המחלקה למתמטיקה שימושית

חקירת משחק האורות

מאת :

ולדיסלב ברקנס

מנחה :

אלכס גולוורד

14 בפברואר 2022

תוכן העניינים

1 הקדמה

עבודה זה הינה עבדות סוף של סטודנט במחלקה למתמטיקה שימושית. עבודה זה מבוססת על משחק האורות ונבנתה על גבי שאלות שנשאלו במהלך חקירת המשחק. חיפוש פתרונות הוביל למחקר ותוצאות מעניינות שלא ברורות מעליהן.

השאלות לדוגמה שעלו בעבודה הן שיטות למציאת פתרון למשחק. נציג שתי שיטות, שבדיעבד נראו שונות אבל הצלחנו להראות את הקשר בשתי השיטות.

בנוסף לאחר שנמצא אלגוריתם שפותר את המשחק שמנו לב לתופעה מעניינת כאשר המשחק האורות מתחיל כאשר כל הנורות דלוקות אז קיים לפחות פתרון אחד, תופעה מעניינת שכזה העסיקה רבות את פרויקט זה ומצאנו הוכחה מדוע תופעה זה מתקיימת.

דבר מרכזי נוסף שעסקנו בו הוא בחיפוש סוג מסוים של פתרונות, פתרונות מינמלי שנגדיר בעבודה. סוג הפתרונות שכזה כל כך לא נפוץ שהצלחנו להוכיח את כל המקרים בהם אתכן פתרון שכזה.

עבודה סוף זה הייתה מהנה עבורי אני מודה למחלקה למתמטיקה שימושית
במיוחד לאלכס גולוורד על הזדמנות לעשות עבודה מרתקת שכזה.
תודה רבה

2 תאור של המשחק

משחק האורות או Lights Out בלועזית, זהו משחק על לוח משבצות מלבני. כל משבצת יכולה להיות באחד משני מצבים, נקרא להם דלוק וכבוי. כאשר משתמשים בשמות האלה מתכוונים שבכל משבצת יש נורה והיא יכולה להיות דולקת 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 דולקת לכבות אותה ואם היא כבויה להדליק אותה. לוח בקרה בנוי כך שכאשר מתבצעת לחיצה על משבצת אז מצב של נורה משתנה ומשתנה גם מצב של נורות סמוכות לה. שתי נורות נקראות סמוכות אם הן נמצאות במשבצות בעלות צלע משותפת. המטרה של המשחק היא לעבור ממצב התחלתי למצב בו כל הנורות יהיו דולקות.

הערה 2.1 בחירת מצב התחלתי להיות דלוק או כבוי אינה תשנה את המשחק.

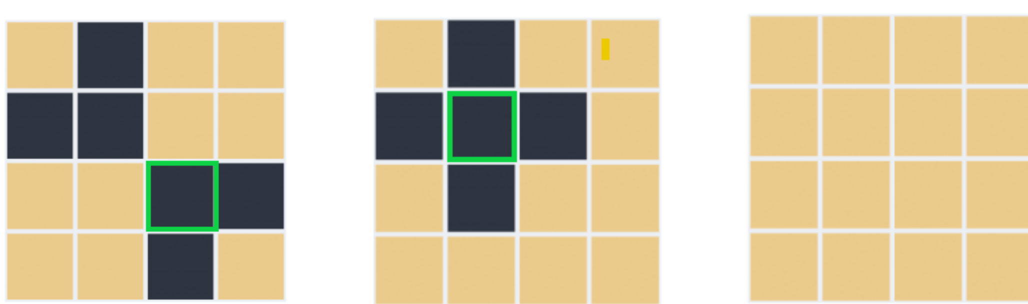
הערה מדגישה כי כל המטרה של המשחק היא לעבור ממצב מסוים בו נמצאים כל הנורות למצב אחר. איך ניקרא למצב או איך שהוא יראה בפועל לא משנה את אופי המשחק אבל מה שחשוב שיש שני מצבים התחלתי וסופי.

2.1 תאור גרפי של המשחק

נתאר את המשחק באיור כאשר המצב התחלתי של משחק שכל נורות צהובות, ומצב הסופי היה שכל נורות שחורות. נסמן את המשבצת שנלחצה בגבולות ירוקים.

איור 1: הסבר שינוי מצב הלוח לאחר לחיצה

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



פירוט: נבחין כי המשחק על לוח 4×4 המצב התחלתי מתואר באיור 1א במצב של הלוח כל הנורות צהובות. לאחר ביצוע לחיצה על משבצת שמסומנת בירוק נעבור ללוח שמתואר באיור 1ב. נתאר לחיצה נוספת באיור 1ג.

אחרי שהסברנו על כללי משחק והצגנו הדגמה קטנה הדרך הטובה ביותר לוודא הבנה היא בלשחק, 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.

אתגר שכל שחקן חווה היא שאין אסטרטגיה גלויה למציאת פתרון, בפועל מנסים להגיע למצבים ידועים שמהם אתה מכיר את איך לפתרו את המשחק. לכן כל פעם שמנסים משחק על לוח בממדים חדשים, חוויה המשחק מתחדשת כאילו ולא שיחקת במשחק מעולם.

2.2 סוגיות בהן נעסוק בפרויקט

1. תיאור ודיון בשני אלגוריתמים למציאת פתרון המשחק.
2. הוכחות לקיום פתרון המשחק לכל לוח $m \times n$.
3. הרחבה של משחק מלוח משבצות למשחק על גרף.
4. חיפוש לוחות של משחקים בהם מתקיים פתרון כזה שנורות שינו את מצבן רק פעם אחת בלבד.
5. נתן התייחסות למספר הפתרון האפשריים בלוח ונדבר על חסם מספר הפתרונות על לוח בממדים כלשהם

קיימים המון שאלות שקשורות למשחק וננסה בפרויקט זה להציג פתרון לחלקם. נרצה בפרויקט זה להציג תופעות מעניינות במשחק, ולהראות שהמשחק אינו רק מהנה עלה גם אתגר מתמטי לא קטן.

2.3 תיאור משחק על גרף

אחרי שכללי המשחק על לוח הובנו אפשר לנסות להכליל את המשחק כמשחק 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. הבדל הוא שבמשחק על גרף הצמתים מקבלים את התפקיד של הלחצנים, לאומת אותם המשבצות שהיו במשחק על לוח. נזכיר שכל לחיצה על צומת הופכת את המצב של אותה הצומת והשכנים שלה, כאשר המשחק הוא על גרף נומר שצמתים שכנים אם קיימת קשת שמחברת ביניהם. נציין כי כאשר כל צומת יכולה להיות בשתי מצבים, דלוקה או כבויה. המטרה במשחק לעבור מגרף הצמתים במצב מסוים נגיד דלוק למצב אחר כבוי.

נמחיש זאת על דוגמה שבאיור 2 כאשר הגרף התחלתי 2א ניתן לראות 6 קודקודיים צבועים באפור כלומר כבויים ומטרה של המשחק להדליק את כל הצמתים כלומר לצבוע את כולם בצהוב. בשלב 2ב צומת שנלחצה נצבע בירוקה היא ושכניה נדלקות ונצבעות בצהוב.

הערה 2.2 בפועל צומת ירוקה גם נצבעת לצהוב צביעה לירוק נועדה להדגשה על מי בוצע הלחיצה.

איור 2: משחק על גרף לדוגמה

(א) מצב התחלתי

(ב) לחיצה על משבצת מסומנת

2.4 תרגום משחק על לוח למשחק על גרף

לאחר שתיארנו כיצד לשחק את המשחק על גרף ניתן לומר ששחק על לוח הוא סוג של משחק על גרף כלומר, כל משחק לוח ניתן לתאר בעזרת משחק על גרף. נרצה בתת פרק לחדד זאת.

כדי לתאר את הלוח על משחק על גרף נשתמש בשני הכללים הבאים:

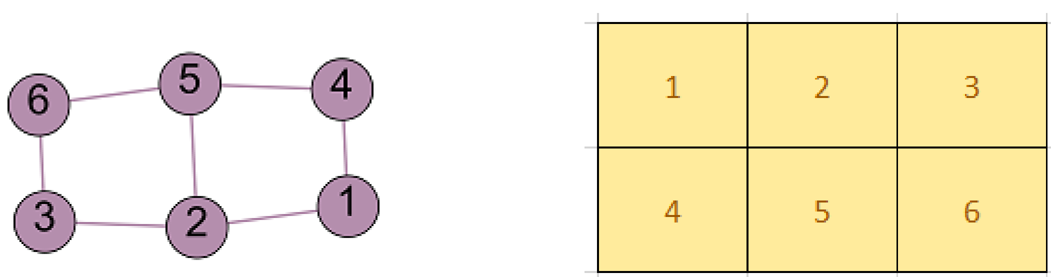
1. כל משבצת על משחק לוח נהפוך לצומת.
2. כל זוג משבצות סמוכות על לוח נחבר את הצמתים בצלע

נמחיש זאת על דוגמה, ניקח לוח למשל 2×3 נמספר את המשבצות כמו באיור 3א. הגרף שנקבל עבור לוח באיור 3א מתואר באיור 3ב

איור 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 הרבה משחקים שמתוארים על גרף אבל לא ניתן לתאר אותם על לוח לדוגמה, גרף בו יש צומת אם יותר מ 4 שכנים לא ניתן לתאר לוח שכזה כיוון שלכל משבצת על לוח יש לכל יותר 4 משבצות סמוכות.

