אורט בראודה
ORT Braude College

המחלקה למתמטיקה שימושית

חקירת משחק האורות

מאת:

ולדיסלב ברקנס

מנחה:

אלכס גולוורד

30 באפריל 2022

תוכן העניינים

1 הקדמה

פרויקט זה הינו פרויקט סוף של סטודנט במחלקה למתמטיקה שימושית. הפרויקט חוקר את משחק האורות המטרה המקורית של הפרויקט הייתה למצוא פתרונות למשחק, אך במהלך המחקר העלנו שאלות נוספות.

נציג שתי שיטות למציאת פתרון של המשחק, בתחילה השיטות נראו שונות אבל בפרויקט ראינו דמיון ביניהם.

דבר מרכזי נוסף שעסקנו בו הוא בחיפוש פתרונות אופטימליים, מהו פתרון אופטימלי נגדיר בהגדרה 5.1 , פתרונות אילו הם מועטים וכן הוכחנו את כולם.

במהלך הפרויקט ראינו שבמשחק המתחיל כאשר כל הנורות דלוקות קיים לפחות פתרון אחד, תופעה זו העסיקה רבות את הפרויקט ובסיומו מצאנו הוכחה לקיום התופעה.

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

2 תיאור של המשחק

משחק האורות, בלועזית Lights Out , מתקיים על לוח משבצות מלבני. 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, בנוסף 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.

הערה 2.1 : מצבם התחלתי של נורות המשחק, דלוקות או כבויות, אינה משנה את תוצאות המשחק.

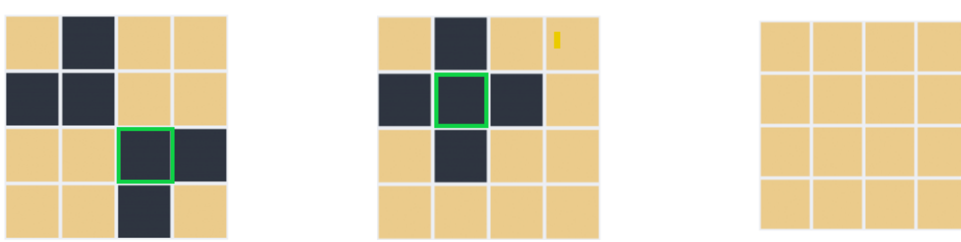
מטרתה של ההערה להדגיש כי מטרתו של המשחק היא להעביר את הלוח ממצב אחד בו נמצאים כל הנורות למצב האחר, וכי אין השפעה למראה של מצבים.

2.1 תיאור גרפי של המשחק

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

איור 2.1 : הסבר שינוי מצב הלוח לאחר לחיצה

(א) לוח במצב התחלתי (ב) לוח לאחר לחיצה בודדת (ג) לוח לאחר שני לחיצות



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

לשם הבנה מומלץ לנסות את המשחק, כפי שנאמר "עדיף לראות פעם אחת, מאשר לשמוע מאה פעמים" או במקרה שלנו לשחק. את המשחק אפשר לשחק בקישור הבאה:
https://www.geogebra.org/m/JexnDJpt#chapter/301822.

האתגר במשחק הוא שאין אסטרטגיה גלויה לכן, במשחקים רבים מנסים להגיע למצבים שפתרון כבר ידוע.

2.2 סוגיות בהן נעסוק בפרויקט

1. תיאור ודיון בשני אלגוריתמים למציאת פתרון המשחק.
2. הוכחה לקיום פתרון המשחק לכל לוח $m \times n$.
3. הרחבה של משחק על לוח למשחק על גרף.
4. נעסוק במספר הפתרון האפשריים בלוח ונדבר על חסם מספר הפתרונות האפשריים.
5. חיפוש לוחות בהם קיים פתרון בו הנורות שינו את מצבם פעם אחת.
6. נציג שיטה למציאת פתרונות בהם כל נורה תשתנה את מצבה פעם אחת.

בנוסף, קיימות שאלות רבות הקשורות למשחק ובפרויקט ננסה להציג פתרון לחלקן. יתרה מזאת, נרצה להציג תופעות מעניינות, ולהראות שהמשחק אינו רק מהנה אלא גם מהווה אתגר מתמטי לא קטן.

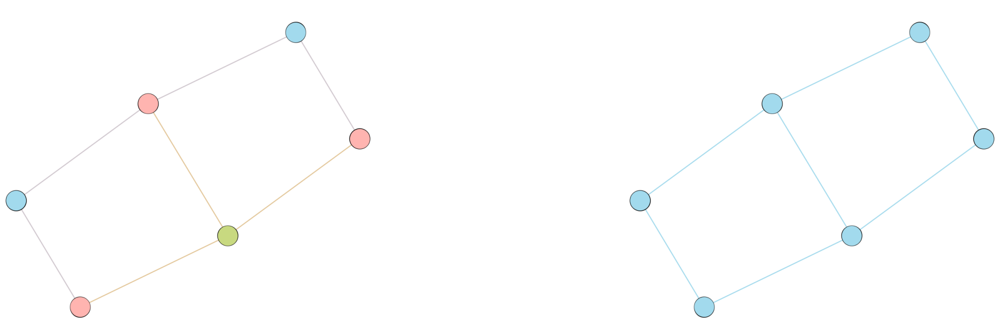
2.3 תיאור משחק על גרף

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. במשחק על גרף הצמתים הם המשבצות לכן, לחיצה על צומת הופכת את מצבה ומצב שכניה. נגדיר שזוג צמתים יקראו שכנים אם קיימת קשת המחברת ביניהם. מטרת המשחק לעבור מגרף שכל הצמתים במצב התחלתי למצב סופי.

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. לחיצה זה שינתה את הצמתים השכנות למצבם הסופי

שמסומן בצבע אדום.

הערה 2.2: בפועל צומת ירוקה גם נצבעת באדום הצביעה לירוק נועדה להצגה.

2.4 השוואה בין משחק על לוח למשחק על גרף

נרצה להראות כי משחק על לוח הוא סוג של משחק על גרף כלומר, כל משחק על לוח ניתן לתאר בעזרת משחק על גרף.

נתאר משחק על לוח כמשחק על גרף בעזרת הכללים הבאים:

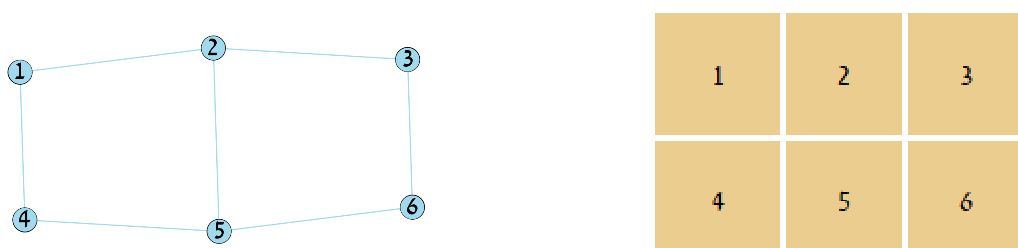
1. כל משבצת במשחק על לוח נהפוך לצומת.
2. כל זוג משבצות סמוכות על לוח נחבר בקשת בגרף.

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 2.3: קיימים משחקים רבים שניתן לתאר על גרף אך, לא ניתן לתאר אותם על לוח. לדוגמה, גרף בו יש צומת אם יותר מ 4 שכנים לא ניתן לתאר על לוח מכיוון שלכל משבצת על לוח יש לכל היותר 4 משבצות סמוכות.

