ ופתרון פרטי של מערכת הלא הומגנית. קל לראות שהווקטור $\vec{x}_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ הוא פתרון המערכת $A\vec{x} = \vec{1}$ כי $A\vec{x}_0 = \vec{1}$. פתרון כללי של מערכת של מערכת הומוגנית מתקבל כך

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \ , \ \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$x_1 + x_2 + x_3 = 0, x_2 = s, x_3 = t, x_1 = s + t \Rightarrow \ \vec{x} = \begin{bmatrix} s+t \\ s \\ t \end{bmatrix} = s \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + t \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

לכן פתרו כללי של מערכת $A\vec{x} = \vec{1}$ הוא $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + s \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + t \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$ ולכן הנוסחה של פתרון כללי מכילה 4 פתרונות.

נעבור לדיון כללי. למערכת $A\vec{x} = \vec{1}$ יש לפחות פתרון אחד. פתרון כללי שלה שווה לסכום פתרון פרטי שלה, נקרא לו וקטור $\vec{p}_0$ ופתרון כללי של מערכת הומגנית $A\vec{x} = \vec{0}$. מספר וקטורי הבסיס של מערכת $A\vec{x} = 0$ שווה ל $\dim \mathrm{Nul}A$. $\dim \mathrm{Nul}A = n - \mathrm{rank}A$ כאשר n מספר צמתי הגרף (אז A מטריצה $n \times n$). נסמן $k = \dim \mathrm{Nul}A$.

$$\mathrm{Nul}A = \mathrm{Span}\{\vec{p}_1, \ldots, \vec{p}_k\}$$

לכן פתרון כללי של מערכת $A\vec{x} = \vec{1}$ הוא

$$\vec{p}_0 + \alpha_1\vec{p}_1 + \ldots \vec{p}_k$$

כאשר $i = 1, \ldots, k$, $\alpha_i \in \mathbb{Z}_2$. מפה נובע שקבוצת הפתרונות של מערכת $A\vec{x} = \vec{1}$ מורכבת מ - 2^k וקטורים ולכן למשחק על גרף יש 2^k פתרונות.

נסכם. למשחק על גרף עם n צמתים כאשר A מטריצת שכנויות שלו יש $2^{n-\mathrm{rank}A}$ פתרונות.

הנוסחה שקיבלנו דורשת לדעת את הדרגה של מטריצת שכנויות. מסתבר שעבור משחק על לוח משבצות בגודל $m \times n$ ניתן לקבל אומדן של מספר הפתרונות בלי לדעת דרגה של מטריצת שכנויות. לפי שיטה של מילוי עקבי במקום לפתור מערכת $A\vec{x} = \vec{1}$ כאשר A מטריצת שכנויות ניתן, לפתור מערכת משוואות קטנה יותר רק עם $\min\{m, n\}$ נעלמים ומשוואות.

את ההסבר לכך נכתוב עבור לוח, למשל 5×3.

x_1	x_2	x_3
x_4	x_5	x_6
$\vdots$	$\vdots$	$\vdots$
x_{13}	x_{14}	x_{15}

לפי שיטה של מילוי עקבי נבטא את המשתנים $x_4, x_5, \ldots, x_{15}$ דרך x_1, x_2, x_3. עבור משבצות שמסומנות ב - x_{13}, x_{14}, x_{15} נכתוב 3 משוואות והן יהיו עם 3 נעלמים x_1, x_2, x_3. אנחנו יודעים שלמערכת הזאת יש פתרון. נסמן את מטריצת המקדמים שלה ב - B. לפי אותם שיקולים כמו עבור מטריצה A מספר פתרונות של מערכת עם מטריצה B הוא $2^{n-\mathrm{rank}B}$. נשים לב ששיטת מילוי עקבי ניתן להפעיל עבור עמודות. כלומר, דרך משתנים של עמודה ראשונה אפשר לבטא את שאר המשתנים ואחר כך לכתוב מערכת משוואות לפי משתנים של עמודה אחרונה אז נקבל מערכת 13×13. נסמן את מטריצת המקדמים של מערכת המשוואות ב- C ומספר פתרונות של אותה מערכת יהיה $2^{m-\mathrm{rank}C}$. מפה מתקבלת מסקנה הבאה.