הערה 2.4 בעזרת שיטה שתיארנו אפשר להפוך כל משחק לוח למשחק על גרף, אבל להפך הוא לא נכון כלומר לא כל משחק על גרף אפשר להפוך למשחק על לוח.

בגלל שכל משחק לוח ניתן לתאר אותו כמשחק על גרף לכן המשפטים המרכזיים ננסה לנסח על משחקים על גרף כי אז הם היו נכונים גם על משחקים על לוח.

3 אלגוריתם למציאת פתרון

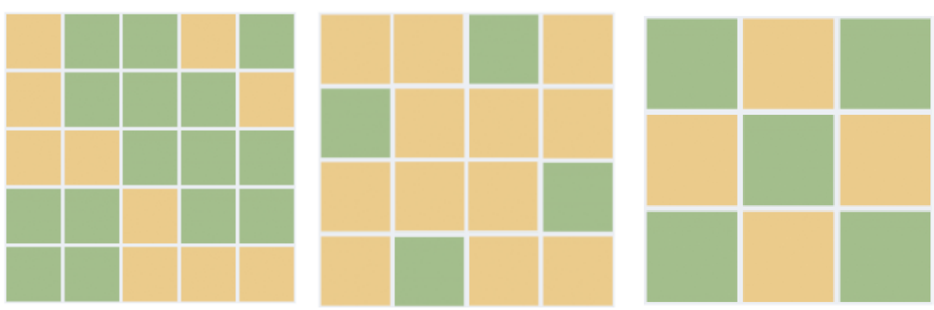
לפני שנציג את שיטות למציאת פתרון, נרצה להמחיש את האתגר במשחק על ידי הצגה כמה תופעות שקוראות במשחק.

באיור 4 מוצגים כמה פתרונות אפשריים ללוחות שונים, כאשר לחיצה על הלחצנים ירוקים בסדר כלשהו תוביל לפתרון המשחק. אפשר לראות שכמות הלחיצות שצריך לפתרון על לוח 4×4 דורשת פחות לחיצות מהלוח 3. ההנחה הגיונית שאפשר היה לחשוב היא שככל שהלוח גדול יותר נדרש יותר לחיצות כדי להגיע לפתרון, אבל כפי שרואים באיור 4 זה לא נכון.

תופעה נוסף שקוראת במשחק היא שכמות הפתרונות משתנה. עבור לוח 3×3 קיים פתרון יחיד, אבל ללוח 4×4 קיים 16 פתרונות. בהפתעה רבה ללוח 5×5 יש רק 4. עובדה זה שללוח 5×5 יש פחות פתרונות מלוח 4×4 מפתיע כי אפשר היה לצפות שלוח יותר גדול אז כמות הפתרונות תגדל.

אפשר לחדד את חוסר הבנה לכמות הפתרונות אם ניקח לדוגמה לוחות ריבועים כלומר $n\times n$ כמות הפתרונות כל כך לא צפויה שאם נשאל את עצמנו, מה הלוח אם הכי הרבה פתרונות וכמה פתרונות יש ללוח כאשר $n \in [1, 20]$? נקבל שמספר הפתרונות הגדול ביותר הוא על לוח $n = 19$ ומספר פתרונות 65536. בנוסף $n = 19$ הוא הלוח היחיד ב $n \in [1, 20]$ שמקבל כמות הפתרונות שכזה. לאומת זאת מספר הפתרונות השני הגדול ביותר הוא רק 256 ומתקיים ל $n \in \{9, 16\}$.

איור 4: פתרונות של משחק על לוחות שונים



שתי השיטות למציאת פתרון שנציג בעבודה היו מבוססות על מידול הבעיה לשדה לינארי ולמערכת משוואות שפתרון שלה יוביל לפתרון של המשחק עצמו. אתכן ויש כמה דרכים להגיע לאותו מודלו לינארי שנציע, נציג בעבודה זה שני דרכים. כשנתאר את שתי שיטות הן יראו שונות בתכלית אבל, היופי הוא שאפשר להראות ששתי השיטות מובילות לאותה מערכת משוואות כלומר צורת הפתרון המתוחכמת יותר נעזרת בצורה של הלוח כדי לפתור ביעילות גבוה יותר את הבעיה.

3.1 שיטת פתרון ראשונה בעזרת מטריצת שכנויות

כדי למדל את הבעיה על שדה לינארי נזכר בייצוג גרפי שאומר כי לחיצה על צומת משנה את הצומת ושכניה אם נסמן את צמתים ב n_i אז משלב זה נתאר את המשחק בצורה נוחה יותר לתיאור אלגברי.

1. כל צומת יכול להיות בשתי מצבים, את המצבים נסמן, $0, 1$
2. מצב של צומת i נסמן ב n_i.
3. המצב התחלתי של משחק על גרף הוא שכל צמתים אם הערך התחלתי שהוא 0.
4. מצב סופי של משחק על גרף הוא שכל צמתים אם הערך הסופי שהוא 1.

הערה 3.1 פעולת לחיצה על לחצן משנה את מצב מנורה, שינוי מצב מנורה ניתן בעזרת חיבור בשדה $\mathbb{Z}_2$ עם הערך 1. אם מצב צומת במצב התחלתי לאחר לחיצה תעבור למצב סופי, $0 + 1 = 1$. אם מצב צומת במצב הסופי לאחר לחיצה תעבור למצב התחלתי, $1 + 1 = 0$.

תיאור של משחק שכזה מאפשרת לנו לתאר המשחק בצורה וקוטרית. אם ניקח לדוגמה משחק בגודל 2×2 נמספר את שורות ואז עמודות מלמעלה למטה כמו שמתואר באיור 5. נוכל לתאר את הלוח שכזה במצבו התחלתי כמטריצה

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

ואם נרצה לתאר את לוח ממצב התחלתי לאחר לחיצה על משבצת 1.

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \xrightarrow{n_1} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

כפי שתיארנו בהערה 3.1 אפשר לתאר שינוי מצב הנורה על ידי חיבור 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 ווקטור קואורדינטות בסיס סטנדרטי של מרחב מטריצות אז נוכל לרשום את השוויון הנ״ל גם כך

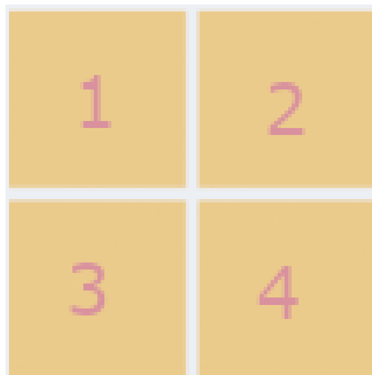
$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

וקטור שחיברנו עם מצב הלוח התחלתי הוא וקטור שמתאר את הלחיצה ונקראה לו בעבודה זה וקטור שינוי.

הגדרה 3.1 תהי משחק על גרף בעל n צמתים ממספרים מ 1 עד n, וקטור שינוי t_i של צומת i הוא וקטור שייך ${\mathbb{Z}_2}^n$, שאם תחבר אותו אם מצב הלוח הנוכחי התוצאה המתקבלת תהיה מצב הלוח לאחר לחיצה על צומת i.

כדי לבנות וקטור שינו של צומת i נשים ערך 1 בכל אינדקסים בהם האינדקס שווה למספר של צומת שכנה לצומת i ובאינדקס של צומת עצמה כלומר, באינדקס i. שאר הערכי הוקטור הם אפס.

איור 5 : מספור לוח



הערה 3.2 מספור לוח שעובר על שורות ואז עמודות מלמעלה למטה כמו שמתואר באיור 5, היה שיטת המספור הקבוע בפרויקט זה עבור משחקים על לוח.

ניקח דוגמה קצת יותר מסובכת עבור גרף באיור 6. באיורים עלו נצבע בכחול צמתים שמצבם 1 ובאדום צמתים שמצבם 0. בעזרת וקטור השינוי אפשר לתאר תוצאה של מספר לחיצות, נעשה זאת בעזרת חיבור וקטור שינויים וחיבור מצב הגרף. התוצאה שנקבל תהיה הגרף המתקבל לאחר לחיצה של צמתים הללו. נדגים רעיון זה כאשר מצב של גרף מתואר באיור 6א. נניח שצומת 1 היא יחידה שדלוקה. נרצה להראות איך הגרף יראה אם ילחצו על כפותרים $1, 3$. וקטור שינוי של צומת 1 הוא

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

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

ומצב התחלתי שמתואר באיור נסמן ב S_0 לכן מתקבל

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

הגרף המתקבל לאחר חיבור אכן תואם לתוצאה המצופה מתואר באיור 6ב.