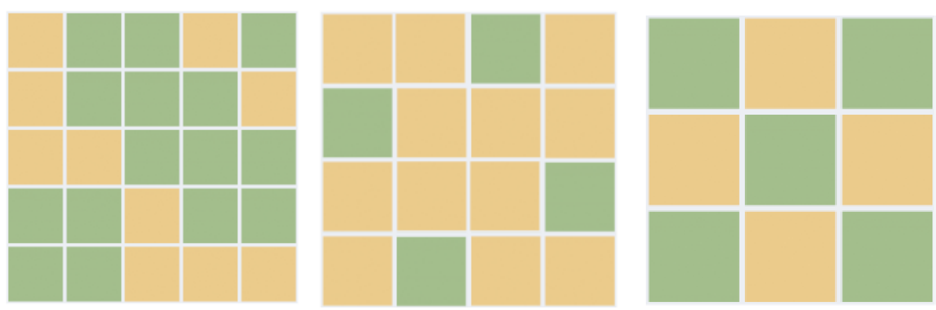
הערה 2.4: בעזרת השיטה שתיארנו אפשר לתאר כל משחק על לוח כמשחק על גרף, אבל ההפך הוא לא נכון. כלומר, לא כל משחק על גרף אפשר לתאר כמשחק על לוח.

מכיוון שמשחק על לוח ניתן לתאר כמשחק על גרף לכן, טענות שמתקיימות במשחק על גרף נכונות במשחק על לוח.

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 זה לא נכון. תופעה נוספת המתקיימת במשחק היא שכמות 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. תופעה זה מפתיע משום שאפשר היה לצפות שככל שהלוח גדול יותר כך, מספר הפתרונות יגדל. על מנת לחדד תופעה זו, נסתכל על לוחות ריבועים ($n \times n$). נשאל מהו המימד של הלוח אם הכי הרבה פתרונות ומהו מספר פתרונות ללוח זה כאשר $n \in [1,20]$? התוצאה המתקבלת היא שמספר הפתרונות הגדול ביותר הוא כאשר $n = 19$ ומספר הפתרונות הוא $65,536$. בנוסף $n = 19$ הוא הלוח היחיד ב $n \in [1,20]$ המקבל את מספר פתרונות זה. לאומת זאת מספר הפתרונות השני בגודלו הוא 256 ומתקיים בעבור $n \in \{9,16\}$.

איור 3.1 : פתרונות של משחק על לוחות שונים



שתי גישות למציאת פתרון שנציג בעבודה מבוססות על מידול הבעיה לשדה לינארי ולמערכת משוואות שפתרונה יוביל לפתרון המשחק. נתאר את השיטות אומנם, בתחילה הן נראות שונות אך, נציג את הקשר ביניהן.

3.1 אלגוריתם שמבוסס על מטריצת שכנויות

כדי למדל את הבעיה על שדה לינארי נשתמש בייצוג גרפי, לחיצה על צומת משנה את מצב הצומת ומצב שכנותיה. נסמן את הצמתים ב i. נתאר את המשחק בצורה אלגברית:

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

הערה 3.1: עבור משחקים על לוח נתאר את המשבצות ומצבם הנוכחים ב $a_{i,j}$. זאת מכיוון שמשחק על לוח ניתן לתאר בעזרת מטריצה.

הערה 3.2: פעולת לחיצה על לחצן משנה את מצב המנורה, שינוי מצב מנורה ניתן לתאר בעזרת חיבור בשדה $\mathbb{Z}_2$. נורה שמצבה הוא n_i לאחר לחיצה תעבור למצב $n_i + 1$.

למה 3.1: סדר הלחיצות אינו משנה את מצב הלוח.

הוכחה. לפי הערה 3.2 אפשר לתאר לחיצה כחיבור בשדה $\mathbb{Z}_2$, כלומר כל לחיצה משנה את מצבה ומצבן של המשבצות הסמוכות. לכן עבור סדרת לחיצות כלשהי מצב המשבצת תלוי אך ורק במספר השינויים שבוצעו עליה בסדרת הלחיצות ואינו תלוי בסדר שלהן. □

הערה 3.3: מספר זוגי של לחיצות על צומת יחידה אינו משנה את מצב הלוח.

הוכחה. כאשר מספר הלחיצות הוא זוגי מספר השינויים של משבצת ושל שכנותיה הוא זוגי כלומר, מצבן לא ישתנה. □

מסקנה 3.1: בעבור משבצת מסוימת שנלחצה m פעמים. מצב הלוח היה זהה אם למצב בו הינו לוחצים על אותה משבצת m מודולו 2 פעמים.

מסקנה 3.1 ממחישה שכדי לתאר פתרון של משחק מספיק לתאר רק את המשבצות שנלחצו.

למה 3.2: כמות האפשרויות לחיצה על לוח $m \times n$ הוא $2^{m \cdot n}$

הוכחה. לפי הערה 3.1 כל לחצן יכול להיות בשתי מצבים והיות לפי למה 3.1 סדר הלחיצות לא משנה, לכן ללוח $m \times n$ מספר האפשרויות ללחיצה $2^{m \cdot n}$. □

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 לפתור בעזרת מעבר על כל האופציות.

נתאר את המשחק בצורה וקטורית (בעזרת הערה 3.2). ניקח לדוגמה משחק בגודל 2×2, נתאר את הלוח במצבו התחלתי כמטריצה

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

נתאר לחיצה על משבצת $(1,1)$:

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

כפי שתיארנו בהערה 3.2 אפשר לתאר שינוי מצב הנורה על ידי חיבור מצבה עם אחד.

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

אם נציג כל מטריצה ע״י וקטור קואורדינטות בבסיס סטנדרטי של מרחב מטריצות אז, נוכל לרשום את השוויון הנ״ל גם כך :

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

וקטור המתאר הלחיצה על משבצת מסוימת יקרא וקטור שינוי שלה.

הגדרה 3.1 : תהי משחק על גרף בעל n צמתים הממוספרים מ 1 עד n, וקטור שינוי t_i של צומת i הוא :

- וקטור השייך $\mathbb{Z}_2{}^n$.
- וקטור שתוצאת החיבור עם וקטור לוח הוא וקטור לוח לאחר לחיצה על i.

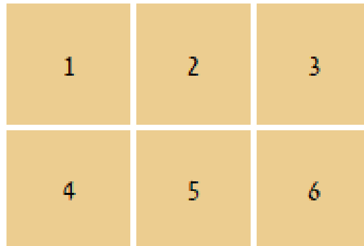
כדי לבנות וקטור שינו t_i :

- ערכי הוקטור שיקבלו ערך 1 היו באינדקסים של הצומת ושכנותיה
- שאר הערכי וקטור היו 0.

הגדרה 3.2 : וקטור המתאר את מצב הלוח יקרא וקטור הלוח.

הערה 3.4 : שיטת המספור בפרויקט היא מעבר על שורות ואז על עמודות כמו שמתואר באיור 3.2.

איור 3.2 : שיטת מספור משבצות על לוח



דוגמה 3.1: תהי גרף בעל 4 צמתים, כפי שמתואר באיור 3.3. צמתים שמצבם 1 יצבעו באדום, בכחול יצבעו צמתים שמצבם 0. נדגים צירוף לחיצות במשחק בעזרת וקטורי השינויים ווקטור הלוח.

באיור 3.3א מתואר גרף במצבו הנוכחי. נראה את שינוי הגרף לאחר לחיצה על הצמתים $1,3$.

וקטור שינוי של צומת 1:

$$t_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

נשרשר את וקטורי השינוי של שתי הלחיצות:

$$t_1 + t_3 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

אם נחבר וקטור זה עם וקטור הלוח שנסמן ב S_0 נקבל:

$$S_0 + t_1 + t_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

וקטור הלוח שמתקבל לאחר חיבור אכן תואם לתוצאה המצופה, כפי שמתואר באיור 3.3ב.

הערה 3.5: היות ווקטור שינוי שייך לשדה ${\mathbb{Z}_2}^n$, ניתן לתאר צירוף לחיצות במשחק בעזרת צירוף ליניארי בשדה ${\mathbb{Z}_2}^n$. כפי שתיארנו במסקנה 3.1 סקלרים בצירוף ליניארי זה הם 0 או 1.

הערה 3.6: מכיוון שכל משחק מתחיל כאשר מצב כל הצמתים הינו 0, ניתן להשמיט את וקטור הלוח ההתחלתי.