מסקנה 4.1: מספר פתרונות עבור משחק על לוח $m \times n$ קטן או שווה ל - $2^{\min\{m,n\}}$.

לסיכום נציג באיור 4.2 את כמות הפתרונות שיש בלוח. השורות והעמודות בטבלה מייצגות את ממדי הלוח.

איור 4.2 : טבלה מתארת מספר פתרונות בלוחות $m \times n$

	1	2	3	4	5	6	7	8	9
1	1	2	1	1	2	1	1	2	1
2	2	1	4	1	2	1	4	1	2
3	1	4	1	1	8	1	1	4	1
4	1	1	1	16	1	1	1	1	16
5	2	2	8	1	4	1	16	2	2
6	1	1	1	1	1	1	1	64	1
7	1	4	1	1	16	1	1	4	1
8	2	1	4	1	2	64	4	1	2
9	1	2	1	16	2	1	1	2	256

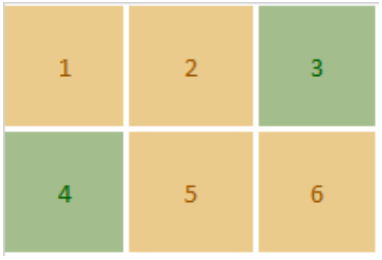
5 פתרון אופטימלי עבור לוחות מלבניים

בפרק הזה נציג סוג מסוים של פתרונות, שנקרא לו פתרונות אופטימליים. הפתרונות האלה קל לחפש באופן ויזואלי.

הגדרה 5.1: פתרון אופטימלי של משחק הינו פתרון בו כל נורה משנה את מצבה רק פעם אחד. כלומר השחקן פתר את המשחק כאשר כל הנורות עברו ממצב התחלתי למצב הסופי פעם אחת בלבד.

באיור 5.1 ניתן לראות דוגמא לפתרון מינמלי עבור משחק על לוח 2×3. כשלוחצים על לחצנים $3, 4$ כל הנורות נדלקות, ואף אחת מהם לא נכבית באף לחיצה.

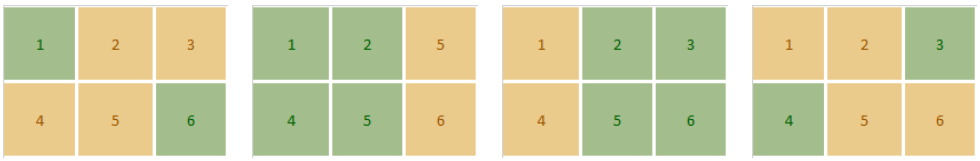
איור 5.1 : פתרון מינמלי של משחק



נרצה לחדד ולהדגיש עד כמה קל למצוא פתרונות אופטימליים. אם ניקח לוח 2×3 כפי שמתואר באיור 5.1, ונתבונן במספר כל הפתרונות שיש ללוח זה, כפי שמתואר בטבלה 4.2, נראה שיש 4 פתרונות. שני פתרונות אופטימליים ושני פתרונות לא אופטימליים, נזמין את הקורא לחפש את כל הפתרונות. כנראה ששני הפתרונות

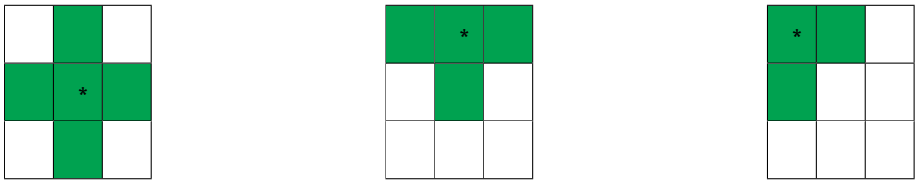
האופטימליים נמצאו מיידית. כנראה שכדי למצוא את הפתרונות הנותרים נצטרך לקחת דף ועט, ולחפש אותם גם עבור לוח בממד מצומצם שכזה. את כל ארבעת הפתרונות נציג באיור 5.2.

