אורט בראודה
ORT Braude College

המחלקה למתמטיקה שימושית

חקירת משחק האורות

מאת:

ולדיסלב ברקנס

מנחה:

אלכס גולוורד

8 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

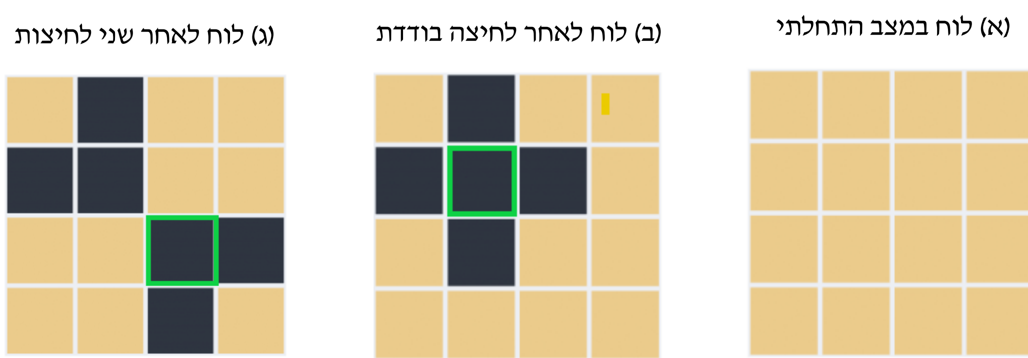
2 תאור של המשחק

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.

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



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`.

קושי המרכזי במשחק מניסיון האישי שלי שאין ממש כיוון שאתה מנסה לפיו להגיע לפתרון, בפועל מנסים להגיע למצבים ידועים. שמהם אתה מכיר את איך לפתרו את המשחק. בהתחלה מצבים ידועים עלה הם תור לפני פתרון וכל משחק שאתה פותר מגדיל לך את כמות המצבים המוכרים לשתי תורות לפני פתרון שלוש וכן

הלאה. לכן כל פעם שמנסים משחק על לוח בממדים חדשים נהיה חוויה חדשה של משחק ואתה לומד מחדש את המצבים שמהם אתה מכיר איך להגיע לפתרון.

פרויקט זה באה להציע שיטות לפתרון, נעזרנו במספר רב של כלים מתמטיים מתקדמים כמו תורה על שדות מציאת פתרון על מערכות לינאריות מידול מתמטי ועוד.

קיימים המון שאלות שקשורות למשחק וננסה בפרויקט זה להציג פתרון לחלקם. חוץ מאתגר של המשחק עצמו קיים אתגר מתמטי שנרצה בפרויקט זה להציג.

2.2 משחק האורות על גרף

אחרי שכללי המשחק על לוח הובנו אפשר לנסות להכליל את המשחק כמשחק 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.

כדי לתאר את משחק האורות על גרף נשתמש באותם כללים שהגדרנו.
הבדל המרכזי במשחק על גרף הוא שהצמתים הם הלחצנים לאומת אותם המשבצות שהיו במשחק על לוח. נזכיר שכל לחיצה על צומת הופכת את המצב של אותה הצומת והשכנים שלה. נזכיר כי צמתים יקראו שכנים אם הם צמתים שיש קשת ביניהם בגרף.

נציין כי כאשר כל צומת יכולה להיות בשתי מצבים, דלוקה או כבויה המטרה היא לעבור מכל הצמתים במצב מסוים דלוק למצב אחר כבוי.

העובדה שמצב התחלתי הינו דלוק או כבוי אינה תשנה את המשחק עלה רק לאיזה מצב סופי צריך לעבור לכבוי או דלוק.

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 מציגים לחיצה על צומת ירוקה היא ושכניה נדלקות ונצבעות בצהוב.

הערה 2.1 בפועל צומת ירוקה גם נצבעת לצהוב צביעה לירוק נועדה להדגשה על מי בוצע הלחיצה

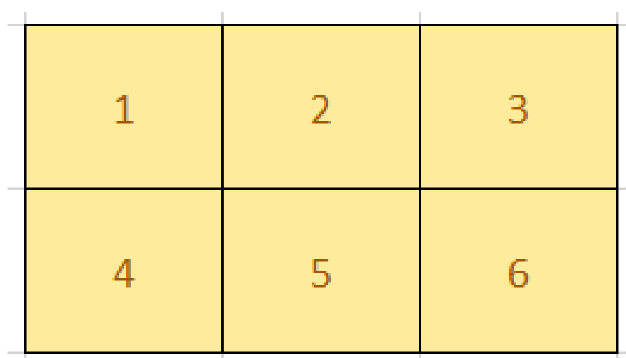
איור 2 : משחק על גרף לדוגמה

(א) מצב התחלתי   (ב) לחיצה על משבצת מסומנת

משחק על גרף הינה הכללה של משחק על לוח כלומר, כל משחק לוח ניתן לתאר בעזרת משחק על גרף.
נמחיש זאת על דוגמה, ניקח לוח למשל 3×2 נמספר את המשבצות כמו באיור 3
כדי לתאר את הלוח על על משחק גרף נשתמש בשני הכללים הבאים :

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

איור 3 : משחק על לוח 3×2 ממוספר

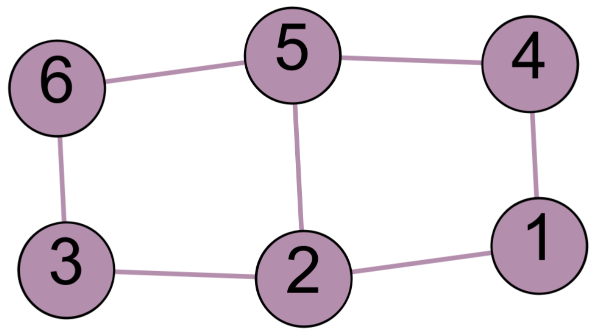


הגרף שנקבל עבור לוח באיור 3 מתואר באיור 4

הערה 2.2 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 2.3 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.