איור 6: דוגמה לתיאור וקטור שינוי במהלך משחק על גרף

(א) מצב של הגרף לפני לחיצה

(ב) מצב של הגרף לפני לחיצה

הערה 3.3 היות ווקטור שינוי שדה ${\mathbb{Z}_2}^n$ חיבור בין וקטורים הינו חיבור בין האינדקסים מודולו 2 וכפל בסקלר הוא לכפול את כל ערכי וקטור בסקלר כאשר הסקלרים יכולים להיות 0 או 1

הערה 3.4 מספר זוגי של לחיצות אינו משנה את מצב הלוח

היות ואנחנו עובדים על שדה מודולו 2 $t_i + t_i = \vec{0}$.

הערה 3.5 לא משנה כמה תלחץ על לחצן בודד הלחצן יכול להעביר אותך לשני מצבים.

מספר הלחיצות על אותו לחצן אינו משנה לחצן עכשיו לחוץ אם נלחץ מספר אי זוגי של פעמים כי מספר לחצות הזוגיות לא שינו את הלוח. בנוסף זה מסביר את הסיבה למה כפל בסקלר שאנחנו מוכנים לקבל הוא הערכים $0, 1$. לכן בהמשך הפתרון יתואר אם יש צורך ללחוץ בלחצן או לא, לא תהיה התייחסות לכמות הלחיצות כי התוצאה מתקבלת רק תלויה בזוגיות של מספר הלחיצות.

הערה 3.6 היות ומצב התחלתי הוא שכל הצמתים במצב 0 לכן, ניתן לתאר את המצב עליו נעבור רק בעזרת צירוף לינארי של וקיטורי שינוי בלבד.

היות ומצב התחלתי נסמן כרגע ב S_0 הוא כולו וקטור האפס 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 לצורה הבאה :

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

כאשר $\vec{1}$ וקטור שכל ערכיו אחדים, שזהו מצב הסופי של הגרף ו n מספר הצמתים בגרף. נשים לב שאם ידוע צירוף $x = \begin{bmatrix} x_1, & x_2, & \cdots, x_n \end{bmatrix}$ שמקיים את המשוואה 2 קיבלנו פתרון של משחק על גרף. כדי להגיע לפתרון על גרף נלחץ על הצמתים שמספרם שווה לאינדקסים שמקיימים $x_j = 1$ בצירוף x.

בנוסחה 1 קיימים מספר תכונות שנרצה לציין.

למה 3.1 סדר הלחיצות לא משנה את התוצאה הסופית

בגלל אסוציאטיביות של חיבור בשדה סדר לחיצות לא משנה.

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

לפי הערה 3.5 כל לחצן יכול להיות בשתי מצבים והיות לפי למה 3.1 סדר הלחיצות לא משנה לכן ללוח $m \times n$ מספר אפשרויות לחיצה $2^{m \cdot n}$. כדי להבין את עוצמה של מספר מסדר שכזה נסתכל במשחק על לוח 6×6 כמות אפשרויות לחיצה גדולה מכמות הסטנדרטית שמציגים מספר שלמים, 4 בתים כלומר המספר הגדול ביותר שאפשר להציג בעזרת 4 בתים הוא $2^{32} - 1$ המטרה של המחשה זה להדגיש כמה לא פרקטית לנסות לפתור בעזרת ניסיון כל האופציות האפשריות.

מערכת משוואות שמתוארת בנוסחה 2 אפשר לתאר במספר צורות. נפוצה מבניהם היא בעזרת מטריצה כמו שמתואר בנוסחה 3.

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

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

נשים לב שלמטריצה A במשוואה 3 מתקבל ש $A_{i,j} = 1$ כאשר i, j צמתים שהם שכנים או זהים $i = j$.

הגדרה 3.2 מטריצה שמתארת את משחק שקבלנו במשוואה 3 תקראה מטריצת שכנויות של משחק.

הערה 3.7 היות וכל צומת שכנה היא שכנה אחד לשני לכן במטריצה סימטרית

דוגמה למטריצה המתקבלת מגרף באיור 6ב:

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

הגדרה 3.3 תהי משחק ומערכת משוואת שמתארת אותו מצורה 3 נגדיר את וקטור פתרון כצירוף הערכים מביאים לפתרון של מערכת המשוואות ונסמן אותו ב $\vec{x}$

נזכיר שאם $\vec{x}$ וקטור פתרון של המערכת ו $x_i = 1$ אז המשמעות שכדי לפתור את המשחק צריך ללחוץ על לחצן i. בנוסף נזכיר שאתכן שהיו כמה פתרונות אפשריים.

הגדרה 3.4 שיטת פתרון בעזרת יצירת מטריצה שכנויות על ידי וקטור שינויים תקראה שיטה מציאת פתרון לפי מטריצת שכנויות.

תוצאה דומה אפשר לראות בפרק מהספר [2]. מרגע שהצלחנו לתאר את הבעיה מערכת משוואות לינארית על שדה ${\mathbb{Z}_2}^n$ מכן נוכל להיעזר בכלים של אלגברה לינארית כדי למצוא את הפתרון כמו מציאת פתרון בעזרת דירוג, מציאת מטריצה פסאודו הפוכה וכולי.

הערה 3.8 כמות תאים במטריצה שכנויות שצריך ללוח ריבועי $[n \times n]$ הוא n^4.

עבור לוח $[n \times n]$ כמות הלחצנים n^2 ולכן גודל המטריצה שתוארה במשוואה 3 היא $[n^2 \times n^2]$ כמות הזיכרון שצריך כדי לשמור מטריצה ללוח ריבועי שאורך צלע הוא n דורש לשמור מטריצה עם n^4 ערכים.

3.2 שיטה למציאת פתרון לפי שורה העליונה

הצגנו גישה פתרון בעזרת מטריצת שכנויות, כפי שתיארנו בהגדרה 3.4. נרצה להראות שיטה נוספת למציאת פתרון. שיטת הפתרון שנציג נובעת מהערה 3.8 שכמות המידע שצריך לשמור גדל בקצב n^4,
כאשר משחק הוא על לוח ריבועי ש n מייצג כמות המשבצות לשורה.

המאמר [1] מציג שיטה שמציאה של פתרון עם מערכת משוואות לינארית שונה ממערכת שהצגנו קודם. אומנם המערכות שונות אבל שני המערכות המשוואות מובילות לאותם פתרונות.

מערכת המשוואות המתקבלת בשיטה במאמר [1] על לוח $n \times n$ היא $n \times n$ כלומר, יש n^2 ערכים במטריצה. צמצום כמות הערכים שכזה יכולה להוביל לחישוב מהיר יותר וניצול טוב יותר של מידע.

בפרק זה נציג את הגישה שמתוארת [1] נתאר כמה הבחנות שיסתמכו ששני הגישות שקולות ושגישה החדשה הינה רק אופטימיזציה ספציפית לצורה של מטריצה הנתונה. את הגישה החדשה ניקרא לאורך כל הפרק שיטה למציאת פתרון לפי שורה העליונה.

הגדרה 3.5 שיטה למציאת פתרון לפי שורה העליונה, היא שיטה שמבוססת על עיקרון שאם, ידוע איזה כפתורים צריכים להילחץ בשורה העליונה כדי להגיע לפתרון, אפשר לגלות את כל שאר הכפתורים שצריכים להילחץ כמעט מידית.

נתאר את שיטה למציאת פתרון לפי שורה העליונה כל שלב שנעשה על לוח 3×3. נניח שאיור 7 מתאר את הלוח הנתון כאשר המשבצות הינן הלחצנים ומספור זה אינדקסים הממספרים לפי הערה 3.2. שיטה למציאת פתרון לפי שורה העליונה מתבסס על רעיון, שפתרון של המשחק הוא סדרה של לחיצות על משבצות מסוימות. נשייך לכל משבצת משתנה שיכול לקבל שני ערכים: 0 אם משבצת הזאת מופיעה בסדרת לחיצות של פתרון ו-1 אם משבצת הזאת כן מופיעה בסדרת לחיצות של פתרון המשחק. מראש לא ידוע לנו האם משבצות של שורה ראשונה יופיעו בסדרה הזאת או לא. אז משתנים של משבצות בשורה ראשונה הם x_1, x_2, x_3.

איור 7: לוח 3×3 עם אינדקסים

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

על מנת שהנורה במשבצת ראשונה בשורה ראשונה תהיה דולקת סכום משתנה שלה ומשתנים של משבצות שכנות במודולו 2 חייב להיות 1.

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

לכן משתנה במשבצת ראשונה בשורה שניה, שסימנו במשתנה x_4 חייב להיות שווה ל $x_1 + x_2 + 1$ כי

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

על מנת שהנורה במשבצת שנייה בשורה ראשונה כלומר, x_5 מדוגמה תהיה דולקת סכום משתנה שלה ומשתנים של משבצות שכנות במודולו 2 חייב להיות 1. לכן משתנה שחייבים לשייך למשבצת שנייה בשורה שניה חייב להיות $x_1 + x_2 + x_3 + 1$ כי

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

באופן דומה מחשבים ערכי משתנים של שאר המשבצות בשורה שנייה ואחר כך על פי אותם שיקולים ערכי משתנים של משבצות בשורות הבאות.