איור 5.2 : משחק על גרף לדוגמה



נשים לב שחיפוש פתרונות אופטימליים מזכיר משחקים כמו פאזלים או טטריס. הדימיון בין המשחקים נובע מכך שהמטרה במשחקים אלו היא לרצף איזור מסויים במספר צורות שונות. משחק האורות בו הפתרונות המתקבלים הם רק פתרונות אופטימליים, נהיה למשחק שמנסים למלא את הלוח באמצעות לבנים בצורות שונות. משחק שכזה זה הוא מקרה פרטי של אוסף משחקי ריצוף ע״י אבני פוליאומינו. פוליאומינו הוא אובייקט קומבינטורי המורכב מריבועים המחוברים זה לזה, על ידי הצמדת צלעות הריבועים. אז חיפוש פתרונות אופטימליים על לוח משבצות שקול לריצוף הלוח הזה ע״י אוסף מסוים של פוליאומינו. האוסף הזה עבור לוח $m \times n$ כאשר $m, n > 2$ מכיל 3 פוליאומינו שניתן לראות בציור הבא.

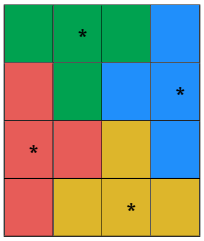
פוליאומינו פנימי | פוליאמינו עבור שפה של לוח | פוליאומינו פינתי



כאשר המשבצות שנלחצו מסומנות בסימן $*$.

קיום פתרון אופטימלי עבור משחק על לוח שקול לשאלה הבאה. האם אפשר לרצף את הלוח עם פוליאומינו של המשחק.

דוגמה לריצוף משחק 4×4

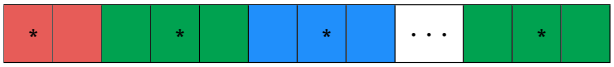


בפרק הזה נעסוק בסוגיה הבאה. לאיזה לוחות קיים פתרון אופטימלי כאשר מצב התחלתי הוא שכל הנורות במצב 0.

משפט 5.1: קיים פתרון אופטימלי לכל לוח $1 \times m$

הוכחה. נפריד בין שלושה מקרים. עבור $n = 3k$ הריצוף הוא

עבור $n = 3k + 2$ הריצוף הוא



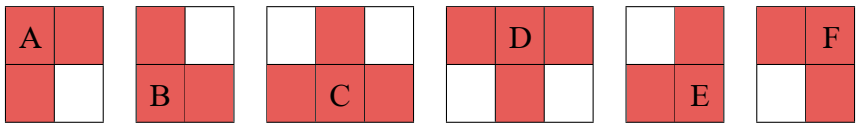
כי $3k + 2 - 2 = 3k$.

עבור $n = 3k + 1$ הריצוף הוא

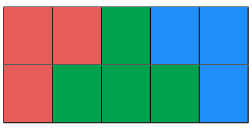
כי $3k + 1 - 4 = 3(k - 1)$. □

משפט 5.2: קיים פתרון אופטימלי עבור לוח $2 \times m$ אם ורק אם m הוא מספר אי זוגי.

הוכחה. נוכיח שעבור לוח $2 \times m$ יש פתרון אופטימלי אם ורק אם m הוא מספר אי זוגי. נניח ש- $m > 1$ כי אם $m = 1$ מתקבל לוח בעל עמודה אחד ועל קיום פתרון אופטימלי עבור לוח הזה דיברנו קודם כאשר דנו בלוח $1 \times m$. עבור לוח $2 \times m$ כאשר $m > 1$ קיימים שישה סוגי פוליאומינו הבאים.