איור 4: משחק על גרף 2×3 ממוספר



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, נציג כמה סיבות מדוע אנחנו חיפשנו שיטה לפתרון. למשחק האורות אין בדיוק אינטואיציה אנושית כיצד לפתור אותו, אם תנסה בגישה חמדנית לכבות כמה שיותר מנורות מהר מאד תימצא את עצמך במבוי סתום. בנוסף, כאדם מאד קשה לנו לראות האם קיים פתרון בכמה מהלכים ספורים. לכן בדרך כלל מה שקורה שאתה מתבסס על לוחות שאתה יודע מהם להגיע לפתרון. אם ניקח לדוגמה לוח 3×3 המשחק ניראה תמים ופשוט כי הוא קטן מספיק כדי שתגיע למצב מוכר. בשלב שאתה מנסה לוח 4×4 הלוח מספיק גדול ומאתגר אבל הוא לאחר מספר משחקים נהיה מוכר ובאופן תמים תנסה לעבור ללוח 5×5 והוא פשוט מרגיש כלא פתיר. החלק הקשה במיוחד בלעבור ללוחות בגדלים שונים שידע שצברת בלוח מסוים אינו עובר ללוח אחר.

באיור 5 מוצגים כמה פתרונות אפשריים ללוחות שונים, כאשר לחיצה על הלחצנים ירוקים בסדר כלשהו תוביל לפתרון המשחק.

הבדל נוסף במשחק על לוחות בגדלים שונים הוא שכמות הפתרונות משתנה לכל לוח. 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 ביות הוא רק 256 ומתקיים ל $n \in \{9, 16\}$.

אפשרי לנסות לפתור בשיטה הנאיבית המנסה כל לחיצה אפשרית. הפתרון הנאיבי נפסל ברגע שחושבים על כמות האפשרויות לחיצה שנציג בלמה 3.3.

איור 5 : פתרונות של משחק על לוחות שונים

כדי לפתור את הבעיה ניעזר במידול מתמטי של הבעיה לבעיה על שדה לינארי. כשנצליח לתאר את הבעיה

כמערכת משוואות לינארית נוכל לפתור את המשחק ולהסביר על כמות פתרונות שונים שיש לבעיה.

אתכן ויש כמה דרכים להגיע לאותו מודל לינארי שנציע, נציג בעובדה זה שני דרכים. אחת בעזרת וקטור שינוי שנתאר בהמשך אותם. דרך שניה תהיה לפי תיאור משחק קצת בדרך שונה שתוביל למערכת משוואות. כשנתאר את שתי שיטות הן יראו שונות בתכלית אבל, היופי זה שאפשר להראות ששתי השיטות מובילות לאותה מערכת משוואות כלומר צורת הפתרון השני נעזרת בצורה של הלוח כדי לפתור ביעילות גבוה יותר את הבעיה.

בחרנו להציג קודם בעזרת וקטור שינוי משום שהגדרת וקטורים פשוטה יותר להסבר לאחר שניראה את הדרך הראשונה הדרך השנייה קלה תהיה יותר מובנת.

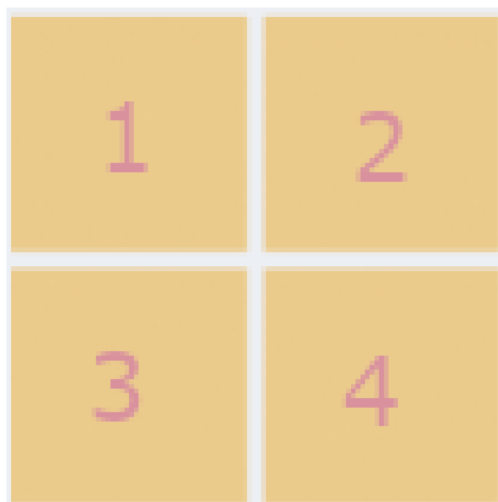
כדי למדל את הבעיה על שדה לינארי נזכר בייצוג גרפי שאומר כי לחיצה על צומת משנה את הצומת ושכניה אם נסמן את צמתים ב n_i אז משלב זה נתאר את המשחק בצורה הבאה.

הערה 3.1 המצב אתחלתי של משחק על גרף הוא שכל צומת אם הערך התחלתי $n_i = 0$. כל צומת יכול לקבל 2 ערכים שנסמן אותם ב $0, 1$ כאשר 0 מצב התחלתי של כל צמתים ו 1 מצב סופי של משחק. המשחק מסתיים כשכל הצמתים במצב $n_i = 1$. בגלל שהלחצנים יכולים להיות במצב $0, 1$ אנחנו עובדים על שדה בינארי שנסמן Z_2.

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 $Z_2{}^n$ כך שערכים בווקטור שערכם שווה ל $t_{i,j} = 1$ הם עם אותם צמתים עם מספור j שלחיצה על צומת i תשנה את מצבם. צמתים שישנו את מצב שסימנו את המספור שלהם j הם השכנים של צומת וצומת עצמה. שאר הערכי הוקטור הם אפס.

לדוגמה ניקח משחק בגודל 2×2 נמספר את הצמתים שורות ואז עמודות מלמעלה למטה כמו שמתואר באיור 6

איור 6 : מספור לוח



הערה 3.2 מספור שעובר על שורות ואז עמודות מלמעלה למטה כמו שמתואר באיור 6, היה שיטת המספור הקבוע בפרויקט זה עבור משחקים על לוח.

