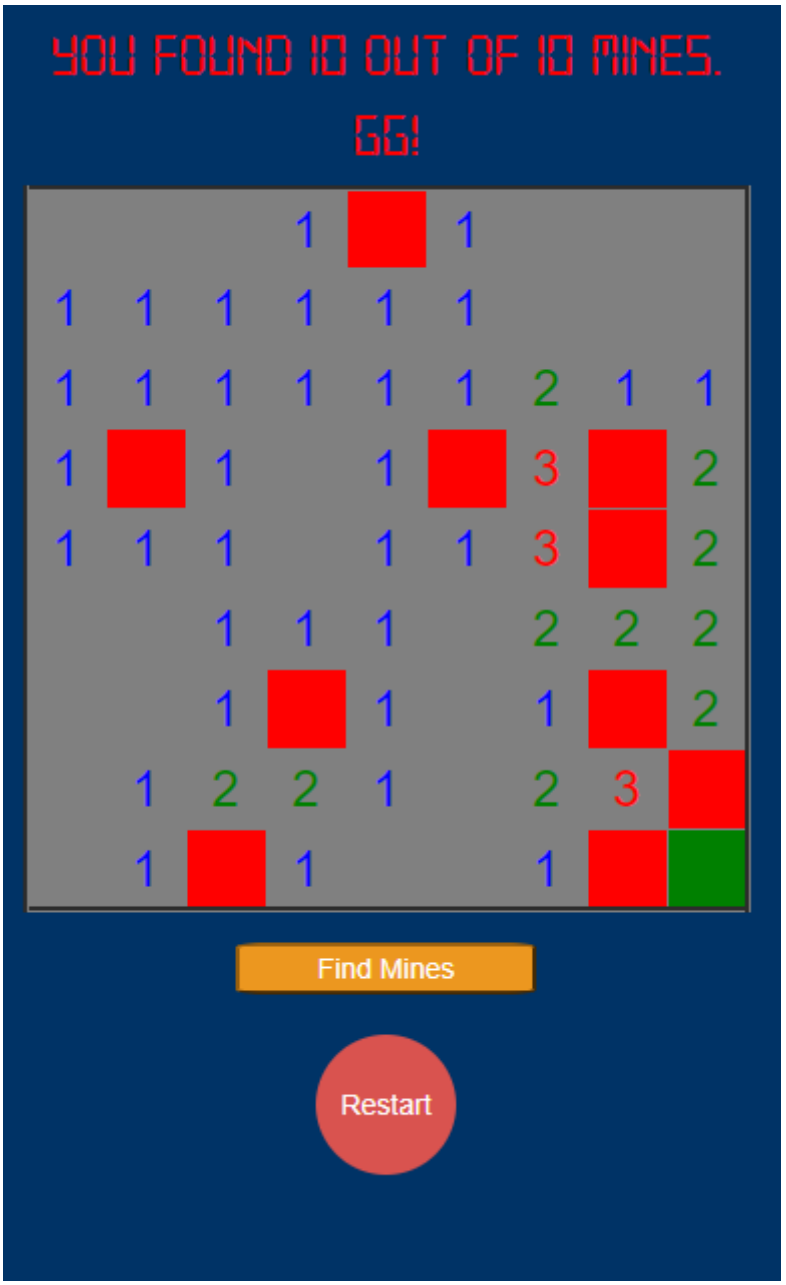
חוברת - פותר ממוחשב שולה מוקשים



מגיש: אלון סגל

ת.ז: 213174857

בית ספר: מקיף ע"ש זיו ומרקס ירושלים

תאריך 25.5.21

תוכן עניינים:

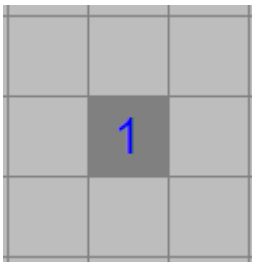
מדריך למשתמש

הסבר על המשחק

לפני הסבר על שימוש באתר, ומכיוון שזהו משחק לא נפוץ ברמה שכל אדם מכיר ויודע איך לשחק בו, יכלל במדריך למשתמש גם מדריכון להתחלה מהירה של משחק שולה מוקשים.

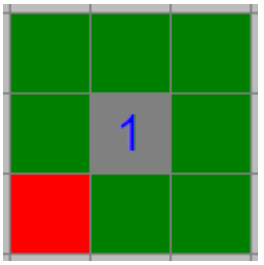
בהקדמה, שולה מוקשים הוא משחק שהיה מאוד נפוץ במחשבים של Windows 7 ומטה, ועל כן, רוב האנשים בשלב כלשהו נגעו בו, גם אם לא הבינו לחלוטין את חוקי המשחק.