הריצוף



ניתן להציג ע״י מילה ACF. המילה שמציגה ריצוף כלשהו חייבת להתחיל מ-A או מ-B ולהסתיים ב-E או ב-F. מספיק לדון במילים שמתחילות מאות A כי מילים המתחילות מאות B מציגות ריצופים שמתקבלים מריצופים שמוצגים ע״י מילים שמתחילות מאות A ע״י שיקוף בישר אופקי בין שתי שורות של הלוח. יש רק שני מקרים. רצף CD מופיע p פעמים בריצוף או שאות C מופיעה בריצוף p פעמים ואות D $(p - 1)$ פעמים. אם רצף אותיות

CD מופיע בריצוף p פעמים, אז המילה שמציגה את הריצוף הזה מסתיימת באות E. נחשב מספר משבצות בשורה שנייה של הלוח עבור מקרה הזה.

$$1 + 3p + p + 2 = 3 + 4p = m$$

לכן m הוא מספר אי זוגי. אם אות C מופיעה בריצוף p פעמים ואות D $p - 1$ פעמים אז המילה שמציגה את הריצוף הזה מסתיימת באות F. מספר משבצות בשורה ראשונה של הלוח הזה שווה

$$2 + p + 3(p - 1) + 2 = 1 + 4p = m$$

ולכן m מספר אי זוגי. □

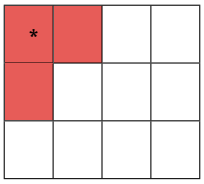
כדי להוכיח טענת הכותרת של תת הפרק הזה, נעזר בלוח מלבני שרירותי.

משפט 5.3: עבור לוח בעל מימד כלשהו גדול מ 3, אין פתרון אופטימלי בו המשבצת הפינתית נלחצה.

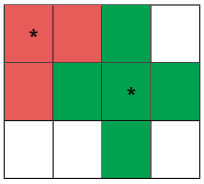
הוכחה. בהוכחה נסמן את המשבצות באופן הבא:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \dots \\ a_{2,1} & a_{2,2} & \dots \\ \dots & \dots & \dots \end{bmatrix}$$

לאחר לחיצה על $a_{1,1}$ מתקבל לוח הבא:



כדי להדליק את הנורה $a_{2,2}$ נצטרך ללחוץ על $a_{2,3}$ או $a_{3,2}$. אם נילחץ על $a_{2,3}$ נקבל את הלוח הבאה:



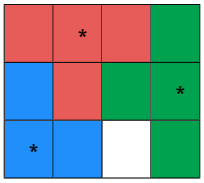
נשים לב שהגענו למבוי סתום משום שניתן להדליק את $a_{3,2}$ רק על ידי לחיצה על $a_{4,2}$, לאחר מכן לא היה ניתן להדליק את $a_{3,1}$. ניתוח דומה אפשר לתאר עבור לחיצה על $a_{3,2}$. □

לפי טענה 5.3, אם ברצוננו למצוא פתרון אופטימלי ללוח בעל מימד כלשהו גדול מ 3 לא נוכל להדליק את נורה $a_{1,1}$ על ידי לחיצה עליו. לכן על מנת להדליק את $a_{1,1}$ נהיה חייבים ללחוץ על $a_{1,2}$ או $a_{2,1}$. בשלב זה נראה שגם

לחיצה על $a_{1,2}$ לא תוביל למציאת פתרון אופטימלי. היות והלוח סימטרי גם לחיצה על $a_{2,1}$ לא תוביל למציאת פתרון אופטימלי.