איור 3.3 : דוגמה לתיאור וקטור שינוי במהלך משחק על גרף

(א) מצב של הגרף לפני לחיצה

(ב) מצב של הגרף לאחר הלחיצות

כאשר נסמן מצב הלוח ההתחלתי ב S_0 , מתקיים :

$$S_0 + \sum_{j=1}^{n} \vec{t}_j x_j = \sum_{j=1}^{n} \vec{t}_j x_j \tag{1}$$

בעקבות כך ניתן לתאר את בעיית המשחק בצורה הבאה :

$$\sum_{j=1}^{n} \vec{t}_j x_j = \vec{1} \tag{2}$$

כאשר $\vec{1}$ מתאר את וקטור הלוח כאשר המשחק פתור. n מתאר את מספר הצמתים בגרף. נשים לב שאם ידוע צירוף $x = \begin{bmatrix} x_1, & x_2, & \cdots, x_n \end{bmatrix}$ שמקיים את המשוואה 2 אז הוא פתרון של משחק על גרף. כדי להגיע לפתרון על גרף נלחץ על הצמתים שמספרם שווה לאינדקסים j שמקיימים $x_j = 1$ בצירוף x.

מערכת משוואות שמתוארת בנוסחה 2 אפשר לתאר בעזרת מטריצה כמו שמתואר בנוסחה 3.

$$\begin{bmatrix} t_1 & t_2 & \cdots & t_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \cdots \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,n} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ t_{i,j} & t_{i,2} & \cdots & t_{i,n} \\ \cdots & \cdots & \cdots & \cdots \\ t_{n,1} & t_{n,2} & \cdots & t_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_i \\ \cdots \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \cdots \\ 1 \\ \cdots \\ 1 \end{bmatrix} \tag{3}$$

נשים לב שלמטריצה במשוואה 3, נסמנה ב A, ערכי המטריצה שמקיימים $A_{i,j} = 1$ הם באינדקסים i, j, בהם צמתים n_i, n_j שהם שכנים או זהים ($i = j$).

הגדרה 3.3: מטריצה שקבלנו במשוואה 3 תקראה מטריצת שכנויות של משחק.

הערה 3.7: היות ובגרף יחס שכנות הוא סימטרי מטריצה שכנות היא גם סימטרית.

דוגמה 3.2: מטריצה שכנות עבור גרף באיור 3.3 :

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

הגדרה 3.4: וקטור פתרון של משחק הוא וקטור $\vec{x}$ שניראה במשוואה 3.

נזכיר שאם $\vec{x}$ וקטור פתרון של המערכת ו $x_i = 1$ אז כדי לפתור משחק צריך ללחוץ על לחצן i. בנוסף נציין שאתכן קיימים כמה פתרונות אפשריים.

הגדרה 3.5: שיטת פתרון הנעזרת ביצירת מטריצה שכנויות ומציאת וקטור פתרון תקראה אלגוריתם מבוסס מטריצת שכנויות.

שיטת הפתרון מבוססת מטריצת השכנויות אפשר לראות ב [2]. מרגע שהצלחנו לתאר את הבעיה, בצורה משוואות לינארית על שדה $\mathbb{Z}_2{}^n$, נוכל להיעזר בכלים של אלגברה לינארית כדי למצוא פתרון, כמו מציאת פתרון בעזרת דירוג או מציאת מטריצה פסאודו הפוכה וכולי.

הערה 3.8: 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 $[m \cdot n \times m \cdot n]$, האם קיימות שיטות לפתור את המשחק בעזרת מערכת המשוואת יותר מצומצמת?

3.2 אלגוריתם שמבוסס על מילוי עקבי של שורות

עד כה הצגנו בעבודה שיטת פתרון הנעזרת במטריצת שכנויות. נרצה להראות שיטה נוספת למציאת פתרון. שיטת הפתרון שנציג נובעת מהערה 3.8, נציג שיטה משופרת שמצמצמת את כמות המשתנים והמשוואות למציאת פתרון למשחק על לוח $m \times n$ ב $\min(m, n)$ משוואות ומשתנים. צמצום גודל מערכת משואות יכולה להוביל לחישוב מהיר יותר וניצול טוב יותר של המידע.

המאמר [1] מציג שיטה למציאת פתרון של משחק על לוח. בפרק זה נציג את הגישה שמתוארת ב [1] ונראה את הקשר בין השיטות. לגישה החדשה ניקרא "אלגוריתם שמבוסס על מילוי עקבי של שרות" או בקצרה מילוי עקבי.

נתאר אלגוריתם מילוי עקבי בשלבים, כל שלב נדגים על לוח 3×3 שמתואר באיור 3.4. הלוח 3×3 הנתון ממוספר באינדקסים כפי שתיארנו בהערה 3.4. אלגוריתם מילוי עקבי מתבסס על רעיון, שפתרון של המשחק הוא סדרה של לחיצות על משבצות מסוימות. נשייך לכל משבצת משתנה שיכול לקבל שני ערכים : 0 אם המשבצת אינה מופיעה בסדרת הלחיצות של הפתרון ו- 1 אם המשבצת כן מופיעה בסדרת הלחיצות של הפתרון. מראש לא ידוע לנו האם משבצות של השורה הראשונה יופיעו בסדרת הפתרון. נסמן ב x_1, x_2, x_3 משתנים של משבצות בשורה ראשונה כפי שמתואר באיור 3.4 . את שאר המשתנים נתאר בעזרת משתנים משורה הראשונה.

איור 3.4 : לוח 3×3 עם אינדקסים

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix}$$

נבחין בתופעה הבאה : על מנת שנורה 1 תהיה דולקת סכום המשתנה שלה ומשתנים של משבצות השכנות במודולו 2 חייב להיות 1. המשוואה המתקבלת עבור איור 3.4 :

$$x_1 + x_2 + x_4 = 1 \tag{4}$$

לכן משתנה x_4, חייב להיות שווה ל $x_1 + x_2 + 1$ היות ומתקיים :

$$x_1 + x_2 + x_4 = 1 \Rightarrow x_4 = x_1 + x_2 + 1$$

על מנת שהנורה 2 תהיה דלוקה, סכום המשתנה שלה והמשתנים של משבצות שכנות במודולו 2 חייב להיות 1. לכן מתקבל שערכו של x_5 חייב להיות שווה ל $x_1 + x_2 + x_3 + 1$ היות ומתקיים :

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

באופן דומה מחשבים את המשתנים של שאר המשבצות בשורה השנייה. על פי אותם שיקולים נחשב את המשתנים של משבצות בשורות הבאות.

נבחין שלאחר שמילאנו את כל הלוח כמו שמתואר באיור 3.5. אם היה ידוע איזה משבצות משורה הראשונה שייכים לסדרת הלחיצות של פתרון אז, פתרנו את המשחק.

הגדרה 3.6: המשוואה שתיארה את סכום משתנה של משבצת i ומשתנים של כל שכניה תקראה משוואת אילוצים על משבצת i.

הערה 3.9: תיאור כל משוואות האילוצים כמערכת משוואות לינארית, הינה גישה נוספת למדל את המשחק. פתרון של מערכת זו יוביל לפתרון המשחק. בנוסף נראה בהמשך שנקבל אותה מערכת משוואות כמו אלגוריתם מבוסס על מטריצת שכנויות שהוגדר ב 3.5.

משוואה 4 הינה משוואת האילוצים של משבצת 4. עבור משחק לוח $[m \times n]$ שהלחצנים בו ממוספרים לפי הערה 3.4 ניתן לנסח את משוואת אילוצים:

$$x^*_{i-n} + x^*_{i-1} + x^*_i + x^*_{i+1} + x^*_{i+n} = 1 \quad x^*_i = \begin{cases} x_i & \text{if } i \in [1, n \cdot m] \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

דוגמה 3.3: חישוב משתנה x_7 בעזרת משתנים מהשורה ראשונה. היות ו- x_7 משורה השלישית לכן משוואת האילוצים שנעזר בה תהיה מהשורה השנייה של המשבצת שמעליו, שזאת משבצת 4.