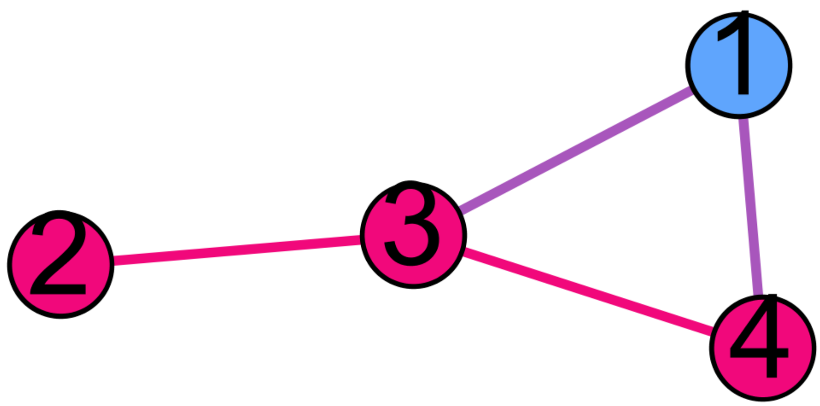
לאחר מספור שכזה נוכל לומר שלחיצה על משבצת 1 וקטור שינוי שלה נסמן ב $t_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$

כדומה עבור גרף באיור 7 מתקבל וקטור שינוי של צומת 1 הוא $t_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

הערה 3.3 היות ווקטור שינוי שדה $Z_2{}^n$ חיבור בין וקטורים הינו חיבור בין האינדקסים מודול 2 וכפל בסקלר הוא לכפול את כל ערכי וקטור בסקלר כאשר הסקלרים יכולים להיות 0 או 1

בעזרת וקטור השינוי אפשר לתאר תוצאה של מספר לחיצות, נעשה זאת בעזרת צירוף לינארי של וקטור שינויים. את המצב המתקבל נוסיף למצב הקיים ונקבל את השינוי שנוצר בלחיצה של כפתורים עלו. נדגים רעיון על איור 7 .

איור 7 : מצב התחלתי של הגרף



נניח שצומת 1 היא יחידה שדלוקה. נרצה להראות איך הגרף יראה אם ילחצו על כפותרים $1, 3$.

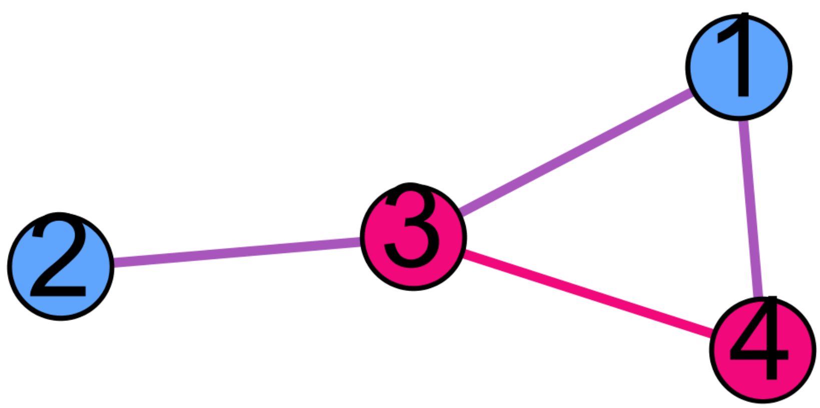
$$t_1 + t_3 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

ומצב התחלתי שמתואר באיור נסמן ב S_0 לכן מתקבל

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

וקטור התוצאה שהתקבל אכן תואם לתוצאה המצופה מתואר באיור 8.

איור 8: מצב לאחר ביצוע הלחיצות



למה 3.1 מספר זוגי של לחיצות אינו משנה את מצב הלוח

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

הערה 3.4 כל לחצן יכול להיות בשתי מצבים בלבד לחוץ או לא.

היות ולפי למה 3.1 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.

לכן בהמשך שנציג את הפתרון נסמן לחצות על לחצן i ב x_i אז $x_i = 1$ יסמן את המצב שלוחצים על הלחצן אבל לא ישנה כמה פעמים נלחץ הלחצן כל עוד הוא נלחץ מספר אי זוגי של פעמים.

נשים לב שעבור איך שהגדרנו את המשחק Light out המשחק מתחיל כשכול נורות דלוקות או כבויות ומטרה היא לכבות או להדליק את כל נורות כלומר לעבור ממצב דלוק לכבוי או ההפך.

אין באמת משמעות בין עם להתחיל את המשחק שכל הלחצנים דלוקים ולנסות לכבות אותם או ההפך ההבדל רק מה המשמעות שנותנים לערכים $0, 1$.

מפרק זה נגדיר שאנחנו פותרים את המשחק שלוח התחלתי כולו באפסים ומטרה להגיע ללוח שכולו אחדים.

מתיאור שכזה מובן כי S_0 זהו וקטור אפסים לכן כדי לתאר את ממצב התחלתי למצב לצירוף לינארי של וקטור שינוי אין צורך לחבר בין מצב התחלתי וצירוף לינארי היות ומצב התחלתי הוא כולו וקטור אפס מתקיים:

$$S_0 + \sum_{j=1}^{n} a_i\vec{t_j} = \sum_{j=1}^{n} a_j\vec{t_j} \qquad (1)$$

בעקבות כך ניתן לתאר את בעיית המשחק לצורה הבאה:

$$\sum_{j=1}^{n} a_j\vec{t_j} = \vec{1} \qquad (2)$$

כאשר $\vec{1}$ וקטור שכל ערכיו אחדים ו n מספר הצמתים בגרף.