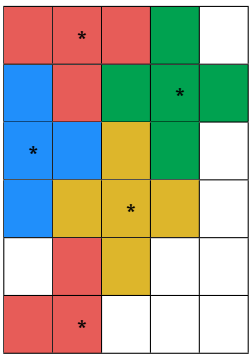
אם לחצנו על משבצת $a_{1,2}$ אז על מנת להדליק את $a_{2,1}$ בלי ללחוץ עליו חייבים ללחוץ על $a_{3,1}$. אז על מנת להדליק את $a_{2,3}$ בלי ללחוץ עליו נלחץ על $a_{2,4}$.

מתקבל לוח הבא:



זה כבר מוכיח שלכל לוח $3 \times n$ כאשר $n > 2$ אין פתרון.

אחרי שלחצנו על $a_{2,4}$ על מנת להדליק $a_{3,3}$ בלי ללחוץ עליה חייבים ללחוץ על $a_{4,3}$. אם מדובר על לוח 4×4 זה נותן פתרון אופטימלי עבורו כפי שניתן לראות באיור 5.4. אם לא, אז על מנת להדליק $a_{5,2}$ בלי ללחוץ עליה חייבים ללחוץ על $a_{6,2}$ וזה מביא למובי סתום, כפי שניתן לראות בלוח הבאה:



כך הוכחנו טענה הבאה.

משפט 5.4: עבור לוח $m \times n$ כאשר $\min\{m,n\} > 2$ קיים פתרון אופטימלי אם ורק אם $m = n = 4$.

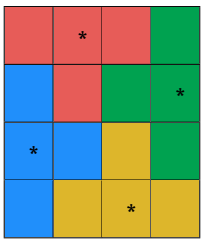
אומנם כבר מצאנו עבור אילו לוחות קיים פתרון אופטימלי וגם מצאנו את הפתרונות עצמם אבל נציין פה שניתן לחפש פתרונות אופטימליים בעזרת כלים של אלגברה לינארית. לשם כך ננסח מערכת משוואות שפתרון אופטימלי אמור לקיים. נדגים עבור לוח 2×3

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \end{bmatrix}$$

$$\begin{cases} x_1 + x_2 \quad + x_4 = 1 \\ x_1 + x_2 + x_3 \quad + x_5 = 1 \\ \quad x_2 + x_3 \quad + x_6 = 1 \\ x_1 \quad + x_4 + x_5 = 1 \\ \quad x_2 \quad + x_4 + x_5 + x_6 = 1 \\ \quad x_3 \quad + x_5 + x_6 = 1 \end{cases}$$

בגלל שכל נורה נלחצת רק פעם אחת $x_i \in \{0,1\}$, $i = 1,\ldots,6$ ומשום שמדובר על ריצוף, פעולת חיבור פה היא חיבור רגיל ולא חיבור של שדה $\mathbb{Z}_2$. אז מדובר על מציאת פתרון של מערכת משוואות שרכיבים שלו רק 0 או 1. בנספחים מופיע קוד שממש את הרעיון הזה.

איור 5.4 : פתרון ללוח 4×4



6 נספחים

מימוש של הפרויקט בוצע על ידי שפת Python בעזרת הספריות Sage ו numpy.

6.1 יצירת מטריצת שכנויות

קוד זה יוצר מטריצת שכנויות של משחק על לוח $m \times n$.

[8]:
```
import numpy as np
def genenerate_neighbord_matrix_m_n(m,n) -> np.array:
    mat = np.zeros((m*n, m*n), dtype= np.int8)

    # the general case
    for j in range(0, m*n):
        if j-n > -1 :
            mat[j-n,j] = 1

        if j % n != 0 :
            mat[j-1,j] = 1

        mat[j,j] = 1

        if (j+1) % n != 0 :
            mat[j+1,j] = 1

        if j+n < m*n :
            mat[j+n,j] = 1

    return mat
def genenerate_neighbord_matrix(n) -> np.array:
    return genenerate_neighbord_matrix_m_n(n,n)
```

```
print('Adj matrix for 3,2 board:')
print(genenerate_neighbord_matrix_m_n(3,2))
```

```
Adj matrix for 3,2 board:
[[1 1 1 0 0 0]
 [1 1 0 1 0 0]
 [1 0 1 1 1 0]
 [0 1 1 1 0 1]
 [0 0 1 0 1 1]
 [0 0 0 1 1 1]]
```