משוואת האילוצים של משבצת 4:

$$x_1 + x_4 + x_5 + x_7 = 1$$

לכן:

$$x_7 = 1 + x_1 + x_4 + x_5$$

היות ומילוי הוא עקבי, שורה אחרי שורה, ידוע מראש המשוואות של המשתנים משורה השנייה:

$$x_4 = 1 + x_1 + x_2$$
$$x_5 = 1 + x_1 + x_2 + x_3$$

נציב ערכים אילו:

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

איור 3.5: תרגום כל הלחצנים לפי המשתנים של השורה הראשונה

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ 1 + x_1 + x_2 & 1 + x_1 + x_2 + x_3 & 1 + x_2 + x_3 \\ 1 + x_1 + x_3 & 0 & 1 + x_1 + x_3 \end{bmatrix}$$

על מנת ליצור מערכת משוואות שתפתור את המשחק מחבר מאמר [1], מוסיף לשורה האחרונה עוד שורה. שורה זו הינה שורה דמיונית ומחשבים בא משבצות לפי אותם שיקולים. משום שזאת שורה דמיונית, אנחנו לא מדליקים אף נורה, ערכי המשתנים של המשבצות שלה חייבים להיות 0.

איור 3.6 : לוח 3×3 מלא, כולל שורה וירטואלית

$$\left[\begin{array}{ccc} x_1 & x_2 & x_3 \\ 1+x_1+x_2 & 1+x_1+x_2+x_3 & 1+x_2+x_3 \\ 1+x_1+x_3 & 0 & 1+x_1+x_3 \\ \hline 1+x_2+x_3 & x_1+x_2+x_3 & 1+x_1+x_2 \end{array}\right]$$

לכן אם נסתכל על מערכת המשוואות שנוצרה בשורה הדמיונית, מתקבל מערכת משוואות עם 3 משוואות ו- 3 משתנים היות וערכי השורה הם אפס.

איור 3.7 : מערכת משוואות המתקבלת משיטת מילוי עקבי בלוח 3×3

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

הגדרה 3.7: שיטת פתרון הנעזרת ביצרת מערכת משוואת ממשתנים בשורה הראשונה, הנעזרת בשורה דמיונית ומציאת וקטור פתרון בעזרת אותה מערכת, תקרא אלגוריתם המבוסס על מילוי עקבי של שורות.

את השיטה הדגמנו על לוח 3×3, אפשר היה להדגים על כל לוח אחר ושיטה עדיין אתה עובדת. בנוסף, שיטת מילוי עקבי אינה חייבת להיות מבוססת על מילוי שורות אלה אפשר היה לבנות את אותה שיטה גם על עמודות. מכיוון שאפשר להפעיל את השיטה על שורות או עמודות כדי לקבל מערכת משוואות מינימלית, נבחר את הכיוון שיש בו פחות משבצות. בשלב זה נראה את הקשר בין שתי השיטות.

משפט 3.1: מטריצה מיצג של מערכת משוואות האילוצים, היא מטריצת שכנויות.

נזכיר שהגדרנו מטריצת שכנויות בהגדרה 3.3. משוואות האילוצים הן בסיס אלגוריתם מילוי עקבי. אם נפרוס את משוואות האילוצים נקבל גם מערכת משוואות שפותרת את המשחק, כפי הזכרנו בהערה 3.9, אך כמות המשוואות הינה n^2. נבחין שאם נרשום את מערכת משוואות בסדר מספור המשבצות, כפי שמתואר באיור 3.4, אז המטריצה המייצגת מערכת משוואות זו, תהיה מטריצה שכנויות.

דוגמה 3.4: נבנה את מערכת המשוואות של משוואות האילוצים על לוח 2×2 ומטריצת השכנויות, לפי סדר המספור שקבענו.

איור 3.8: לוח 2×2

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

וקטורי השינויים:

$$t_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}, t_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}, t_3 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}, t_4 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

לכן מטריצת שכנויות נראית כך:

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

אם נסדר את המשוואות במערכת המשוואות, לפי סדר האינדקסים של המשבצות, נקבל מערכת מהצורה:

$$x_1 + x_2 + x_3 = 1$$
$$x_1 + x_2 + x_4 = 1$$
$$x_1 + x_3 + x_4 = 1$$
$$x_2 + x_3 + x_4 = 1$$

אפשר לראות שהמטריצה המייצגת של המערכת, זהה למטריצת שכנויות.

ניתן הסבר נוסף לשורה הדמיונית ואיך מערכת המשוואות מצטמצמת ביחס למטריצה שכנויות. ההסבר ניתן בעזרת תיאור אלגוריתם מילוי עקבי, במקום על לוח נשתמש במטריצת שכנויות ופעולות דירוג. נראה את פעולות הדירוג על לוח 3×3. בדוגמא נראה שאלגוריתם מילוי עקבי הוא דירוג מתוחכם של מטריצה באיור 3.9 מורחבת $[M|\vec{1}]$.

כדי לחשב את x_4 באיור 3.4 השתמשנו במשוואת האילוצים של משבצת 1:

$$x_1 + x_2 + x_4 = 1 \Rightarrow x_4 = x_1 + x_2 + x_3 + 1$$

איור 3.9 : מערכת משוואות מורחבת של משחק על לוח 3×3

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

נבחין שלושת השורות הראשונות של מטריצה 3.9 מתארות את המשתנים x_4, x_5, x_6 בעזרת משתנים משורה הראשונה. נשים לב ששורות אלו זהות למשוואות האילוצים של משבצות $1,2,3$ משורה הראשונה בלוח כפי שהצגנו בשיטה למילוי עקבי המקורית. אם ננסה לתאר את משתנה x_7 בשיטה המקורית, נסתכל על משוואת האילוצים של המשבצת שמעליו, x_4. משוואת האילוצים שלו :

$$x_1 + x_4 + x_5 + x_7 = 1$$

אפשר לראות שקיבלנו תלות בערכי המשתנים x_4, x_5. באותה דרך נסתכל על שורה 4 במטריצה, ניראה שהיא גם תלויה ב x_4, x_5. היות ואמרנו שתיאורנו אותם בשורות $1,2$ במטריצה, נעזר בשורות אלו כדי לתאר את x_7 בעזרת המשתנים x_1, x_2, x_3 . אופן דירוג השורה היה בעזרת הפעולות :

$$r_4 \leftarrow r_4 + r_1$$
$$r_4 \leftarrow r_4 + r_2$$

פעולות אלו שקולות לפעולות האלגברית :

$$(x_1 + x_4 + x_5 + x_7 + 1) + (x_1 + x_2 + x_4 + 1) = 0 \Rightarrow x_2 + x_5 + x_7 = 0$$
$$(x_2 + x_5 + x_7) + (x_1 + x_2 + x_3 + x_5 + 1) = 0 \Rightarrow x_1 + x_3 + x_7 + 1 = 0$$

הראנו ששורה במטריצה מתארת משוואת אילוצים, וניתן בעזרת פעולות שורות לבטא את המשוואה רק בעזרת משתנים משורה הראשונה בלוח ומשתנה שרצינו לתאר. כך תרגמנו את אלגוריתם מילוי עקבי לאלגוריתם דירוג של מטריצה מורחבת. התוצאה שמתקבלת, לאחר מעבר על כל השורות היא המטריצה באיור 3.10.

איור 3.10 : המטריצה לאחר פעולת שורות על כל שורות

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

נשים לב שבמטריצה 3.10 שלושת השורות התחתונות אינן מתארות אף משתנה בלוח, ומתוארות אך ורק על ידי המשתנים x_1, x_2, x_3. אם נתאר את השורות הללו כמערכת משוואות נקבל את אותה מערכת משוואת של אלגוריתם מילוי עקבי על לוח. אפשר לראות מערכת משוואות זו באיור 3.7.

בעזרת תיאור אלגוריתם מילוי עקבי בעזרת דירוג מטריצה התקבלנו המסקנות :

מסקנה 3.2: הסבר לשורה הדמיונית, משבצות בשורה הדמיונית הן השורות במטריצה שלאחר דירוג, בשורות המדורגות מופיעים רק משתנים משורה הראשונה בלוח.

מסקנה 3.3: צמצום כמות המשתנים, כמות המשתנים במטריצת השכנויות קטנה כיוון שניתן לתאר את הפתרון אך ורק בעזרת משבצות שנלחצו מהשורה הראשונה. שאר הלחיצות של הפתרון מאולצות.