תיאור שכזה מדגיש מספר תכונות

למה 3.2 סדר הלחיצות לא משנה את התוצאה הסופית

בגלל אסוציאטיביות של חיבור בשדה סדר לחיצות לא משנה. נשים לב שאם ידוע קבוצת לחיצות ממצב התחלתי הערך של משבצת מסוימת נקבע לפי הערך של וקטור תוצאה בנוסחה 1.

למה 3.3 כמות האפשרויות לחיצה על לוח $m \times n$ הוא $2^{m \cdot n}$

לפי הערה 3.4 כל לחצן יכול להיות בשתי מצבים והיות לפי למה 3.2 סדר הלחיצות לא משנה לכן ללוח $m \times n$ מספר אפשרויות לחיצה $2^{m \cdot n}$.

כבר בלוח $6x6$ כמות אפשרויות לחיצה גדולה מכמות הסטנדרטית שמציגים מספר שלמים, 4 בתים או 2^{32} מספרים, המטרה של המחשה זה להדגיש כמה לא פרקטית אופציית הפתרון שכזה.

מערכת משוואות שמתוארת בנוסחה 2 אפשר לתאר במספר צורות.

נפוצה מבניהם היא בעזרת מטריצה כמו שמתואר בנוסחה 3.

$$\begin{bmatrix} t_1 & t_2 & \cdots & t_n \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

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

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

הגדרה 3.2 מטריצה שמתארת את משחק תקראה מטריצת שכנויות של משחק.

הערה 3.5 היות וכל צומת שכנה היא שכנה אחד לשני לכן במטריצה סימטרית

המטריצה המתקבלת מגרף באיור 8 :

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

הגדרה 3.3 תהי משחק ומערכת משוואת שמתארת אותו מצורה 3 נגדיר את וקטור פתרון כאוסף הלחצנים ומצב להגיע לפתרון ונסמן אותו ב $\vec{x}$

הגדרה 3.3 מתכוונת שאם מתקבל $\vec{x}$ וקטור פתרון של המערכת ו $x_i = 1$ אז המשמעות שכדי לפתור את המשחק צריך ללחוץ על לחצן i.

בנוסף $\vec{x}$ אחד הפתרונות אתכן והיו כמה.

הגדרה 3.4 שיטת פתרון בעזרת יצירת מטריצה שכנויות על ידי וקטור שינויים תקראה שיטה הסטנדרטית

בגלל שאמרנו שנציג כמה שיטות ונרצה להתייחס לשיטה זה בשם לכן קראנו שיטה סטנדרטית

תוצאה דומה אפשר לראות בפרק מהספר [2]. מרגע שהצלחנו לתאר את הבעיה מערכת משוואות לינארית על שדה $Z_2{}^n$ מכן נוכל להיעזר בכלים של אלגברה לינארית כדי למצוא את הפתרון כמו מציאת פתרון בעזרת דירוג, מציאת מטריצה פסאודו הפוכה וכולי.

הערה 3.6 עבור לוח $[n \times n]$ כמות הלחצנים n^2 ולכן גודל המטריצה שתוארה במשוואה 3 היא $[n^2 \times n^2]$

כלומר כמות הזיכרון שצריך כדי לשמור מטריצה ללוח ריבועי שאורך צלע הוא n דורש לשמור מטריצה עם n^4 ערכים .

3.1 פתרון בעזרת שיטה הספרדית

הצגנו גישה פתרון בגישה הסטנדרטית כפי שתיארנו בהגדרה 3.4. נרצה להראות שיטה נוספת למציאת פתרון. הפתרון שנציג הינו חלק מחידה שניתנה למתמטי. שיטת הפתרון שהוא מציג נובעת מהערה 3.6 שכמות המידע שצריך לשמור גדל בקצב n^4,

כאשר משחק הוא על לוח ריבועי ש n מייצג כמות הלחצנים לשורה. הצורך לצמצום כמות המשתנים נבעה מצורך פרקטי כי לא היה מספיק זיכרון לאכלס את כל מידע באותה תקופה ולכן שיטה זה הומצאה.

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 שכתבה במאמר [1] ממדיה הם $n \times n$. כלומר כמות הערכים המטריצה המתקבלת שווה ל n^2 שקטנה משמעותית ממערכת המתקבלת בשיטה הסטנדרטית.

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 הגישה הספרדית.

הגדרה 3.5 גישה הספרדית גישה פתרון שניה שנציג בעבודה זה.

נתאר את הגישה הספרדית זה הינה גישה שמתאימה לכל גודל של לוח משחק. כדי להקל על תיאור בעזרת דוגמה על לוח 3×3

נניח שאיור 9 מתאר את הלוח הנתון כאשר המשבצות הינן הלחצנים ומספור זה אינדקסים הערה 3.2 רק שהפעם נתחיל את המספור מ 0

שיטת הפתרון של מאמר הספרדי מאתחלת את הלוח עם n משתנים, מנסחת n משוואות ובעזרת מערכת שכזה מוצאת את הפתרון. במקרה של דוגמה באיור 9 כמות המשוואות תהיה 3 וכל משוואה תהיה עם 3 נעלמים ולכן נקבל מערכת משוואת שמטריצה המייצגת הינה מסדר 3×3.

שיטת הספרדית מתחיל בכך שממלאים את השורה העליונה במשתנים כפי שמתואר באיור 10. לאחר מכן עוברים שורה שורה וממלאים אותה במשתנים שמשפיעים על הלחצן.