6.2 אלגוריתם שמבוסס על מטריצת השכנויות

קוד זה מוצא פתרון לפי אלגוריתם שמבוסס על ממטריצת שכנויות. הפתרון המתקבל הוא וקטור שורה שאינדקסים של וקטור הם אינדקסים של משבצות לפי שיטת המספור שהצגנו בפרויקט. ניקח לדוגמה את הפתרון עבור לוח 3×3 שהתקבל בפלט:

$$(1,0,1,0,1,0,1,0,1)$$

את אותו פתרון נתאר בעזרת מטריצה כך:

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

[7]:
```
from sage.all import *
n = 3
A = Matrix(Integers(2),genenerate_neighbord_matrix(n)) # A = adjacency␣
 ↪matrix
Y = vector([1 for x in range(n**2)]) # Y = ( 1, 1, ..., 1)
X = A.solve_right(Y)
print('Solution for 3x3 board:')
print(X)
```

```
Solution for 3x3 board:
(1, 0, 1, 0, 1, 0, 1, 0, 1)
```

6.3 אלגוריתם מבוסס על מילוי עקבי של שורות

קוד זה מוצא פתרון לפי אלגוריתם שמבוסס על מילוי עקבי של שורות. הקוד מחולק לשלוש פונקציות:

gaussian_elimination_spanish_alg:

הפונקציה מקבלת כקלט: מטריצת שכנויות של המשחק ווקטור לוח במצבו הסופי כלומר, לוח בו כל הנורות דולקות. באלגוריתם הנ"ל נעשה שימוש בפעולות אלמנטריות על שורות אבל סדר פעולות שונה מסדר שלהן מסדר שלהן בדירוג של גאוס. המטרה שלנו היא לבטא את כל נעלמי המשבצות על ידי נעלמים של משבצות של שורה הראשונה. התוצאה המתקבלת מהפונקציה היא מטריצת שכנויות עם משתנים מחולצים, מצב הלוח הסופי לאחר פעולות על שורות.

mul_mat_sol_based_on_res:

הפונקציה מקבלת כקלט: מטריצת שכנויות עם משתנים מחולצים, מצב הלוח הסופי לאחר פעולות על שורות ופתרון חלקי עבור משבצות בשורה הראשונה בלוח. פונקציה זו לוקחת את הפתרונות של המשבצות בשורה הראשונה של הלוח ומוצאת לשאר המשבצת בלוח את מצבן בפתרון, לחוץ או לא. התוצאה המתקבלת היא שהפתרון החלקי שקיבלנו בקלט התמלאה והפתרון מתייחס לכל המשבצות על הלוח.

generate_mat_spanish_alg:

הפונקציה מקבלת כקלט מטריצת שכנויות. פונקציה זו משתמש בשי הפונקציות הקודמות כדי להחזיר את הפתרון של המשחק בעזרת שיטת מילוי עקבי. הפתרון מוצג כוקטור שורה כפי שתיארנו ב 6.2.