הגדרה 3.6 המשוואה של סכום משתנים של משבצת i וכל שכניה תקראה משוואת אילוצים על לחצן i

משוואה 4 הינה משוואת האילוצים שללחצן 4. עבור משחק לוח ריבועי באורך שורה n שהלחצנים ממספרים לפי הערה 3.2 ניתן לנסח בנוסחה פשוטה:

$$x^*_{i-n} + x^*_{i-1} + x^*_i + x^*_{i+1} + x^*_{i+n} = 1 \quad x^*_i = \begin{cases} x_i & \text{if } i \in [1, n^2] \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

נשים לב שבעזרת גישה שתיארנו כרגע נוכל למלאה כל השורה שניה. כל שורה תלויה בשורה מלפניך לכן כך נוכל למלאה את כל השורות כמו שמתואר באיור 8. נדגים מילוי משבצת 7 משורה שלישית לכן נצטרך ששורה שניה חושבה. נסתכל על משבצת מעל כלומר משבצת 4 ונסתכל למה שווה משוואת האילוצים שלה:

$$x_0 + x_3 + x_4 + x_6 = 1$$

לכן

$$x_6 = 1 + x_0 + x_3 + x_4$$

היות ושורה שניה מולאה וידוע שערך משבצות באותה שורה:

$$x_3 = 1 + x_0 + x_1$$
$$x_4 = 1 + x_0 + x_1 + x_2$$

נציב ערכים אילו

$$x_6 = 1 + x_0 + (1 + x_0 + x_1) + (1 + x_0 + x_1 + x_2)$$
$$x_6 = 1 + x_0 + x_2$$

כדומה נעשה לשאר הערכים. התוצאה מתקבלת מתוארת באיור 8 נבחין שלאחר שמילאנו את כל הלוח כמו

איור 8: לוח 3×3 מלאה

x0	x1	x2
1 + x0 + x1	1 + x0 + x1 + x2	1+ x1 + x2
1 + x0 + x2	0	1 + x0 + x2

איור 9: לוח 3×3 מלאה כולל שורה וירטואלית

x0	x1	x2
1 + x0 + x1	1 + x0 + x1 + x2	1+ x1 + x2
1 + x0 + x2	0	1 + x0 + x2
1 + x1 + x2	x0 +x1 + x2	1 + x0 + x1

שמתואר באיור 8. לו הינו יודעים את את הערכים של משתנים בשורה העליונה ביותר הינו כבר פותרים את המשחק. על מנת ליצור מערכת משוואות שתפתור את המשחק מחבר המאמר [1] מוסיף לשורה האחרונה עוד שורה, שורה דמיונית ומחשב ערכי משתנים של המשבצות שלה לפי אותם שיקולים.

משום שזאת שורה דמיונית, בעצם אנחנו בפועל לא מדליקים אף נורה בה ערכי המשתנים של משבצות שלה חייב להיות אפס. כך נוצרת מערכת משוואות עם n משוואות ו- n משתנים וזה ההסבר שנתן מחבר המאמר. בהמשך כאשר נסביר קשר בין שיטה הזאת ושיטה הקודמת ניתן הסבר אחר. מערכת המשוואות המתקבלת עבור הדוגמה 3×3.

שיטה שתיארנו ביצע מעבר על שורות אפשר היה לעשות בניה דומה גם לעמודות. המאמר [1] מתאר מספר רב של פתרונות בלוחות ריבועים בגדלים שונה ואפילו על לוחות מלבניים. האתגר המרכזי בשיטה הספרדית היא להצדיק אותה למה יש שורה וירטואלית והאם יש קשר בין שני השיטות. בשלב זה נתרכז להראות את הקשר

איור 10 : מערכת המשוואות המתקבלת משיטה פתרון לפי שורה העליונה בלוח 3×3

$$\begin{cases} 1 + x_1 + x_2 = 0 \\ x_0 + x_1 + x_2 = 0 \\ 1 + x_0 + x_1 = 0 \end{cases}$$

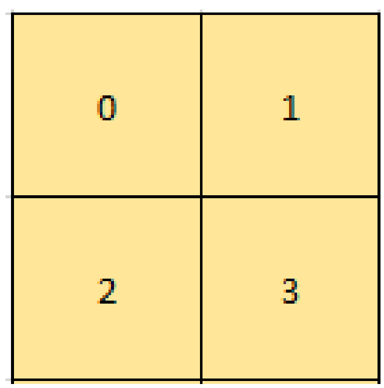
בין שיטה הספרדית ושיטה שהצגנו בפרק הקודם.

משפט 3.1 מטריצה המיצג של מערכת המשוואות האילוצים היא מטריצת שכנויות 3.

משוואות האילוצים פורמלית היא לב השיטה הספרדית מכיוון שהתקדמות בשורות מבוססת על המשוואות עלו. אם נפרוס את משוואת האילוצים נקבל גם מערכת משוואת שפותרת את המשחק אבל כמות המשוואות הינה n^2. נבחין שאם נציג אותם כמטריצה כאשר כל משוואת אילוצים מסודר לפי סדר הלחצנים נקבל את מטריצה שכנויות.

נדגים זאת על לוח 2×2 שמתואר באיור 11.

איור 11 : לוח 2×2



וקטור השינויים :

$$t_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}, t_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}, t_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}, t_3 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

איור 12 : מערכת משוואות מורחבת של משחק על לוח 3×3

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

לכן מטריצת שכנויות נראת 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 את המערכת מהצורה

$$x_0 + x_1 + x_2 = 1$$
$$x_0 + x_1 + x_3 = 1$$
$$x_0 + x_2 + x_3 = 1$$
$$x_1 + x_2 + x_3 = 1$$

אפשר בקלות לראות שאם נבנה את המטריצה המייצגת של המערכת נקבל מטריצה דומה למטריצת השכנויות.

תעלומה נוספת בשיטה הספרדית היא למה צריך שורה וירטואלי, למה ערכה שווה ל 0, וכיצד באמת מתבצע צמצום כמות המשתנים ל n משתנים. הסבר לתופעה זה ניתן בעזרת תיאור שיטת מציאת פתרון לפי שורה העליונה רק שהפעם את הפעולות במקום לעשות על טבלה שתיארה את הלוח נבצע על המטריצה שמתארת את מערכת המשוואות , מטריצת השכנויות. נראה את הפעולות שעושים בשיטה על אותה דוגמה על לוח 3×3

נשים לב ששיטה הספרדית מדרג את המטריצה מורחבת $[M|\vec{1}]$.

כדי לחשב את x_4 באיור 7 השתמשנו במשוואת האילוצים של משבצת 1 שזה בדיוק תיאור שורה הראשונה של מטריצה מורחבת בשורה הראשונה שמתוארת באיור 12.

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

נבחין שלושת השורות הראשונות מאפשרות תיאור פשוט של המשתנים x_4, x_5, x_6. אם ננסה לתאר את משתנה x_7 באותה שיטה נסתכל על שורה ה 4 במטריצה וניראה שהיא תלויה ב x_4, x_5 היות ואמרנו שאפשר בקלות לתאר את משתנים עלו בעזרת שורות $1, 2$ במטריצה לכן נעזר בשורות עלו כדי לתאר את x_7, אופן שימוש בשורות עלו תהיה הפעלה פעולה שורות הבאה:

$$r_4 \leftarrow r_4 + r_1$$

$$r_4 \leftarrow r_4 + r_2$$

שני פעולות שורות הללו הם שקולות לפעולה אלגברית הבאה :

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

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

ועכשיו קיבלנו תיאור של x_7 בעזרת המשתנים של שורה העליונה. כך נמשיך באופן דומה לאופן שמילנו משבצות לפי שורה העליונה כך ניתן להמשיך ולמלאה את השורות במטריצה על ידי פעולות שורות המתבססות על חישוב קודם. אחרי שנעבור על כל השורות בצורה שכזה נקבל את המטריצה: נשים לב שבמטריצה 13 שלושת השורות התחתונות מתוארת אך ורק על ידי המשתנים x_1, x_2, x_3. אם נתאר את שורות עלו כמערכת משוואות נקבל את אותה מערכת משוואת של שיטה למציאת פתרון לפי שורה העליונה על לוח 3×3 שתיארנו במערכת המשוואות 10.

3.3 השוואה בין שתי השיטות למציאת פתרון

לפי חישוב סיבוכיות לדרג מטריצה כללית בגודל $n^2 \times n^2$ זה $O(n^2 \cdot n^4) = O(n^6)$.

דירוג בעזרת שיטה הספרדית אומרת שעל כל עמודה וקטור עמודה של מטריצת שכנויות יש לכל יותר 5 ערכים ששווים 1. כל החוכמה בדירוג בשיטה הספרדית היא שפעולות השורות הם על משתנים שכבר דורגו לכן כמות הפעולות שורות לא משתנה. לכן דירוג שורה היה חיבור של עד כ 5 שורות לכן הסיבוכיות $= O(n^2 \cdot n^2)$ $O(n^4)$.