3.3 השוואה בין שתי השיטות למציאת פתרון

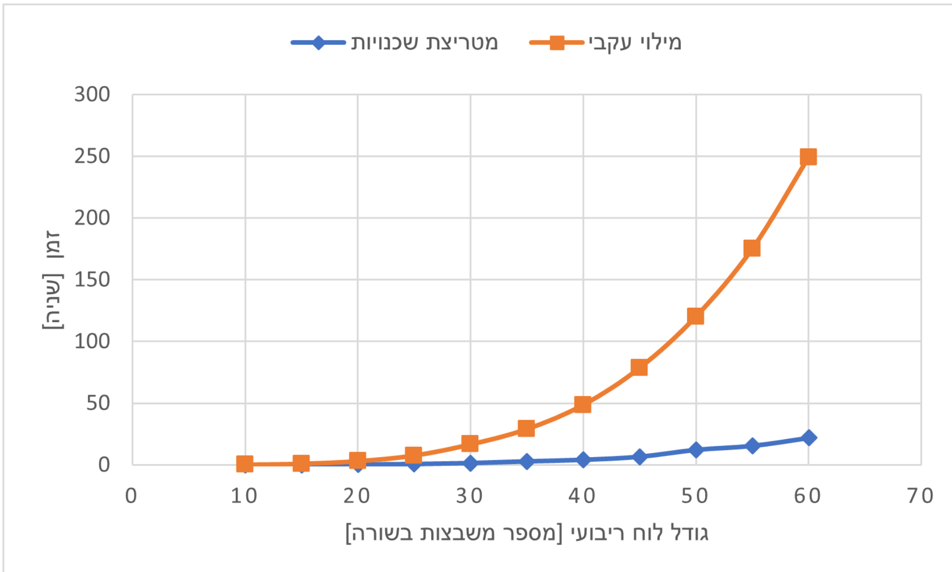
חישוב סיבוכיות של דירוג מטריצה כללית בגודל $n^2 \times n^2$ הוא $O(n^2 \cdot n^4) = O(n^6)$.

דירוג שורה בשיטת מילוי עקבי מבצעים פעולות שורות כמספר השכנים במשואה האילוצים, עבור המשבצת שמעליו. לכן יש לכל יותר 4 פעולות שורות שצריך לבצע כדי לקבל מערכת משוואות מצומצמת. מחישוב הסיבוכיות פעולה זה תיקח $O(n^2 \cdot n^2) = O(n^4)$. דירוג n המשתנים הנותרים הוא בסיבוכיות $O(n \cdot n^2) =$ $O(n^3)$.

נראה את הסיבוכיות בפועל על ידי הצגת זמני חישוב. באיור 3.11 אפשר לראות את הביצועים של שני האלגוריתמים, ציר ה x מתאר גודל שורה של לוח ריבועי. ציר ה y מתאר זמן בשניות שלוקח לאלגוריתם לרוץ.

לפי התוצאות של איור 3.11 ניראה ששיטה למילוי עקבי שבתאוריה יותר אופטימליות לוקחת יותר זמן. אחת הסיבות לקח שפונקציה שפותרת מערכת משוואות הינה פונקציית ספרייה, היודע כאופטימלי.

איור 3.11 : גרף מתאר ביצועים על לוח ריבועי גודל שורה מול זמן



3.4 דיון לגבי משחק על גרף

תיארנו את המשחק על גרף מסיבות הבאות :

1. ככול שהמבנה כללי יותר התאוריה שמפתחים מתאימה ליותר בעיות.
2. תאוריית הגרפים רחבה מאד וקל לתאר בעזרתה בעיות.
3. מבליט את מהות הבעיה והגדרה הבסיסית ביותר של המשחק.

בפועל כשהצגנו את האלגוריתמים לפתרון משחק התגלתה התמונה המלאה. כלומר שני האלגוריתמים שתיארנו מתארים את המשחק כגרף, היות ושני האלגוריתמים בנויים על מטריצת השכנויות, מטריצת שכנויות היא שיטת ייצוג קלאסי לגרפים. קשר זה מדגיש ומראה שלפעמים רק תיאור מדויק של הבעיה מספיק כדי למצוא לבעיה פתרון.

4 קיום פתרון ומספר הפתרונות עבור משחק על גרף

עד כה הצגנו שיטות למציאת פתרון, שיטות עלו האירו את העובדה ששאלת קיום הפתרון למשחק על גרף שקולה לשאלת קיום הפתרון למערכת משוואות לינאריות. בפרק זה נרצה להוכיח קיום פתרון למשחק לכל גרף. העובדה

שקיים פתרון לכל כל גרף אינה מובנת מעליה. אחד המקומות ששאלה זה נשאלה היא בספר [4], בעבודתנו נראה הוכחה קצת שונה בעזרת הכלים שפיתחנו. אומנם הוכחה קיום פתרון הופיע לראשונה במאמר [3]. עובדה מעניינת נוספת שהוצגה במאמר היא, שיש רק הוכחה שמבוססת על אלגברה לינארית לשאלת קיום פתרון.

4.1 הוכחת קיום פתרון על גרף

הגדרה 4.1: תהי $\mathbb{S}$ קבוצה, פעולה בינארית על $\mathbb{S}$ היא פונקציה $\mathbb{S} \times \mathbb{S} \to \mathbb{S}$ המתאימה לכל זוג סדור. פעולה בינארית עבור הזוג (s_1, s_2) תסומן $\langle s_1, s_2 \rangle$.

הגדרה 4.2: מכפלה פנימית היא פעולה בינארית שמוגדרת לכל זוג וקטורים במרחב וקטורי $\mathbb{S}^n \subseteq \mathbb{R}^n$ ומעתיקה אותם ל $\mathbb{S}$. תהי $\mathbb{S}^n \subseteq \mathbb{R}^n$ מרחב וקטורי עם מכפלה פנימית, לכל $\vec{x}, \vec{y}, \vec{z} \in \mathbb{S}^n$ ו $\alpha \in \mathbb{S}$, פעולה בינארית תקראה מכפלה פנימית אם היא מקיימת 4 אקסיומות הבאות:

1. $\langle \vec{x}, \vec{y} \rangle = \langle \vec{y}, \vec{x} \rangle$
2. $\langle \vec{x} + \vec{y}, \vec{z} \rangle = \langle \vec{x}, \vec{z} \rangle + \langle \vec{y}, \vec{z} \rangle$
3. $\langle \alpha\, \vec{x}, \vec{y} \rangle = \alpha \langle \vec{x}, \vec{y} \rangle$
4. (א) $\langle x, x \rangle \geq 0$

   (ב) $\langle x, x \rangle = 0 \iff x = \vec{0}$

הגדרה 4.3: לכל שני וקטורים $\vec{x}, \vec{y} \in \mathbb{Z}_2{}^n$ נגדיר פעולה הבאה:

$$\vec{x} \cdot \vec{y} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix} = x_1 y_1 + x_2 y_2 + \cdots x_n y_n \tag{6}$$

לפעולה זה בין שני וקטורים ב $\mathbb{Z}_2{}^n$ ניקרא מכפלה סקלרית.

הערה 4.1: פעולה שהגדרנו בהגדרה 4.3 נקראת מכפלה סקלרית למרות שהיא מקיימת רק 3 מתוך 4 תכונות של מכפלה סקלרית ב - R^n. תכונה $< \vec{u}, \vec{u} >= 0 \Leftrightarrow \vec{u} = \vec{0}$ לא מתקיימת.

דוגמה 4.1: המחשה להערה 4.1:

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

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 $\mathrm{Col}A^T \perp \mathrm{Nul}A$ ו $\mathrm{Col}A \perp \mathrm{Nul}A^T$

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,

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

נציין שעבור מטריצות $A \in Z_2{}^{m\times n}$ מאותם שיקולים אותה הוכחה נכונה, רק עבור מכפלה הסקלרית שהגדרנו בהגדרה 4.3. □

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 4.2: 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: לכל משחק על גרף קיים פתרון.