[3]:
```python
def gaussian_elimination_spanish_alg(mat : np.array, sol_vec : np.array):
    n = int(sqrt(mat.shape[0]))
    #all rows but the last one
    for i in range(0, n**2-n):
        # the lamp that is affected
        affected_lamp = i + n
        row_i = mat[i][:affected_lamp+1]
        # check rows below
        # for j in range(i+1, n**2):
```

```
        for j in [i-1 + n, i+n, i+n+1, i+ 2*n]:
            if j> -1 and j < n**2 and mat[j][affected_lamp] == 1:
                row_j = mat[j][:affected_lamp+1]
                row_j = row_j + row_i
                row_j = row_j % 2
                mat[j][:affected_lamp+1] = row_j
                sol_vec[j] = ( sol_vec[j] + sol_vec[i] ) % 2

# get result to [n, n**2-1] from solution [0, n-1]
def mul_mat_sol_based_on_res(mat : np.array, end_state : list, res :␣
 ↪list):
    n = int(sqrt(mat.shape[0]))
    for i in range(0,n**2-n):
        res_i_plus_n = int(end_state[i])
        for j in range(0,i+n):
            res_i_plus_n = (res_i_plus_n + mat[i][j] * res[j]) % 2
        res.append(res_i_plus_n)

# facade for the intire spanish method
def generate_mat_spanish_alg(mat : np.array):
    n = int(sqrt(mat.shape[0]))
    end_state = np.ones(n**2) # end_state = (1, 1, ... , 1)
    gaussian_elimination_spanish_alg(mat, end_state)
    # the matrix we need to solve for parmeter [0, n-1]
    new_mat = np.array(mat[n**2-n:n**2, 0:n], copy=True)
    # the solution vector after row operation
    new_sol = np.array(end_state[n**2-n:n**2],  copy=True)

    # find solution for n variables
    A = Matrix(Integers(2),new_mat)
    Y = vector(Integers(2),new_sol)
```

```
    X = A.solve_right(Y)
    res = [x for x in X] # solution for parmeter [0, n-1]
    mul_mat_sol_based_on_res(mat, end_state, res)
    return res

mat = genenerate_neighbord_matrix(4)
A = Matrix(Integers(2),mat)
res = generate_mat_spanish_alg(mat)
print('solution for board n=4:')
print(res)

print('check solution by multiply matrix with soultion vector:')
X = vector(Integers(2),res)
Y = A*X
print(Y)
```

```
solution for board n=4:
[0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1]
check solution by multiply matrix with soultion vector:
(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
```

6.4 השווה בין שתי האלגוריתמים

קוד זה דוגם זמני ריצה של שני האלגוריתמים ומחזיר כפלט טבלה זמני ריצה.

[4]:
```
import datetime
import numpy as np

def matrix_solve(mat):
    A = Matrix(Integers(2),mat)
    Y = vector([1 for x in range(n**2)])
    Z = vector([0 for x in range(n**2)])
    X = A.solve_right(Y)
    return X

val = []
# run on range(10 ,61,5)
for i,n in enumerate(range(10 ,15)):
    # print(i)
    mat = genenerate_neighbord_matrix(n)

    a0 = datetime.datetime.now()
    matrix_solve(mat)
    b0 = datetime.datetime.now()
    c0 = b0 - a0
    t0 = c0.total_seconds()
    # print(t0)

    a1 = datetime.datetime.now()
    generate_mat_spanish_alg(mat)
    b1 = datetime.datetime.now()
    c1 = b1 - a1
    t1 = c1.total_seconds()
```

```
    # print(t1)

    val.append((n, t0, t1))

res = np.array(val)
# np.savetxt("benchmark.csv", res, delimiter = ',')
print('board size, adj method, row by row method')
print(res)
```

```
board size, adj method, row by row method
[[10.         0.029358   0.319221]
 [11.         0.042352   0.406416]
 [12.         0.051597   0.548713]
 [13.         0.064825   0.781002]
 [14.         0.101306   1.072234]]
```

6.5 מציאת פתרון אופטימלי

קוד זה מוצא פתרון אופטימלי. מציאת הפתרון מבוססת על גישה של מציאת פתרון במערכת משוואות על שלמים.