בשלב זה נסביר את אופן המילוי ומשמעות המשתנים. נסמן ב $x_i = 1$ אם נילחץ על לחצן i. נזכיר לפי הערה 3.4 שלחצן נחשב ללחוץ עם הוא נלחץ מספר אי זוגי של פעמים כי מספר זוגי מחזיר את הלוח למצב המקורי לכן אנחנו מתייחסים רק האם נלחץ הלחצן או לא.

איור 9 : לוח 3×3 עם אינדקסים

0	1	2
3	4	5
6	7	8

אם נתייחס לכל הלחצנים כווקטור מסודר לפי אינדקס i נקבל $\vec{x}$ כפי שהגדרנו בהגדרה 3.3.
היות ומטרה לשנות את מצב הלחצנים למצב 1.
כלומר צריך שכמות לחיצות על לחצנים שמשפיעים על משבצת סכום במודול 2 היה 1.
נדגים על כמה לחצנים מאיור 9. מצב הלחצן תלוי האם הלחצנים סמוכים לו ועצמו לחוצים .
עבור לחצן 0. מתקבלת המשוואה :

$$x_0 + x_1 + x_3 = 1$$

ועבור לחצן 3 :

$$x_0 + x_3 + x_4 + x_6 = 1 \tag{4}$$

וכך ניתן להגדיר אילוצים לכל הלחצנים.

הגדרה 3.6 המשוואה המתקבלת עבור לחצן i מחיבור מודל 2 עם נלחצו לחצנים המשפעים על לחצן תקראה משוואת אילוצים על לחצן i

משוואה 4 הינה משוואת האילוצים שללחצן 3.

עבור משחק לוח ריבועי באורך שורה n שהלחצנים ממספרים לפי הערה 3.2 ניתן לנסח בנוסחה פשוטה :

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

איור 10 : לוח 3×3 מאותחל

x0	x1	x2
3	4	5
6	7	8

לאחר שהגדרנו את משוואת האילוצים נסביר כיצד למלאה את השורות הנותרות לפי השיטה הספרדית.

כל לחצן ימלא לפי משוואת האילוצים של הלחצן שמעליו.

כלומר כדי למלאה את לחצן 3 נסתכל על משוואת האילוצים של לחצן 0.

$$x_0 + x_1 + x_3 = 1$$

ניזכר כי המשוואה שהתקבלה הינה על שדה מודול 2. לכן בעזרת העברת אגפים מתקבל.

$$x_3 = 1 + x_0 + x_1$$

ונכתוב ערך זה בלחצן 3 כמו שמתואר באיור 11

איור 11 : לוח 3×3 מילוי משבצת 3

x0	x1	x2
1 + x0 + x1	4	5
6	7	8

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 12

נדגים מילוי משבצת 6 משורה שלישית לכן נצטרך ששורה שניה חושבה.

נסתכל על משבצת מעל כלומר משבצת 3 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

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

איור 12: לוח 3×3 מלאה

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

כדומה נעשה לשאר הערכים. התוצאה מתקבלת מתוארת באיור 12

נבחין שלאחר שמילאנו את כל הלוח כמו שמתואר באיור 12 נותרו לנו עוד n משוואות אילוץ שתלויות בשורה אחרונה ולכן מאמר [1] מציאה להוסיף שורה וירטואלית כדי להשתמש במשוואות עלו כמו שמתואר באיור 13

איור 13 : 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

המאמר טוען שהיות שורה זה לא באמת קיימת לכן ערכי של משבצת המתקבלות חייב להיות שווה ל 0 ולכן קיבלנו n משוואת על n נעלמים כאשר בדוגמה שלנו $n = 3$ לכן אפשר לנסות לפתור את המערכת הנתונה.

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 בין שיטה הספרדית ושיטה שהצגנו בפרק הקודם.

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

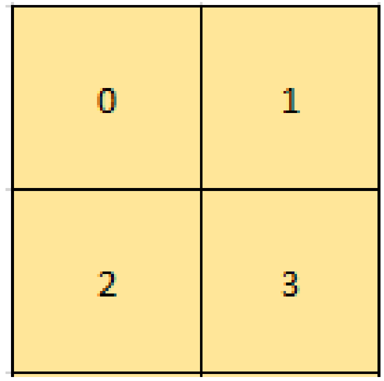
משוואות האילוצים פורמלית היא לב השיטה הספרדית מכיוון שהתקדמות בשורות מבוססת על המשוואות עלו.

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 שהצגנו במשוואה 3 מתקבל ממשוואת האילוצים שמשתנה x_i מופיעה בהם באותם אינדקסים j בווקטור שינוי של לחצן i כך שבערך $t_{i,j}$ מתקבל ערכים שווים ל 1 ולכן מתקבלת אותה מטריצה.

נדגים זאת על לוח 2×2 שמתואר באיור

וקטור השינויים :

איור 14 : לוח 3×3 מלאה כולל שורה וירטואלית



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

לכן המטריצה מהצורה :

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

ומשוואת האילוצים מהצורה

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

תעלומה נוספת בשיטה הספרדית היא למה צריך שורה וירטואלי, למה ערכה שווה ל 0 ואיך המערכת משוואת שלו מצטמצמת ל n משתנים

הסבר לתופעה זה ניתן בעזרת הצגה המטריצה נראה זאת על מטריצה של משחק על לוח 3×3

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

איור 15 : המטריצה לאחר דירוג לפי שיטה הספרדית המתקבלת מלוח 3×3

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

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

נבחין כי מילוי משבצת j בשיטה הספרדית מחייבת פעולת הצבה של משבצת k שכבר מולאה פעולה זה שקולה לחיבור עם אות שורה במטריצה, כלומר אפשר לתאר זאת על ידי פעולת שורה המתוארת כך,

$$j \leftarrow j + k$$

תקבל ששיטה הספרדית הינה שיטה חכמה לדרג את הבעיה עד שנותר n משוואות. נבחין שמשבצת שחושבו שהם משבצות מ 3 עד 8 באיור 9 בשיטה הספרדית שקולה לשורה $i - 3$ במטריצה המורחבת המדורגת באיור 15

לסיכום השיטה הספרדית היא שקולה לדירוג חכם של המטריצה. אפשר לומר שמספיק היה לפתור רק n המשוואות של שיטה הספרדית כדי למצוא את הפתרון של המשחק.