הוכחה. תהי $A \in Z_2{}^{n\times n}$ מטריצת השכנויות של משחק שהגדרנו ב 3.5. למטריצה זו מתקיימים תכונות הבאות:

1. מטריצה סימטרית לפי הערה 3.7
2. המטריצה ריבועית.
3. האיברים על האלכסון מטריצה ערכם שווה ל 1.

כדי להראות שלמשחק יש פתרון צריך להראות שקיים פתרון למערכת:

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

במקרה ש A מטריצה הפיכה קיים פתרון, ופתרון יחיד. עבור המקרה שמטריצה אינה הפיכה כלומר, $\mathrm{Nul}A \neq \{\vec{0}\}$. תהי $\vec{x} \in \mathrm{Nul}A$ מהגדרה זו מתקיים $A\vec{x} = \vec{0}$ לכן:

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

נסמן $\vec{x} = [x_1, x_2, \cdots, x_n]^T$

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

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:

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

הבחנה נוספת היא, לא משנה אם הערך 0 או 1 מתקיים השוויון, $x^2 = x$ לכן פישוט נוסף למשוואה 7:

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

היות ו $\vec{x}^T A\vec{x} = 0$ לכן מתקיים:

$$a_{1,1}x_1 + a_{2,2}x_2 + a_{n,n}x_n = 0$$

כלומר $\vec{x} \perp \vec{1}$ כאשר $x \in \text{Nul}A$ לפי משפט 4.1 מתקבל $\vec{1} \in \text{Col}A^T$. מטריצה A הינה סימטרית לכן $A^T = A$ ומתקיים $\vec{1} \in \text{Col}A$. למעשה הוכנו שקיים $\vec{x}$ שמקיים פתרון למערכת $A\vec{x} = \vec{1}$ כלומר, קיים פתרון למשחק. □

הוכחת קיום הפתרון עבור המשחק כפי שהגדרנו על גרף הושגה, מסקנה נאיבית שניתן אולי לחשוב היא, שלכל מצב התחלתי אפשרי היה לפתור את המשחק. בחלק זה של הפרק ננסה לחדד מתי קיים פתרון כאשר מצבים התחלתיים וסופיים של אותו משחק שונים מהמשחק המקורי.

הגדרה 4.4: משחק אחר שאפשר להציע הוא משחק האורות כללי יותר מוגדר כך: לוח הבקרה נשאר זהה למשחק המקורי כלומר, שינוי נורות לאחר לחיצה מתנהג נשאר כפי שהוגדר במשחק המקורי. הבדל בין משחק החדש למקורי: מצב התחלתי הוא שחלק מנורות דולקות וחלק כבויות ורוצים להגיע למצב סופי שגם בו חלק מנורות דולקות וחלק כבויות.

הערה 4.4: רק בפרק זה נשתמש במשחק כפי שהגדרה בהגדרה 4.4. בכל שאר פרקים נשתמש בהגדרה המקורית של המשחק.

עבור המשחק שהגדרנו 4.4 אותו קורא נאיבי יכול להניח שגם עבור משחק שכזה תמיד קיים פתרון.

דוגמה 4.3: דוגמה למשחק על לוח לפי הגדרה 4.4, שאין לו פתרון. תהי לוח 1×2 בו מצב התחלתי הוא שהנורה הימנית ביותר דלוקה ונרצה לעבור למצב הסופי בו כל הנורות דולקות. נראה שהמשחק אינו פתיר לפי הצגת כל הלוחות האפשריים להתקבל במשחק על ידי כל הצירופים השונים של לחיצות על הלוח. אם נמספר את המשבצות לפי שיטת המספור שציינו בהערה 3.4, אז אוסף כל צירופי הלחיצות הם:

$$(), (1), (2), (1,2)$$

כפי שניתן לראות באיור 4.1 באף צירוף לחיצות לא ניתן להגיע למצב הסופי שהוגדר במשחק.

איור 4.1: מצבי הלוחות לאחר לחיצה של צירוף

(א) עבור צירוף $()$	(ב) עבור צירוף (1)	(ג) עבור צירוף (2)	(ד) עבור צירוף $(1,2)$
$\begin{bmatrix} 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \end{bmatrix}$

כדי לבדוק אם קיים פתרון למשחק הכללי שהגדרנו בהגדרה 4.4 נוכל להיעזר בהוכחה 4.2.

משפט 4.3: למשחק קיים פתרון אם ורק אם וקטור הפרש בין מצב סופי ומצב ההתחלתי שייך למרחב העמודות של מטריצת שכנויות.

הוכחה. נגדיר קודם את המשחק שהגדרנו בהגדרה 4.4 אלגברית. לפי משוואה 1 אפשר לנסח אלגברית את המשחק כך. תהי A מטריצת השכנויות, $\vec{S_0}$ מצב התחלתי של המשחק, ו $\vec{S_e}$ מצב הסופי של המשחק. נחפש צירוף לא סדור של לחיצות $\vec{x}$ כך שמתקיים:

$$\vec{S_0} + A\vec{x} = \vec{S_e}$$

נעביר אגפים ונקבל:

$$A\vec{x} = \vec{S_e} - \vec{S_0}$$

הוכחה של משפט 4.2 בנויה על עובדה שצריך להוכיך שווקטור $\vec{1}$ שייך ל $\mathrm{Col}A$. במקרה של המשחק החדש כדי להראות שקיים פתרון למשחק מספיק להוכיח ש $\vec{S_e} - \vec{S_0}$ שייך ל $\mathrm{Col}A$. $\square$

הערה 4.5: כדי לבדוק שווקטור $\vec{v} \in \mathbb{R}^n$ שייך ל $\mathrm{Nul}A$ של מטריצה $A \in \mathbb{R}^{m\times n}$ מספיק להראות שמתקיים:

$$A\vec{v} = \vec{0}$$

הערה 4.6: בגלל הערה 4.3 לא ניתן לומר שווקטורים $z \in \mathbb{Z}_2{}^n$ המקיים $z \in \mathrm{Col}A$ אז הוא לא שייך ל $\mathrm{Nul}A$. לו היה ניתן לומר זאת, היה קל לבדוק שלמשחק קיים פתרון על ידי בדיקה:

$$\vec{S_e} - \vec{S_0} \notin \mathrm{Nul}A$$

משפט 4.4: תהי משחק לפי הגדרה 4.4, $\vec{S_e}$ מצב הסופי, $\vec{S_0}$ התחלתי של משחק, ו A מטריצת שכנויות. אם $\vec{S_e} - \vec{S_0} \notin \mathrm{Nul}A$ אז למשחק קיים פתרון.

הוכחה. אם $\vec{S_e} - \vec{S_0} \notin \mathrm{Nul}A$ אז בהכרח

$$\vec{S_e} - \vec{S_0} \in \mathrm{Col}A^T = \mathrm{Col}A$$

לפי משפט 4.3 למשחק קיים פתרון. □

משפט 4.4 יכול להקל על הבדיקה אם המשחק פתיר. אך, קיימים מקרים שמתקיים:

$$\vec{S_e} - \vec{S_0} \in \mathrm{Nul}A$$

ועדיין קיים למשחק פתרון.

4.2 מספר הפתרונות עבור כל גרף

הוכחנו שלכל משחק על גרף יש פתרון שסדר לחיצות אינו משנה את התוצאה על הלוח. השאלה שנשאל בפרק זה מה אפשר לומר על מספר פתרונות בעזרת הכלים שהצגנו. נציין ששני פתרונות יקראו שונים אם, קיים לפחות לחצן אחד שמבדיל בין הפתרונות. זאת אומר שקיים לחצן ששייך לפתרון ראשון ולא שייך לפתרון שני. כפי שציינו קודם סדר הלחיצות לא משנה את הפתרון לכן, פתרון הינו קבוצה של לחצנים שצריכים להליחץ כדי להגיע למצב סופי של המשחק.