[5]:
```
from sage.all import *
n = 3
m = 2
a = genenerate_neighbord_matrix_m_n(m,n)
A = Matrix(ZZ,a)
Y = vector([1 for x in range(m*n)])
Z = vector([0 for x in range(m*n)])
X = A.solve_right(Y)
print('Optimal solution:')
print(X)
```

```
Optimal solution:
(0, 0, 1, 1, 0, 0)
```

6.6 מספר הפתרונות על לוח

קוד זה מחשב מספר הפתרונות שיש על לוחות $m \times n$ כאשר $m, n \leq 9$. הפלט שמתקבל הוא טבלה, שהשורות והעמודות מתארות את מימדי הלוח. לדוגמה אפשר לראות מטבלת התוצאות שללוח 3×5 מספר הפתרונות הוא 8 כפי שמתואר בשורה 3 ובעמודה 5.

[6]:
```
def num_solution_board(m,n):
    a = genenerate_neighbord_matrix_m_n(m, n)
    A = Matrix(Integers(2),a)
    num_solutions = 2**A.kernel().dimension()
    return num_solutions


m = 9
n = 9
res = np.zeros((m, n), dtype= np.int32)
for i in range(1,m+1):
    for j in range(1,n+1):
        res[i-1][j-1] = num_solution_board(i,j)
print('Number solution based on m x n board size:')
print(res)
```

```
Number solution based on m x n board size:
```

	1	2	3	4	5	6	7	8	9
1	1	2	1	1	2	1	1	2	1
2	2	1	4	1	2	1	4	1	2
3	1	4	1	1	8	1	1	4	1
4	1	1	1	16	1	1	1	1	16
5	2	2	8	1	4	1	16	2	2
6	1	1	1	1	1	1	1	64	1
7	1	4	1	1	16	1	1	4	1
8	2	1	4	1	2	64	4	1	2
9	1	2	1	16	2	1	1	2	256

6.7 כל הפתרונות עבור לוח נתון

קוד זה מחשב את כל הפתרונות עבור לוח בגודל $m \times n$. הפתרון שמתקבל הוא רשימה של פתרונות כאשר כל פתרון הוא וקטור שורה כפי שתאירנו ב 6.2. מציאת כל הפתרונות מסתמכת על הגישה של חיבור פתרון פרטי עם כל הוקטורים במרחב האפס.

[81]:
```
def get_all_sol(m,n):
    """
    helper function to recursivly sum all combinations for sol_vector + null_vector
    """
    def get_all_sol_rec(cur_sol,index_in_null_base):
        if len(null_base) == index_in_null_base:
            all_sol.append(cur_sol)
            return
        get_all_sol_rec(cur_sol + null_base[index_in_null_base], index_in_null_base+1)
        get_all_sol_rec(cur_sol, index_in_null_base+1)

    # generates all structer that the helper function needs
    a = genenerate_neighbord_matrix_m_n(m,n)
```

```
    A = Matrix(Integers(2),a)
    Y = vector([1 for x in range(m*n)]) # Y = ( 1, 1, ..., 1)
    X = A.solve_right(Y)

    null_base = A.right_kernel_matrix().rows()
    all_sol = []
    get_all_sol_rec(X,0)
    return all_sol

m = 2
n = 3
res = get_all_sol(m,n)
print('All solution(each solution is row vector) based on m x n board␣
 ↪size:')
print(*res, sep = '\n')
print(f'number of soultion generated: {len(res)}')
```

```
All solution(each solution is row vector) based on m x n board size:
(1, 1, 0, 1, 1, 0)
(1, 0, 0, 0, 0, 1)
(0, 1, 1, 0, 1, 1)
(0, 0, 1, 1, 0, 0)
number of soultion generated: 4
```

מקורות

[1] Rafael Losada, Translated from Spanish by Ángeles Vallejo, *All lights and light out*, SUMA magazine's

[2] Jamie Mulholland, *Permutation Puzzles, Lecture 24: Light out Puzzle* , SFU faculty of science department of mathematic

[3] K. Sutner, *Linear Cellular Automata and the Garden-of-Eden*, The Mathematical Intelligencer, Vol. 11, No. 29, 1989, Springer-Verlag, New York.