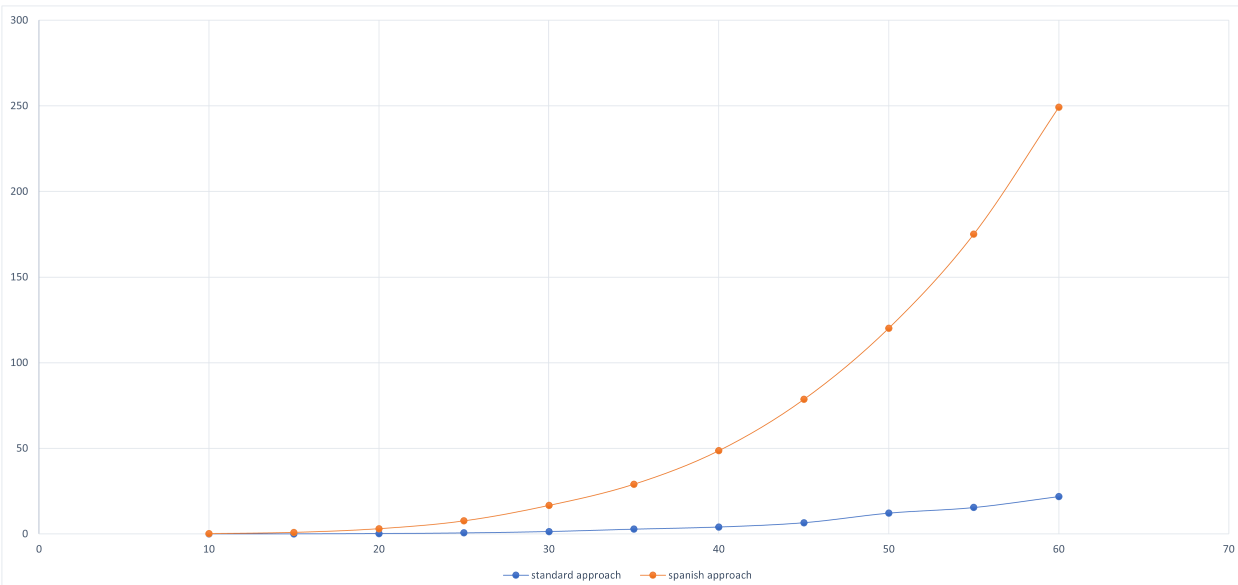
לפי חישוב סיבוכיות לדרג מטריצה כללית בגודל $n^2 \times n^2$ זה $O(n^2 \cdot n^4) = O(n^6)$.

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 16 : גרף מתאר ביצועים על לוח ריבועי גודל שורה מול זמן



באיור 16 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 16 ניראה 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.

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 $Z_2{}^n$ ניקרא לה מכפלה סקלרית תסומן $x \cdot y$ ונגדיר אותה כך:

תהי $\vec{x}, \vec{y} \in Z_2{}^n$ 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 מודול 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 $\vec{x}.\vec{y} \in Z_2{}^n$ יקראו מאונכים אחד לשני נסמן זאת $\vec{x} \perp \vec{y}$ 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 מודול 2. היות ותוצר של מכפלה פנימית הינו אפס גם לאחר מודול 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 3.5
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 $\vec{x} = [x_1, x_2, \cdots, x_n]^T$

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

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 מודל 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 3.4 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 17 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.

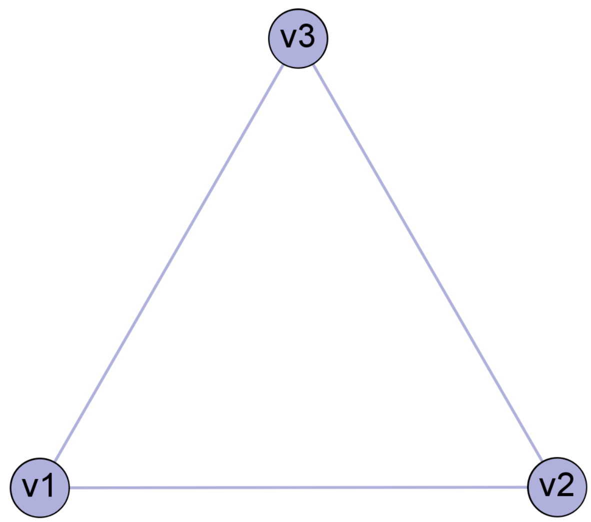
שאלה טביעת שנובעת כשגילנו שיש היא כמה פתרונות יש למשחק מסוים.

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.

איור 17 : משחק על גרף



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.