דוגמה 4.4: נרצה להראות שקיימים מספר פתרונות למשחק. ניקח לדוגמה משחק על גרף שמתואר באיור 4.2. היות וגרף הינו קליקה לכן לחיצה בודדת על אחד הצמתים תדליק את כל הלחצנים. מתקבל שקבוצה

$$G = \{\{v_1\}, \{v_2\}, \{v_3\}\}$$

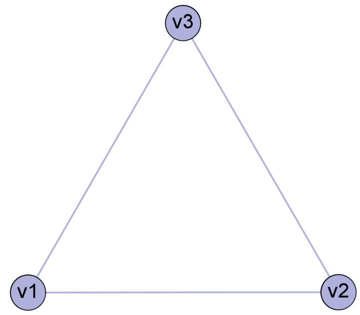
היא חלק מקבוצת הפתרונות של המשחק על גרף. כפי שניתן לראות קיימים לפחות שלושה פתרונות למשחק הנתון.

נשאל פרק זה איך אפשר למצוא את כמות הפתרונות במשחק. כדי לענות על שאלה נצטרך להציג כמה מושגים מאלגברה לינארית. בשיטת דירוג של גאוס על מטריצה, עוברים על השורות ומנסים בעזרת פעולות שורות ליצור עמודות בהם מופיע איבר בודד ששונה מאפס.

הגדרה 4.5: איבר מוביל בשורה הוא האביר הראשון בשורה ששונה מאפס לאחר דירוג.

הגדרה 4.6: נעלמים שאברהם מובילים אחרי דירוג אקראו נעלמים מובילים.

איור 4.2 : משחק על גרף



הגדרה 4.7: במטריצה מדורגת בשיטת גאוס, עמודות שאין בהם איבר מוביל ניקראות עמודה חופשית.

הגדרה 4.8: נעלמים שעמודה שלהם חופשית אחרי דירוג יקראו נעלמים חופשים.

הגדרה 4.9: מספר נעלמים חופשיים במטריצה נקרא דרגת החופש.

הגדרה 4.10: מספר עמודות במטריצה בהם יש איבר מוביל, נקרא דרגה. נסמן את הדרגה של המטריצה ב $\text{rank}(A)$.

דוגמה 4.5: לסיכום המושגים ניקח מטריצה מדורגת הבאה :

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

איברים הצבועים בכחול הם איברים מובילים, עמודות חופשיות צבועות בסגול. דרגת החופש של המטריצה A היא 3.

הערה 4.7: דרך נוספת לחשב את דרגת החופש $F(A)$ של מטריצה $A \in \mathbb{R}^{m\times n}$ כשידוע שדרגה של מטריצה A הוא $\text{rank}(A)$ בעזרת הנוסחה הבאה :

$$F(A) = n - rank(A)$$

לאחר שהגדרנו את מושג דרגת החופש נוכל לנסח את המשפט המרכזי של הפרק.

משפט 4.5: מספר הפתרונות של משחק שווה ל 2^k. כאשר k שווה לדרגת החופש של מטריצה השכנויות של המשחק.

הוכחה. נזכיר שאת מטריצה השכנויות הגדרנו בהגדרה 3.3. נסמן ב A את מטריצת השכנויות, ב X את כל הוקטורים שהם פתרון של המערכת:

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

לפי משפט 4.2 ידוע שקיים לפחות פתרון אחד למשחק. היות וידוע שיש לפחות פתרון אחד אפשר לתאר את כל פתרונות כך:

$$X = X_n + x_0$$

כאשר X_n קבוצת כל הוקטורים השייכים לפתרון מרחב האפס של מטריצה A, x_0 פתרון פרטי, ו X קבוצת כל פתרונות של המשחק. לכן גודל קבוצת הפתרונות שווה לגודל מרחב האפס. ידוע שגודל מרחב האפס תלוי בדרגת החופש לכן, מספר הווקטורים בבסיס מרחב האפס שווה לדרגת החופש שנסמן ב k. כמות הווקטורים במרחב האפס שווה לכל וקטורים שמוגדרים בצירוף לינארי:

$$x = a_1x_1 + a_1x_1 + a_2x_2 + \cdots + a_kx_k$$

כאשר הערכים של $a_i \in Z_2$ לכן, לכל מקדם יכול להיות 2 ערכים. היות ווקטורים בלתי תלויים לינארית, קיבלנו שגודל קבוצת הפתרונות הוא בדיוק 2^k. $\square$

הבחנה נוספת שנציין, אפשר לחסום את כמות הפתרונות. חסם עליון טריוויאלי לכמות המקסימלית של פתרונות הוא 2^n פתרונות כאשר n שווה למספר הלחצנים. זאת מכיין שלא יכול להיות יותר פתרונות מאשר כמות הלחיצות השונות האפשריות במשחק. השאלה שנשאלת האם אפשר למצוא חסם הדוק יותר?

הערה 4.8: עבור משחק לוח מלבני בגודל $m \times n$ קיים לכל יותר 2^k כאשר $k = \min\{m, n\}$ פתרונות שונים

הוכחה. הערה זה נכונה לפי אלגוריתם שמבוסס על מילוי עקבי של שורות שהגדרנו 3.7 ניתן לתרגם את משחק ל k משוואות ש k יכול להיות מספר שורות או עמודות. $\square$

לסיכום נציג באיור 4.3 את כמות הפתרונות שיש בלוח. השורות והעמודות בטבלה מיצגות את ממדי הלוח. טבלה זאת חושבה על ידי משפט 4.5.

איור 4.3: 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

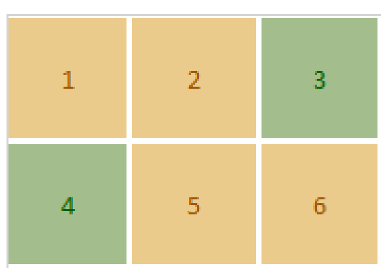
5 פתרון אופטימלי עבור לוחות מלבניים

בפרק זה נציג סוג מסוים של פתרונות, לסוג זה נקראה פתרון אופטימלי. פתרון זה מקל רבות על המשחק כיוון שמצמצם את כמות הלחיצות האפשריות בכל מצב במשחק.

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

באיור 5.1 ניתן דוגמא לפתרון מינמלי בלוח 3×2. כשלוחצים על לחצנים $3,4$ כל הנורות נדלקות, ואף אחת מהם לא נכבת באף לחיצה.

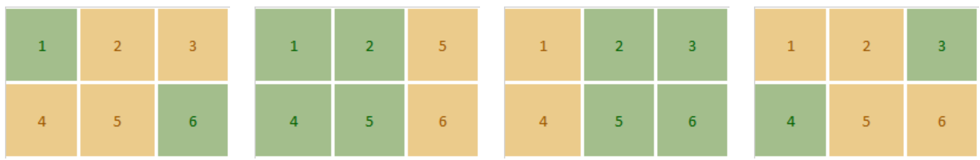
איור 5.1: פתרון מינמלי של משחק



נרצה לחדד ולהדגיש עד כמה קל למצוא פתרונות אופטימלים. אם ניקח לוח 3×2 כפי שמתואר באיור 5.1, ונתבונן במספר כל הפתרונות שיש ללוח זה, כפי שמתואר בטבלה 4.3, ניראה שיש 4 פתרונות. שני פתרונות אופטימליים ושני פתרונות לא אופטימליים, נזמין את הקורא לחפש את כל הפתרונות. כנראה ששני הפתרונות

האופטימליים נימצאו 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



בפרק זה נפתור את השאלה : לאיזה לוחות קיים פתרון מינמלי כאשר מצב התחלתי הוא שכל הנורות במצב 0 ?

5.1 הוכחת אי קיום לפתרון אופטימלי על לוחות ששני הממדים גדולים מ 4×4

משפט 5.1: קיים פתרון אופטימלי למשחק $m \times 2$ כאשר m הוא אי זוגי.