לדרג את n משתנים הנותרים הוא בסיבוכיות $O(n \cdot n^2) = O(n^3)$.

ננסה להראות זאת בפועל על ידי חישוב זמני חישוב.

איור 13 : המטריצה לאחר פעולת שורות על כל שורות

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

באיור 14 אפשר לראות ביצועים של שני האלגוריתמים ציר ה x גודל שורה של לוח המלבני הרצנו על לוח בגדלים מ 10 עד 60 משבצות. ציר ה y זמן שלקח בשניות
לפי התוצאות של איור 14 נראה שגישה הספרדית שבתאוריה יותר אופטימליות לוקחת יותר זמן. אחת הסיבות לקח שפונקציה שפותרת מערכת משוואות הינה פונקציה של ספירה שנעזרתי וכנראה יש מימוש אופטימלי לפתרון הבעיה שאפילו ששיטה הספרדית מקטינה את כמות המשתנים היא אינה יכולה להתחרות במימוש אופטימלי שממשה בספריה.

3.4 דיון לגבי משחק על גרף

קיימים הרבה סיבות בהם תירצה להגדיר את הבעיה על מבנה כללי שכזה :

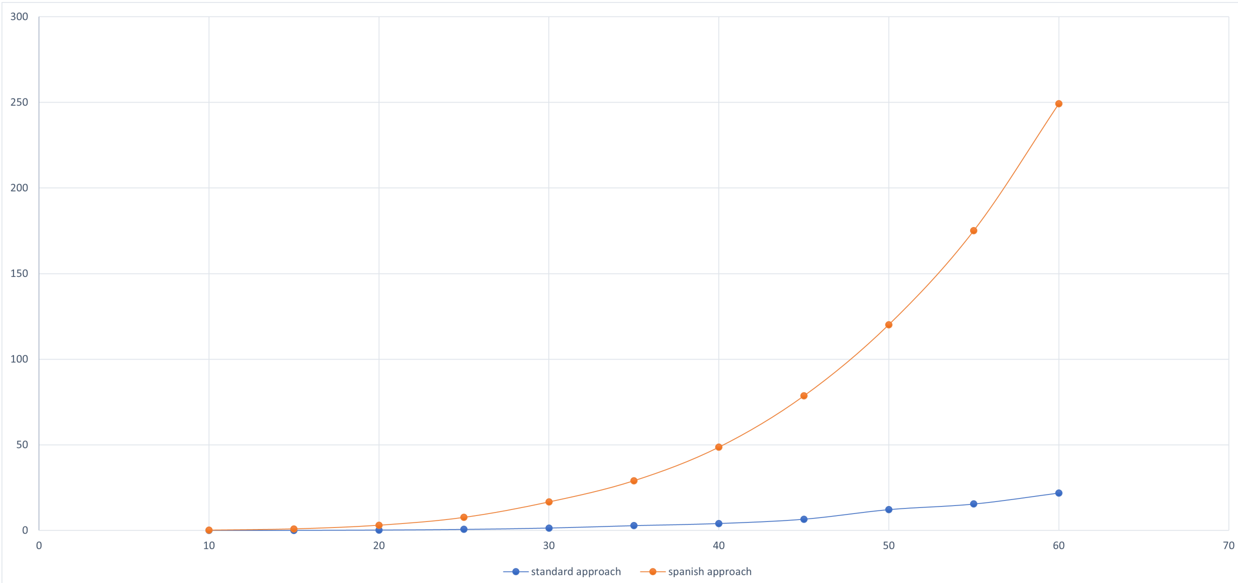
1. ככול שמבנה כללי יותר תאוריה שאתה מפתח מתאימה ליותר בעיות.

2. קיימת תאוריה רחבה שפותחה על גרפים ואתכן שנעזר בה.

3. מבליט את מהות הבעיה והגדרה הבסיסית ביותר של המשחק.

ארצה להתייחס לנקודה אחרונה, תיאור הבעיה של משחק כאוסף של כללים על גרף.
הקשר בין המשחק עצמו לתיאור לגרפי כל כך מהותי שבשלב מסוים של הפתרון נקבל את אחת הצורות לייצג גרפים וזאת על ידי מטריצת שכנויות.

איור 14 : גרף מתאר ביצועים על לוח ריבועי גודל שורה מול זמן



4 הוכחת קיום פתרון עבור כל גרף

עד כה הסתכלנו על שני גישות שונות למציאת פתרון אבל שאלה טבעית לשאול היא האם בכלל קיים פתרון למשחק על לוח כלשהו? אחת הדרכים לענות על שאלה שכזה היא פשוט לקחת את הלוח ולפתור בעזרת דרכי הפתרון שהצגנו. אחת הבעיות בגישה של לחפש פתרון על לוח כלשהו היא נניח ואנחנו רוצים לממש את המשחק האורות שמיצר לוחות אקראיים, היות ולא בהכרח ידוע אם קיים פתרון נצטרך לבדה שקיים פתרון על כל לוח בעזרת אלגוריתמים למציאת פתרון שלוקח זמן אתכן והלוח ללא פתרון נצטרך לחפש לוח אקראי אחר מה שיגרום לתהליך יצירת משחק להיות איטי. לכן, נרצה בשיטה מתמטית להוכיח לעבור איזה משחקים יש פתרון. בנוסף שאלה נוספת שאפשר לשאול היא כמה פתרונות יש ללוח. מספר הפתרונות של הלוח יכול להעיד האם הלוח יותר קל או קשה לשחקן שמנסה לפתור אותו לבד ללא אלגוריתם. בפרק זה נענה על השאלות הללו.
אחד המקומות ששאלה זה נשאלה היא בספר [3], בעבודתנו נראה הוכחה קצת שונה בעזרת הכלים שפיתחנו.

הגדרה 4.1 נגדיר פעולה בין שני וקטורים ב ${\mathbb{Z}_2}^n$ ניקרא לה מכפלה סקלרית תסומן $x \cdot y$ ונגדיר אותה כך :
תהי $\vec{x}, \vec{y} \in {\mathbb{Z}_2}^n$ אז

$$\vec{x} \cdot \vec{y} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix} = x_1y_1 + x_2y_2 + \cdots x_ny_n$$

כאשר פעולת חיבור בין האיברים הינה חיבור מודולו 2

הערה 4.1 המכפלה הסקלרית שהגדרנו ב 4.1 אינה מכפלה פנימית,
היות ותכונה $\vec{u} = \vec{0} \Leftrightarrow <\vec{u}, \vec{u}>= 0$ לא מתקיימת.

דוגמה שמסבירה את הערה 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 נסמן זאת $\vec{y} \perp \vec{x}$ אם המכפלה הסקלרית שהלם שווה ל 0 $\vec{x} \cdot \vec{y} = 0$

משפט 4.1 תהי מטריצה $A \in Z_2{}^{m\times n}$ אז $ColA^T \perp NulA$ $ColA \perp NulA^T$

ידוע שתכונה זה מתקיימת ב מטריצה $A \in R^{m\times n}$ ההוכחה ל $Z_2{}^{m\times n}$ זה פרט ל למכפלה הפנימית שדורשת לבצע על תוצר ב $R^{m\times n}$ עוד מודלו 2. היות ותוצר של מכפלה פנימית הינו אפס גם לאחר מודלו 2 היה 0.

משפט 4.2 לכל משחק על גרף כאשר המצב התחלתי בו כל הנורות במצב 0
קיים פתרון למשחק.

לפי שיטת פתרון סטנדרטית שהגדרנו 3.4 ניתן לתאר את פתרון המשחק על גרף בעזרת מטריצה שכנויות לפי הגדרה 3.2.
מטריצה שכנויות נסמן ב $A \in Z_2{}^{n\times n}$.
נזכיר כמה תכונות חשובות

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

כדי להראות שלמשחק יש פתרון צריך להראות שקיי פתרון למערכת

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

במקרה ש A מטריצה הפיכה אז קיים פתרון יחיד. עבור המקרה שמטריצה אינה הפיכה כלומר $NulA \neq \{\vec{0}\}$ ניקח $\vec{x} \in NulA$ כלומר $A\vec{x} = \vec{0}$

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

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

$$\vec{x}^T A\vec{x} = a_{1,1}x_1^2 + 2(a_{1,2} + a_{2,1})x_1x_2 + \cdots 2(a_{1,n} + a_{n,1})x_1x_n + \\ + a_{2,2}x_2^2 + 2(a_{2,3} + a_{3,2})x_2x_3 + \cdots + 2(a_{2,n} + a_{n,2})x_2x_n + \cdots \tag{6}$$

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

$$a_{i,j} - a_{j,i} = a_{i,j} + \cdots + a_{j,i} = 1$$

נזכיר כי תוצאות של פעולת חיבור וחיסור מודלו 2 זהות.