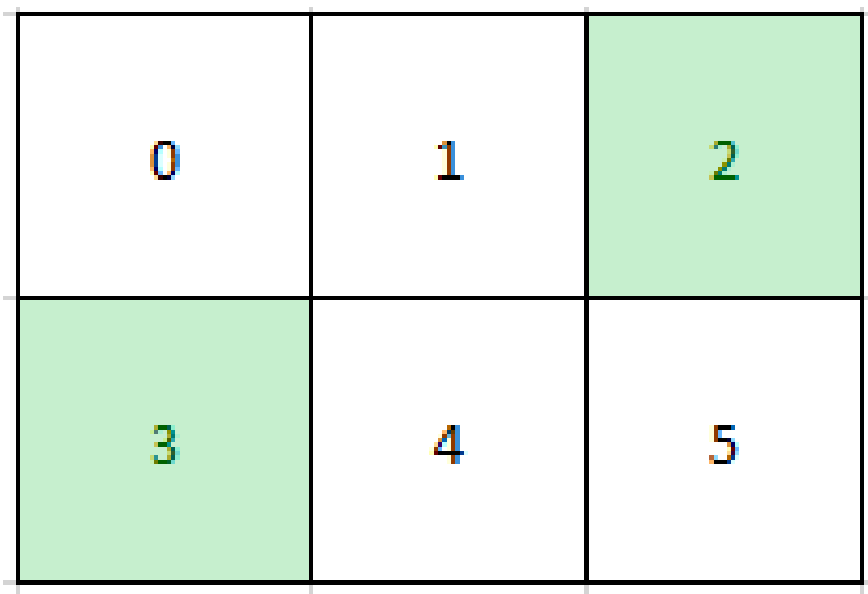
5 פתרון מינימלי עבור לוחות מלבניים

בפרק זה נציג פתרון לסוג מסוים של פתרונות שרצינו להציע. סוג זה של פתרונות מביאים רמז ונראה 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 18 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.

איור 18 : פתרון מינמלי של משחק



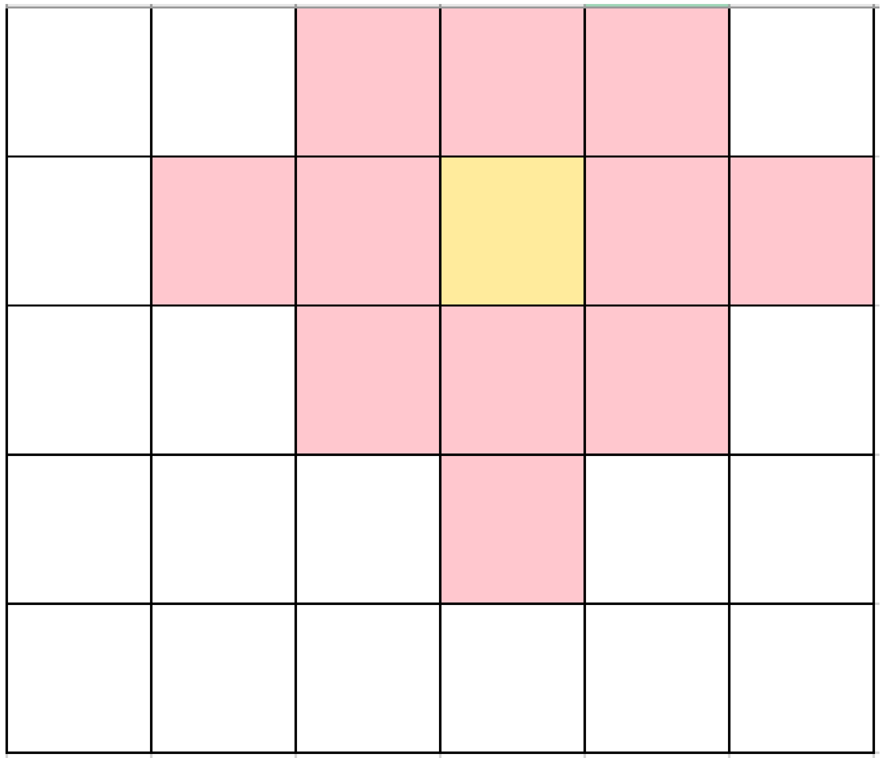
השאלה שנפתור בפרק זה לאיזה לוחות קיים פתרון מינמלי כאשר מצב התחלתי הוא שכל הנורות במצב 0.

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

אזורים מתים על איור 18 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

איור 19 : אזור מת שנוצר מלחצן באמצע הלוח



באיור 19 אפשר לראות שאם נלחץ על לחצן בצהוב האזור המת הי האזור באדום כולל הלחצן עצמו. תכונה זה כלי מרכזי בהוכחה במשפט הבאה

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,

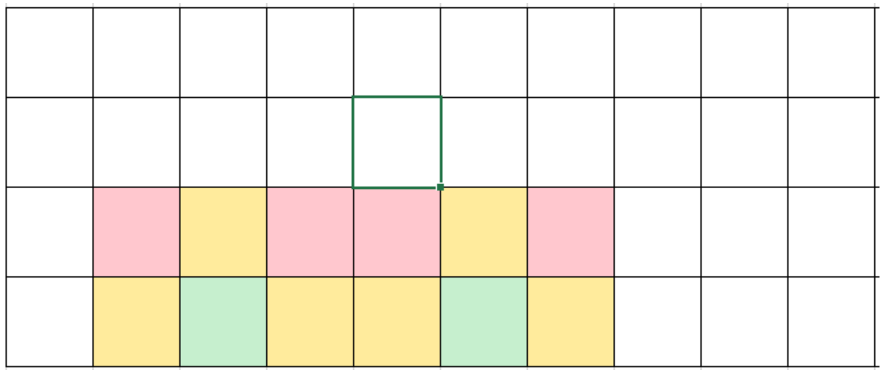
היות ושורת אינסופיות נציעה אסטרטגיה להדלקת השורה וניראה את הקשיים שנפגוש.