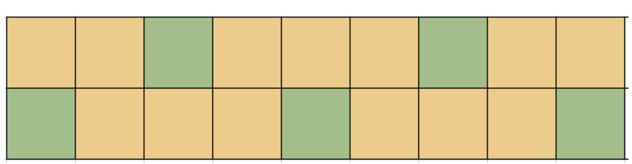
הוכחה. נוכיח בעזרת אינדוקציה. נתחיל על לוח 1×2, כל לחיצה בודדת על משבצת מובילה לפתרון אופטימלי של המשחק. נניח וקיים לוח $m \times 2$ בו מתקיים פתרון אופטימלי כאשר m אי זוגי. ללוח קיים פתרון אופטימלי לכן הנורות השתנו רק פעם אחת, מה שאומר שמשבצות בעמודה האחרונה בהכרח שונו על ידי אותה לחיצה. לחיצה זו בהכרח תהיה על משבצת בעמודה האחרונה. נניח, בלי הגבלת הכלליות, שהמשבצת שנלחצה היא בשורה 1. עבור לוח $(m+2) \times 2$ נבצע את אותם מהלכים כמו שביצענו עבור לוח $m \times 2$. נשים לב שהנורות שנותרו במצבם ההתחלתי הן :

$$\{n_{1,m+2}, n_{2,m+1}, n_{2,m+2}\}$$

נשים לב שלחיצה על משבצת $n_{2,m+1}$ פותרת את המשחק. □

הערה 5.1: טענה 5.1 אינה בהכרח נכונה עבור לוח m זוגי. היות ולפי ההוכחה, בסיס האינדקוציה עבור m זוגי אינו מתקיים. עבור לוח 2×2 כל לחיצה בודדת תוביל למשבצת בודדת שתשאר במצבה ההתחלתי. לכן, כל לחיצה נוספת תשנה את שאר הנורות והפתרון שיתקבל אינו אופטימלי. משום כך לא ניתן להוכיח בעזרת טענה 5.1 שקיים פתרון אופטימלי ללוחות עם m זוגי.

איור 5.3 : פתרון ללוח 2×9



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

הגדרה 5.2: לוח אינוספי של משחק מקיים את התכונות הבאות:

- נגדיר משבצת $a_{1,1}$ כראשית הצירים.
- בלוח זה הצירים ממשיכים אינסוף ימינה ולמטה.
- את הלוח אפשר לתאר בעזרת מטריצה אינסופית.

משפט 5.2: עבור לוח אינסופי, אין פתרון אופטימלי בו המשבצת בראשית הצירים נלחצה.

הוכחה. לאחר לחיצה על $a_{1,1}$ מתקבל הלוח הבאה:

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

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:

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

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 5.2 אם ברצוננו למצוא פתרון אופטימלי ללוח האינסופי 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.

משפט 5.3: אין אף פתרון אופטימלי בו נלחצה משבצת $a_{1,2}$

הוכחה. כדומה להוכחה טענה 5.2 נגיע למבוי סתום. אם נסתכל על מצב הלוח לאחר לחיצה על משבצת $a_{1,2}$ ניראה כי יש סידרת לחיצות מאולצות, כדי להדליק נורות מסוימות. אם נתחיל את המשחק על ידי לחיצה על $a_{1,2}$ נקבל את הלוח הבאה:

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

כדי להדליק את הנורות $a_{2,1}$ נהיה חייבים ללחוץ על $a_{3,1}$. כדי להדליק את הנורות $a_{2,3}$ נהיה חייבים ללחוץ על $a_{2,4}$. כדי להדליק את הנורות $a_{3,3}$ נהיה חייבים ללחוץ על $a_{4,3}$. כדי להדליק את הנורות $a_{5,2}$ נהיה חייבים ללחוץ על $a_{6,2}$. נציג את רצף הלחיצות על הלוח, נראה את המשבצות שנלחצו ב $*$:

$$\begin{bmatrix} 1 & * & 1 & 1 & 0 \\ 1 & 1 & 1 & * & 1 \\ * & 1 & 1 & 1 & 0 \\ 1 & 1 & * & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & * & 1 & 0 & 0 \end{bmatrix} \tag{8}$$

נשים לב שאת נורה $a_{5,1}$ לא ניתן להדליק ללא כיבוי משבצות השכנות, לכן הגענו למבוי סתום. $\square$

אפשר להחליף בין הצירים ולקבל בעזרת אותה הוכחה למה לא קיים פתרון בו נלחצת בהתחלה משבצת $a_{2,1}$.

משפט 5.4: לא קיים פתרון אופטימלי ללוח אינסופי.

הוכחה. היות ועברנו על כל האפשרויות שאפשר כדי לנסות להדליק את $a_{1,1}$, והראינו שבכל מצב מגיעים למבוי סתום. ניתן לומר שלא קיים פתרון אופטימלי ללוח אינסופי. $\square$

מסקנה 5.1: למשחק על לוח 4×4 קיים פתרון אופטימלי. נתבונן על לוח 8. ניתן לראות שקיים פתרון עבור לוח 4×4.

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

מסקנה 5.2: למשחק $4 \times n$ לא קיים פתרון אופטימלי כאשר $n > 4$

עבור לוח $4 \times n$, טענה 5.2 מתקיימת היות וההוכחה נכונה עבור לוח $4 \times n$.

נרצה להראות שגם טענה 5.3 נכונה עבור לוח $4 \times n$. אם נתחיל את המשחק בלחיצה על $a_{2,1}$ בדומה למה שעשינו

כדי לקבל את לוח 8, נקבל את סידרת לחיצות המאולצת $\{a_{2,1}, a_{1,3}, a_{3,4}, a_{4,2}, a_{6,3}\}$ שמתוארת בלוח הבאה:

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

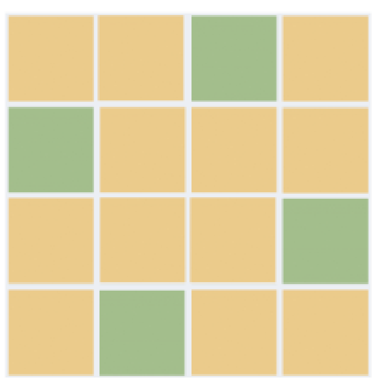
הגענו שוב למבוי סתום.

מסקנה 5.3: במשחק על לוח $m \times n$ שמקיים $\min(m,n) > 4$ אין פתרון אופטימלי.

5.2 אלגוריתם למציאת פתרון אופטימלי

קיימים מספר דרכים למציאת פתרון אופטימלי. אפשר לנסות ולחפש פתרון ידנית, אפשר לעבור על כל הפתרונות של משחק רגיל ולבדוק עם יש בינהם פתרון אופטימלי. נציעה דרך אחרת לחפש פתרון אופטימלי בעזרת שימוש במטריצה שכנויות כפי שהגדרנו בהגדרה 3.3, ההבדל הוא שהפעם נגדיר את המטריצה על החוג $\mathbb{Z}$. שימוש בחוג $\mathbb{Z}$ מאלץ את הפתרונות המתקבלים להדליק את כל נורות רק פעם אחת. זאת מכייון שמשוואות האילוצים שהגדרנו ב 3.6 מאלצות את הסכום להיות שווה לאחד. התיאוריה שפיתחנו באלגברה לינארית הייתה תקפה לשדות אבל כלי התכנות שהשתמשנו בעבודה זו יודע לפתור גם על חוג השלמים. בדקנו את הכלי במקרים שהממדים שייכם לקבוצה $\{(m,n) : 2 < m,n < 7\}$ וקיבלנו שהלוח היחיד בקבוצת הממדים שיש לו פתרון אופטימלי הוא לוח 4×4. פתרון מתואר באיור 5.4.

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)
```

```python
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 לשלושה פונקציות: פעולת דירוג של מטריצה שאלגוריתם מתאר, פונקציה שפותרת את המערכת המשוואות ופונקציה שעוטפת את שני הפונקציה וזהו המימוש שצריך לקרואו המשתמש. הפתרון מוצג כוקטור שורה כפי שתיארנו ב 6.2.

[3]:
```
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)
```

```python
    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 $n \times m$ כאשר $m, n \leq 9$. הפלט שמתקבל הוא טבלה, שהשורות והעמודות מתארות את מימדי הלוח. לדוגמה אפשר לראות מטבלת התוצאות שללוח 3×5 כמות 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 $n \times m$. 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
```

```python
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 LIGHTS 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.

[4] אברהם ברמן, בן-ציון קון, אלגברה ליניארית, תיאוריה ותרגילים , הוצאת בק, חיפה, 1999.