לכן את המשוואה 6 אפשר לפשט ל

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

הבחנה נוספת לערך 0 או 1 $x^2 = x$ לכן פישוט נוסף למשוואה 6 אפשרי:

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

לכן קיבלנו $\vec{x}^T A\vec{x} = 0$ שמתקיים:

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

כלומר $\vec{x} \perp \vec{1}$ כאשר $x \in NulA$ לפי משפט 4.1 מתקבל $\vec{1} \in ColA^T$ היות ומטריצה סימטרית $A^T = A$ לכן $\vec{1} \in ColA$ והוכחנו שלמערכת $A\vec{x} = \vec{1}$ יש פתרון.

4.1 מספר הפתרונות עבור כל גרף

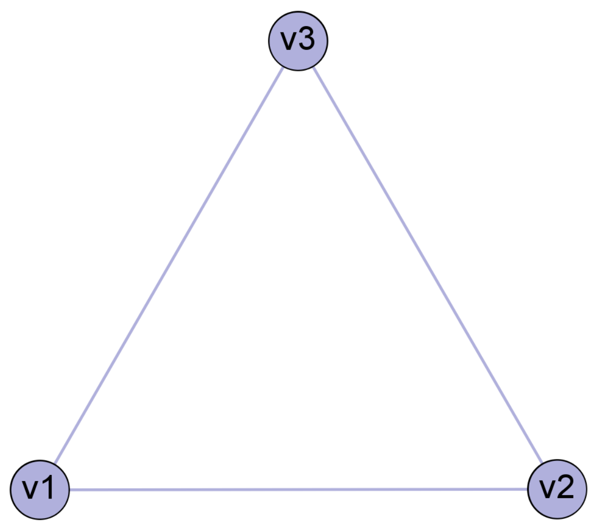
הוכחנו שלכל משחק על גרף שמתחל עם כל לחצנים במצב 0 יש פתרון ניזכר שסדר לחיצות אינו משנה את התוצאה על הלוח לכן אם נילחץ על הלחצנים בסדר כלשהו לפי פתרון נקבל גרף כולו דלוק.

השאלה שנשאל בפרק זה מה אפשר לומר על מספר פתרונות מפיתוח שעשינו. נציין קודם שניקרא לשני פתרונות שונים אם קיים לפחות לחצן אחד שמבדיל בין הפתרונות כלומר קיים לחצן ששייך לפתרון ראשון ולא שייך לפתרון שני כפי שציינו קודם סדר לחיצות לא משנה את הפתרון. לכן פתרון הינו קבוצה של לחצנים. בנוסף נזכר לפי הערה 3.5 מספר אי זוגי של לחיצות נחשב ללחיצה לכן מספר הלחיצות על אותו לחצן לא משנה אלה רק זוגיות של מספר לחיצות לכן לכל לחצן יש רק שני מצבים שיכול להיות לחוץ או לא. כרגע נראה שקיים כמה

פתרונות לדוגמא איור 15 המתאר משחק על גרף בו הצמתים כבויים. היות וגרף הינו קליקה לכן לחיצה בודדת על אחד הצמתים תדליק את כל הלחצנים.

קבלנו $G = \{\{v_1\}, \{v_2\}, \{v_3\}\}$ היא קבוצת של פתרונות. כלומר כבר הראינו שיש מקרים בהם יש יותר מפתרון אחד.

איור 15 : משחק על גרף



שאלה טביעת שנובעת כשגילנו שיש היא כמה פתרונות יש למשחק מסוים.

משפט 4.3 מספר הפתרונות של משחק שווה ל 2^k כאשר k שווה לדרגת החופש של מטריצה A של פתרון הסטנדרטי

היות לכל משחק ניתן להגיר מטריצת שכנויות של משחק שהגדרנו ב 3.2 ופתרונות של משחק וקטורים X של מערכת $AX = \vec{1}$ כאשר A מטריצת שכנויות. ידוע שקיים פתרון למשחק ואם הוא משחק שמתחיל שמצב כל הנורות הוא 0 אז יש משפט 4.2. שמוכיח שקיים פתרון.

היות ומניחים שיש כמה פתרונות אפשר לתאר את כל פתרונות כ $x = x_n + x_0$ כאשר $x_n \in Nul(A)$, x_0 פתרון פרטי שידוע שקיים ו x כל פתרונות הכללים.

לכן מספר פתרונות כללים שווה למספר פתרונות במרחב האפס. ידוע שמספר פתרונות במרחב האפס תלוי לדרגת החופש ולכן מספר הווקטורים שפורשים את מרחב האפס שווה לדרגת החופש שנסמן ב k. כמות הווקטורים במרחב זה שווה לכל וקטורים שניתן ליצור 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,

לכן כל הקונבנציות האפשריות 2^k ששווה לכמות הווקטורים במרחב האפס וכמות הפתרונות השונים של המשחק.

הבחנה נוספת ומעניינת שנרצה לציין היא בנושא חסם עליון לכמות הפתרונות. חסם עליון טריוויאלי לכמות המקסימלית של פתרונות היא 2^n פתרונות כאשר n שווה למספר הלחצנים כלומר לא יכול להיות יותר פתרונות מאשר כמות הלחיצות השונות האפשריות במשחק.

הערה 4.3 עבור משחק לוח מלבני בגודל $m \times n$ קיים לכל יותר 2^k כאשר $k = \min m, n$ פתרונות שונים

הערה זה נכונה לפי גישה פתרון הספרדית שהגדרנו 3.5 ניתן לתרגם את משחק ל k משוואות ש k יכול להיות מספר שורות או עמודות לכן ניקח את המספר הקטן יותר.

5 פתרון מינימלי עבור לוחות מלבניים

בפרק זה נציג פתרון לסוג מסוים של פתרונות שרצינו להציע. סוג זה של פתרונות מביאים רמז וניראה שמקלים את משחק. הקלה שכזאת על משחק אולי יכולה ליצור ביטחון לשחקנים חדשים וכמובן לאפיין תכונות לסוג של פתרון של כזה.

הגדרה 5.1 משחקים על לוח שקיים פתרון שלחצנים שינו את מצב רק פעם אחת. למשחקים כאלו נקראה משחק מנמליים.

באיור 16 ניתן דוגמא לפתרון מינמלי בלוח 3×2. כשלוחצים על לחצנים $2, 3$ על לוח כל נורות נדלקות ואף אחת מהם לא נכבה באף שלב של לחיצה.

השאלה שנפתור בפרק זה לאיזה לוחות קיים פתרון מינמלי כאשר מצב התחלתי הוא שכל הנורות במצב 0.

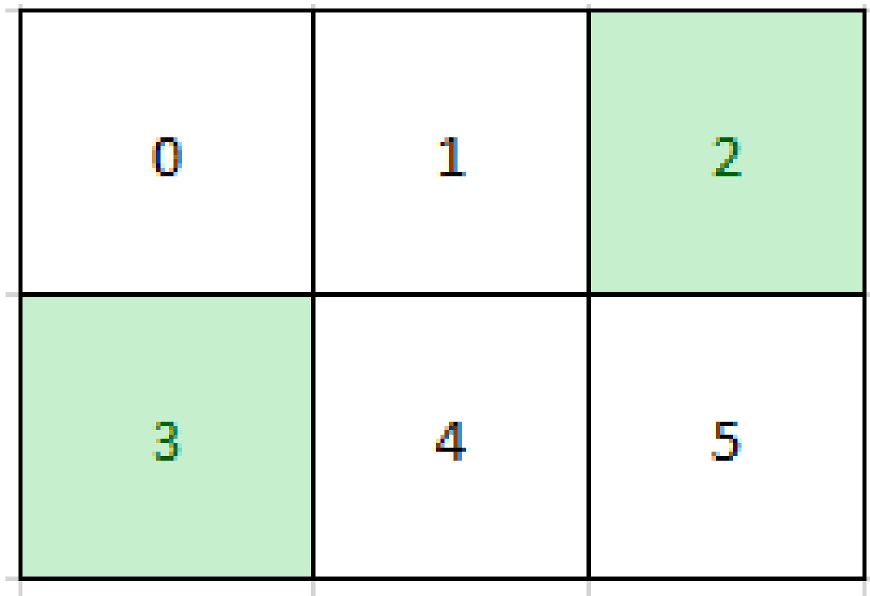
הגדרה 5.2 אזור מת זהו אוסף לחצנים בלוח שלחיצה עליהם גורמת לחצן שכבר השתנה בעבר להשתנות שוב

אזורים מתים על איור 16 נראה שלאחר לחיצה על לחצן 2 האזור מת שנוצר מלחיצה הינו $\{0, 1, 2, 4, 5\}$

הערה 5.1 אזור מת שנוצר מלחיצה הוא כל הלחצנים במרחק לכל יותר 2 משבצות מלחצן שנלחץ כאשר המרחק הוא מרחק מנהטן כלומר כל צדע למשבצת סמוכה למעלה למטה ימינה ושמאלה מגדילה את המרחק באחד

באיור 17 אפשר לראות שאם נלחץ על לחצן בצהוב האזור המת הי האזור באדום כולל הלחצן עצמו. תכונה זה כלי מרכזי בהוכחה במשפט הבאה

איור 16 : פתרון מינמלי של משחק



משפט 5.1 במשחק על לוח $m \times n$ שמתקיים $\min(m, n) >= 7$ למשחק אין פתרון מינמלי

נניח ויש לנו לוח דו ממדי שמתואר כך נקודת התחלה בכיוון למטה "קומת ראשונה" והולך כלפי מעלה לאינסוף ואינסוף לצד ימין וצד שמאל.