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 20 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 20 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

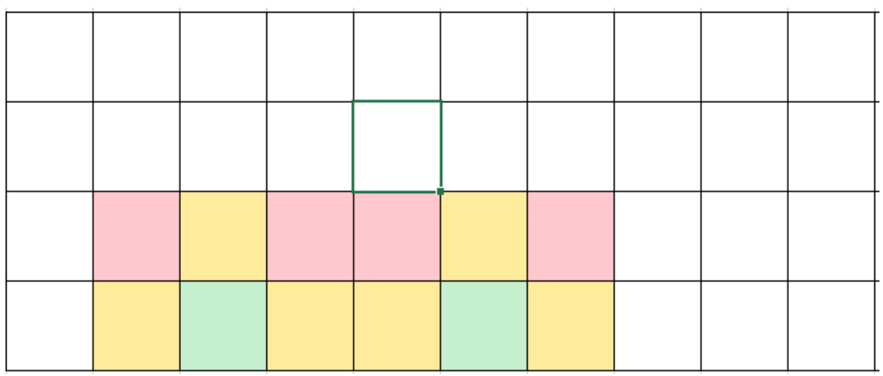
שקומה שלו גדולה מ 1.

איור 20 : מילוי קומה ראשונה על ידי לחיצות רק בקומה ראשונה



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.

איור 21 : מילוי קומה ראשונה

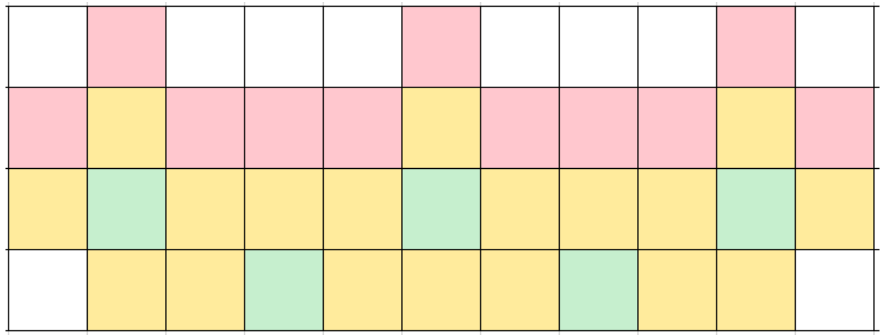


באיור 21 אפשר לראות הדגמה קטנה של אסטרטגיה שכזה.

נרצה להראות אסטרטגיה שכזה מובילה לאזורים מתים שלא ניתן למלאות כל עוד רוצים שהפתרון היה מינימלי. אם נסתכל באיור 22 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.

איור 22 : מילוי קומה ראשונה



נחזור ונסתכל על איור 20 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 22 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.

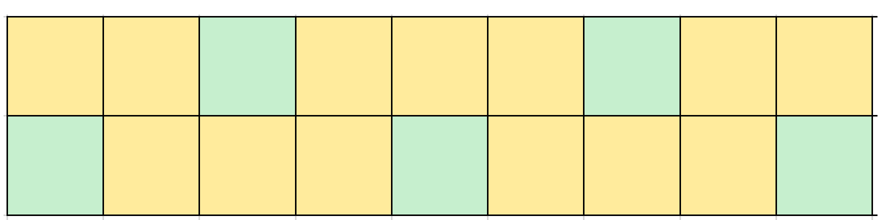
קיבלנו שאפשר להקטין את הלוח לרוחב של 7 משבצות ועדיין לא היה פתרון מינמלי.

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.

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 23

איור 23 : פתרון ללוח 9×2

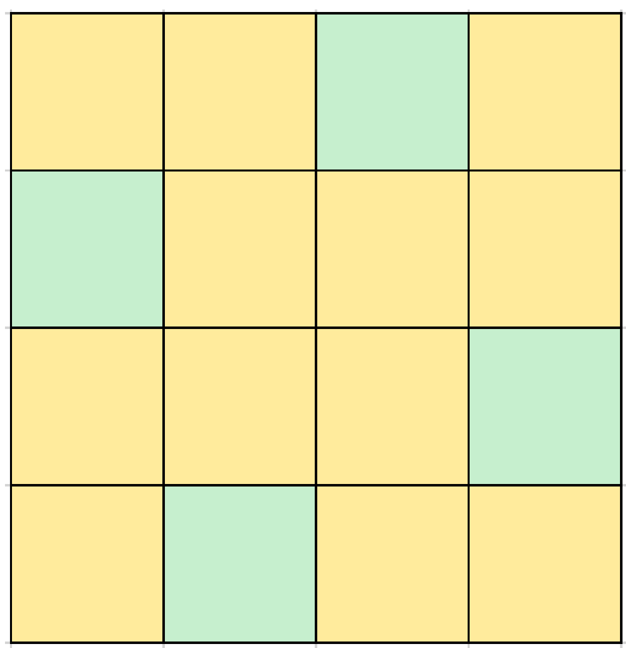


לאחר שהבנו שלוחות בגודל $m \times 2$ כאשר m אי זוגי קיים פתרון מינמלי נשאל מהו הלוח הגדול ביותר בעל פתרון מינמלי כאשר הלוח אורך ורוחב גדולים מ 2.

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 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 24

איור 24 : פתרון ללוח 4×4



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]]
```

מקורות

[1] ALL LIGHTS AND LIGHTS OUT An investigation among lights and shadows by SUMA magazine's article by Rafael Losada Translated from Spanish by Ángeles Vallejo

[2] Lecture 24: Light out Puzzle , SFU faculty of scienc department of mathematics

[3] algebra book