נקרה לכל שורה אינסופית קומה ונמספר אותם מאחת לאינסוף לכן קראנו לקומה נמוכה ביותר קומה ראשונה

נרצה למצוא פתרון מינמלי ללוח וגישה לחיפוש הפתרון תהיה להדליק שורה אחר שורה במלואה, היות ושורת אינסופיות נציעה אסטרטגיה להדלקת השורה וניראה את הקשיים שנפגוש.

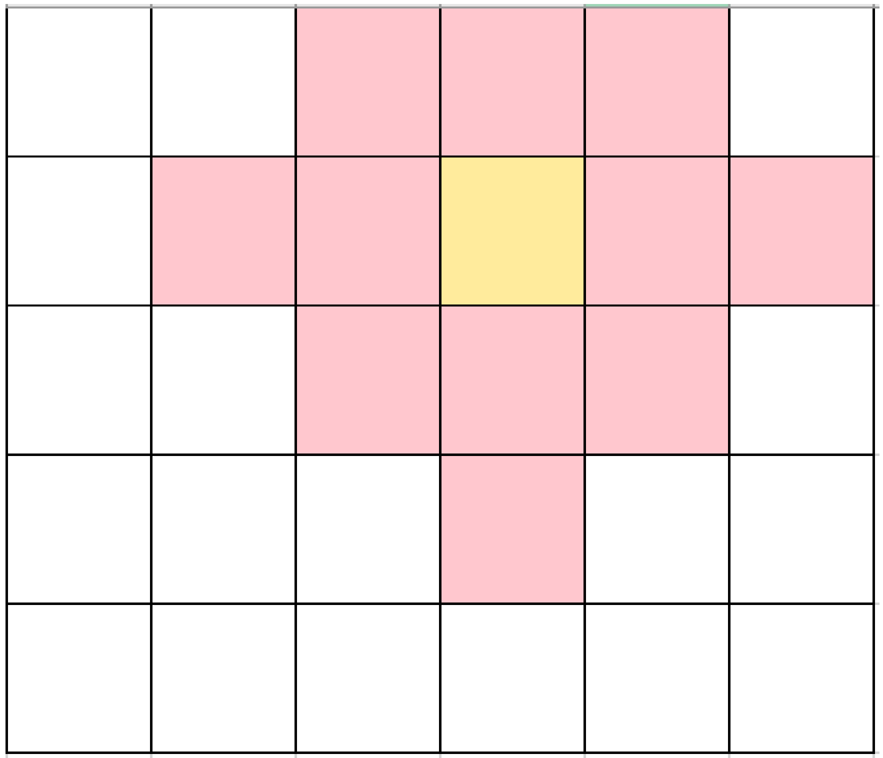
אם נרצה להדליק את כל קומה ראשנה רק על ידי לחצות בשורה ראשונה נקבל את הדפוס שאם לחצתי על לחצן מסוים חייב אני ללחוץ על לחצן 3 מימינו כמו שמתואר באיור 18 שמתאר לחצנים בירוק כלחצנים שנלחצו צהוב לחצנים שנדלקו ובאדום אזורים מתים שלא נדלקו. באיור מוצג רק שתי לחיצות עוקבות של אסטרטגיה זה אבל כך נדליק את השורה הראשונה.

נשים לב באיור 18 על שני אזורים המתים הצמודים שלא נדלקו שצמודים אחד לשני כדי להדליק את שינהם לא נוכל לעשות זאת ללא כיבוי לחצן שכבר נדלק. זאת אומר שאסטרטגיה שכזאת נפסלת עבור מילוי משחק שקומה שלו גדולה מ 1.

אסטרטגיה אחרת ויחידה למילוי קומה ראשונה הינה להדליק פעם לחצן בקומה ראשנה ופעם לחצן בקומה שניה צמודים. היות ורק שני קומות ראשונות משנות את מצב הלחצנים בקומה ראשונה ולא קיים דפוסים נוספים אפשריים למילוי שורה ראשונה בעזרת שני שורות עלו לכן עלו הן כל אסטרטגיות למילוי קומה ראשונה.

באיור 19 אפשר לראות הדגמה קטנה של אסטרטגיה שכזה.

איור 17 : אזור מת שנוצר מלחצן באמצע הלוח



נרצה להראות אסטרטגיה שכזה מובילה לאזורים מתים שלא ניתן למלאות כל עוד רוצים שהפתרון היה מינימלי. אם נסתכל באיור 20 ניראה שאזורים המתים ששלשת המשבצות הרצופות באדום לא ניתן היה למלאה אותם לכן צירוף כזה אין חוקי כלומר הראינו שלמשחק כפי שהגדרנו לא קיים בכלל פתרונות מינימליים.

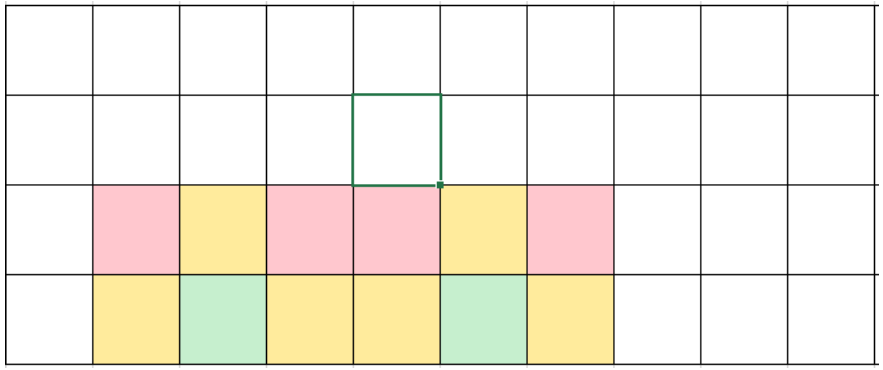
בשלב זה נרצה להקטין את הרוחב ואורך כך שאם קיים משחק אופטימלי בלוח המוקטן אסטרטגיות המילוי קומה קומה היחידות שהיו חוקיות הן עלו שהצגנו. נדע שהקטנה לא היו לה פתרונות אופטימליים אם היו משבצות סמוכות באזורים מתים.

נחזור ונסתכל על איור 18 נשים לב שלכל לוח שמספר המשבצות לרוחב גדול או שווה מ 6 שיטת המילוי שכזה תהיה לא חוקית כי היו 2 משבצות סמוכות שבאזורים לא חוקיים. ובאיור 20 ניראה שלרוחב גדול או שווה מ 5 היות ובניה של קומה ראשונה מסתמכת על זה שיש 7 משבצות ניקח ליתר ביטחון 7 משבצות ולכן באסטרטגיה זה לא היה פתרון מינמלי.

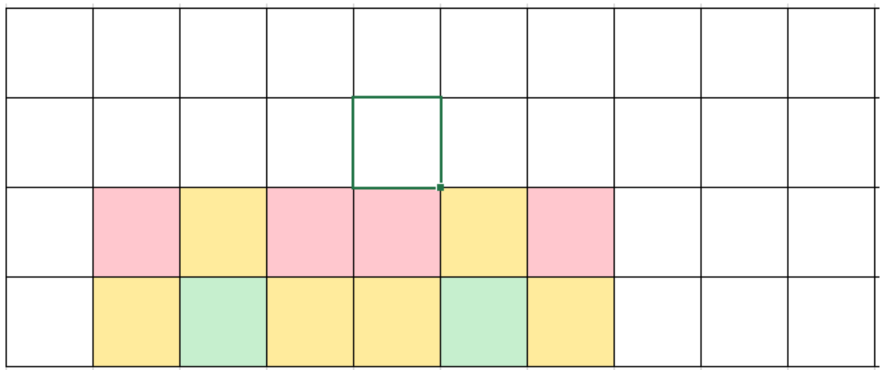
קיבלנו שאפשר להקטין את הלוח לרוחב של 7 משבצות ועדיין לא היה פתרון מינמלי.

את אותם טענות אפשר היה לבנות לא רק להגביל את רוחב ל 7 משבצות עלה גם לגובה. לכן לסיכום קיבלנו במשחק על לוח שאורך או רוחב גדולים או שווים מ 7 אז למשחק אין פתרון מינמלי והוכחנו את הטענה.

איור 18 : מילוי קומה ראשונה על ידי לחיצות רק בקומה ראשונה



איור 19 : מילוי קומה ראשונה



5.1 הלוח הגדול ביותר בעל פתרון מינמלי

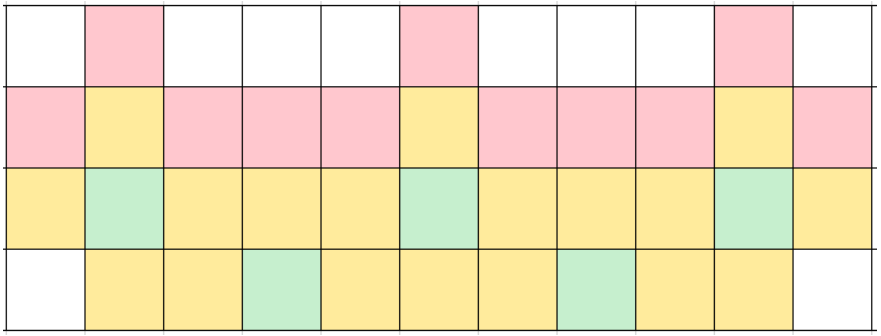
טענה 5.1 מגבילה מאד את המשחקים שיש להם פתרון מינמלי ובשיטת הפתרון שהצגנו אחד המסקנות המתקבלות שאם יש שלוש קומות או יותר מתחילה להיות בעיתיות בגישת מילוי השורות. אפשר להבחין בתופעה זה היות וקיים פתרון מינמלי למשחק $m \times 2$ כאשר m הוא אי זוגי האסטרטגיה השנייה מאפשרת מילוי קומות ולקבל פתרון מינמלי נדגים זאת על באיור 21

לאחר שהבנו שלוחות בגודל $m \times 2$ כאשר m אי זוגי קיים פתרון מינמלי נשאל מהו הלוח הגדול ביותר בעל פתרון מינמלי כאשר הלוח אורך ורוחב גדולים מ 2.

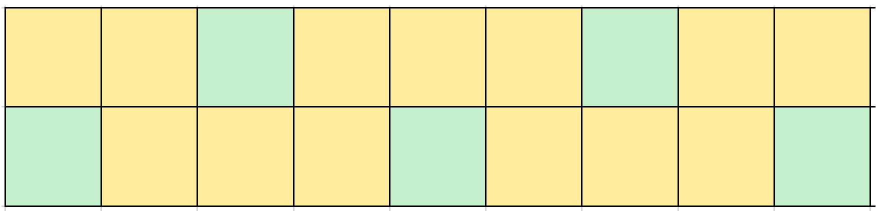
לפי טענה 5.1 אין טעם לבדוק לוחות שעמודות ושורות גדולים מ 7 ומבניית ההוכחה אמרנו שאם הגדול של עמודות או שורות לא קטן מ 3 כלומר נותר לבדוק לוחות שממד שלהם $m \times n$ שייכים לקבוצה $\{(m,n) : 2 < m,n < 7\}$.

אפשר לנסות ולחפש פתרון ידנית או לעבור על כל הפתרונות של משחק רגיל ולבדוק עם יש מבניהם פתרון מינמלי. נציעה דרך אחרת לחפש פתרון מינמלי והיא בעזרת להשתמש באותה מטריצה שכנויות כפי שהגדרנו

איור 20 : מילוי קומה ראשונה



איור 21 : פתרון ללוח 2×9



רק להגדיר את זה שהיא על חוג $\mathbb{Z}$. בעזרת שימוש בחוג $\mathbb{Z}$ מאלצים את שפתרונות המתקבלים שידליקו כל נורות אך ורק פעם אחת, זאת מתקיים בעקבות משוואות האילוצים שהגדרנו ב 3.6 שמאלצות את הסכום להיות שווה לאחד , אם נסתכל על נוסחה של משוואת האילוצים הכללים נוסחה 5 היות וחיבור על השלמים לכן מאולצים במשוואה זה שהיה לחצן בודד לחוץ לכן פתרון מערכת המשוואות מתאר פתרון מינמלי של משחק. התיאוריה שפיתחנו באלגברה לינארית הייתה תקפה לשדות אבל כלי תכנות שהשתמשנו בעבודה זה יודע לפתור גם על חוג של השלמים והסמכנו על הכלי כדי לבדוק את המקרים שממדים שייכם לקבוצה $\{(m,n) : 2 < m,n < 7\}$ וקיבלנו שהלוח היחיד בקבוצת הממדים העלו שיש לו פתרון מינמלי הוא לוח 4×4 ופתרון מתואר באיור 22

6 נספחים

מימוש של הפרויקט בוצע על ידי שפת תוכנה Python עם הכלי .Sage

1 Generate Matrix

general method to generate a square matrix of square game

[1]:
```
import numpy as np
def genenerate_neighbord_matrix(n) -> np.array:
    mat = np.zeros((n**2, n**2), dtype= np.int8)

    # the general case
    for j in range(0, n**2):
        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 < n**2 :
            mat[j+n,j] = 1

    return mat
print(genenerate_neighbord_matrix(3))
```

```
[[1 1 0 1 0 0 0 0 0]
 [1 1 1 0 1 0 0 0 0]
 [0 1 1 0 0 1 0 0 0]
 [1 0 0 1 1 0 1 0 0]
 [0 1 0 1 1 1 0 1 0]
 [0 0 1 0 1 1 0 0 1]
 [0 0 0 1 0 0 1 1 0]
 [0 0 0 0 1 0 1 1 1]
 [0 0 0 0 0 1 0 1 1]]
```

2 Solving game

general method to how solve the game, by solving the matrix.

[2]:
```
from sage.all import *
n = 3
A = Matrix(Integers(2),genenerate_neighbord_matrix(n))
Y = vector([1 for x in range(n**2)])
Z = vector([0 for x in range(n**2)])
X = A.solve_right(Y)
print(X)
```

```
(1, 0, 1, 0, 1, 0, 1, 0, 1)
```

3 Spanish method

[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


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)

def generate_mat_spanish_alg(mat : np.array):
    n = int(sqrt(mat.shape[0]))
    end_state = np.ones(n**2)
    gaussian_elimination_spanish_alg(mat, end_state)
    # the matrix we need to solve
    new_mat = np.array(mat[n**2-n:n**2, 0:n], copy=True)
    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]
    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(mat)
print(res)

print('check solution:')
X = vector(Integers(2),res)
Y = A*X
print(Y)
```

```
[[1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
 [1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0]
 [1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
 [0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0]
 [1 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0]
 [1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0]
 [1 1 1 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
[0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1]
check solution:
(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
```

4 Minimal case

generate matrix for rectengle game.
searching for integer solution.

[4]:
```
import numpy as np

# to prove the minimal case on not square we need to build matrix for not␣
↪rectangler board
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 < n**2 :
            mat[j+n,j] = 1

    return mat
print(genenerate_neighbord_matrix_m_n(3,2))
```

```
[[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 0 0 1 1]
 [0 0 0 0 1 1]]
```

[5]:
```
from sage.all import *
n = m = 4
a = genenerate_neighbord_matrix_m_n(m,n)
print(a)
A = Matrix(Integers(),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(X)
```

```
[[1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
```

```
 [1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0]
 [1 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0]
 [0 1 0 0 1 1 1 0 0 1 0 0 0 0 0 0]
 [0 0 1 0 0 1 1 1 0 0 1 0 0 0 0 0]
 [0 0 0 1 0 0 1 1 0 0 0 1 0 0 0 0]
 [0 0 0 0 1 0 0 0 1 1 0 0 1 0 0 0]
 [0 0 0 0 0 1 0 0 1 1 1 0 0 1 0 0]
 [0 0 0 0 0 0 1 0 0 1 1 1 0 0 1 0]
 [0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1]
 [0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1]]
(0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0)
```

5 Solution Amount

[6]:
```
n = 9
a = genenerate_neighbord_matrix(n)
A = Matrix(Integers(2),a)
print(2**A.kernel().dimension())
```

```
256
```

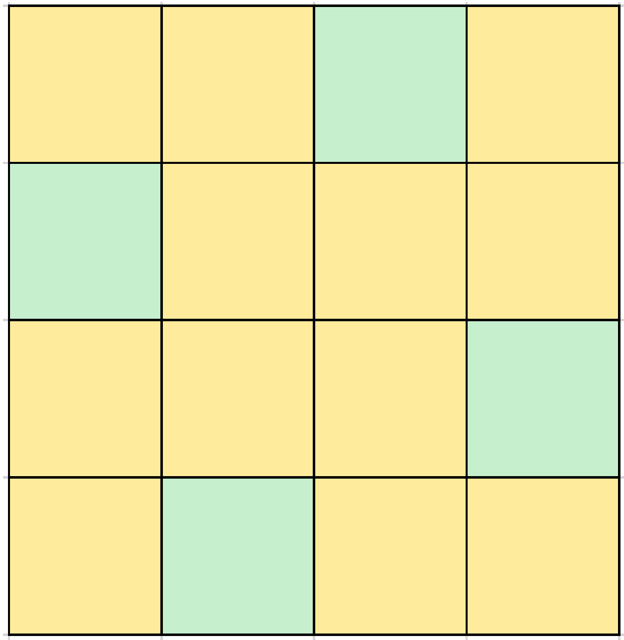
6 Benchmark

[7]:
```
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)
```

```python
    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(res)
```

```
[[10.         0.020791   0.184697]
 [11.         0.029358   0.261447]
 [12.         0.0316     0.366729]
 [13.         0.045727   0.51665 ]
 [14.         0.068553   0.670478]]
```

איור 22 : פתרון ללוח 4×4



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