משחק שולה מוקשים מורכב מלוח מלבני, באותו לוח ממוקמים מוקשים בכמות מסוימת, על השחקן לפתוח את כל הלוח בעזרת לחיצות, כאשר לחיצה על מוקש תוביל לסיום המשחק בהפסד, על כל לחיצה שאינה מוקש, המשחק יחשוף בפני השחקן מספר (או מספרים), מספרים אלו, יורו לשחקן כמה מוקשים יש בשמונה המשבצות המקיפות את אותה משבצת. לדוגמא:



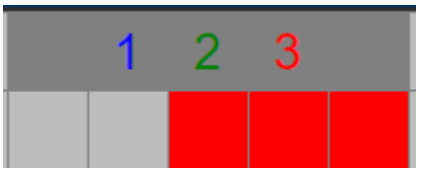
סביב המשבצת המכילה את הספרה אחת, ישנו מוקש אחד.

לפי צירופים של כמה מספרים, לרוב ניתן להסיק על משבצת אחת או אחרת שהיא חפה ממוקשים, או שקיים בה מוקש, במידה וקיים בה מוקש, ניתן להגיד על משבצות אחרות בשטח גם אם הן מוקשים או לא, בהתאם למספר, לדוגמא:

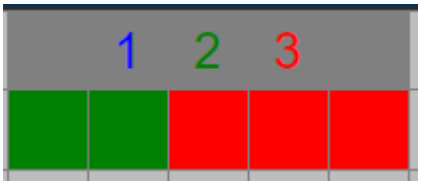


במידה והסקנו שהמשבצת משמאל למטה שהיא מוקש, ניתן להסיק כי כל שאר המשבצות המקיפות את האחד הזה הן לא מוקשים, כלומר, המספרים הם אבסולוטיים, מספר המוקשים שהם מסמנים זה בדיוק מספר המוקשים שיש סביב המשבצת.

דוגמא נוספת:



המשבצת שמכילה את המספר שלוש היא משבצת פינתית, שמוקפת במשבצות אחרות, כלומר, יש סביבה רק שלושה משבצות פנויות, והן כולן מוקשים (לפי המספר שבתוך המשבצת) ולכן ניתן לסמן את כולם בצבע האדום



מכיוון שניתן להסיק שמתחת לשלוש ישנם שלושה מוקשים, אפשר להסיק ששתי המשבצות השמאליות בשורה זו הן לא מוקשים, שכן "דרישת המוקשים" הנחוצה כבר מולאה על ידי המשבצות מימין.

מכאן ניתן לפתח את שיטה זו רבות, וישנן שיטות אחרות כוללות שינון דפוסים וסטטיסטיקה, לשם ההדגמה, הדגמתי את השיטה הנפוצה ביותר וזו שתצליח לפתור אחוז מאוד גבוה מהמשחק.

לקריאה נוספת ולהבנה יותר עמוקה של אסטרטגיות הפתירה של המשחק.

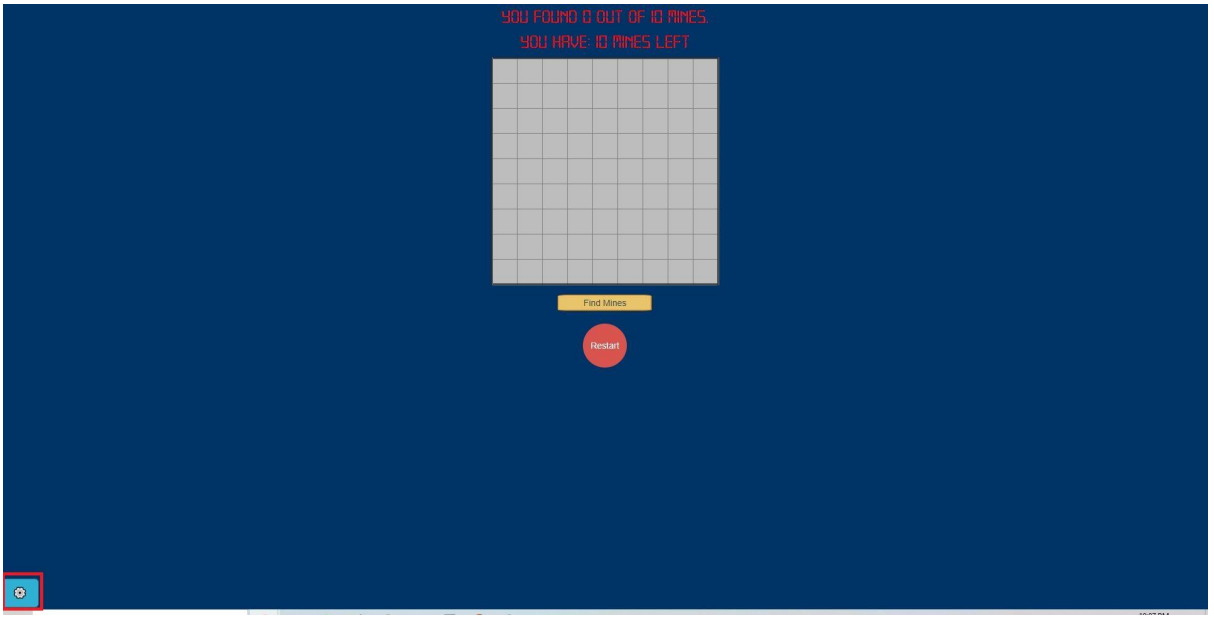
לאחר שעקרונות המשחק הוסברו, ניתן לפנות אל עבר החלק העיקרי:

באתר הזה שלי, בניתי פותר שולה מוקשים, כמשתמש באתר,הכרחי שתשחק שולה מוקשים תוך כדי שימוש באתר.

לאחר שהתחלת משחק שולה מוקשים, עליך להזין במקביל נתונים מלוח המשחק שלך אל הלוח באתר, ומהלוח באתר ללוח במשחק, כלומר, אתה מזין ללוח שבאתר שלי את המספרים שמופיעים על לוח המשחק, והוא "בתמורה" יראה לך היכן נמצאים מוקשים והיכן כן אפשר ללחוץ, השימוש באתר מומלץ לאנשים שמעוניינים לפתור משחקי שולה מוקשים בלי להתאמץ יותר מדי, המשחק יכול לעזור בלימוד המשחק, ויכול לעזור לשחקנים שמשחקים ונתקעו באמצע של משחק.

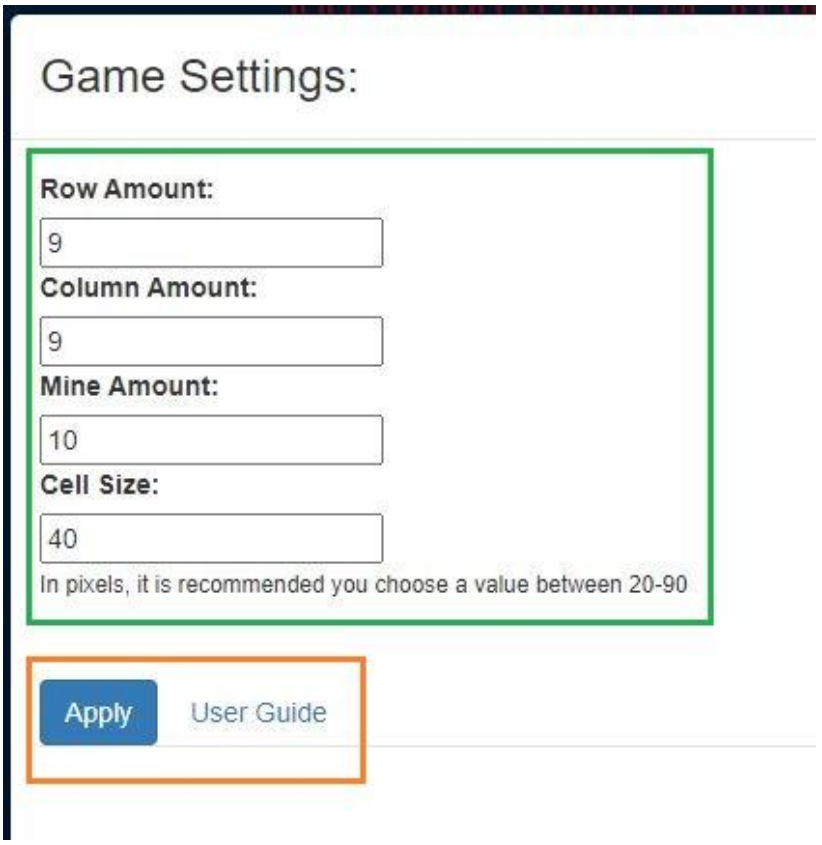
דף הנחיתה באתר:

עם פתיחת האתר יוצג בפני המשתמש הדף הבא:



במידה ותרצו לשחק על לוח מהסוג הקטן ביותר, של גודל תשע על תשע עם עשרה מוקשים בלוח, תוכלו לדלג לקטע שמסביר על השימוש בתוכנה, במידה ולא, עליכם ללחוץ על הכפתור תכלת הקטן מצד שמאל למטה של הדף (שמסומן בריבוע אדום).

לאחר לחיצה על הכפתור, יפתח בפני המשתמש המסך הבא:



בריבוע הירוק:

שורות אלו ישפיעו על הלוח, כאשר השורה הראשונה תקבע את כמות השורות בו, השנייה את כמות העמודות, השלישית את מספר המוקשים (תלוי בלוח המשחק אותו התוכנה תתבקש לפתור) ואת גודל כל תא, כאשר האחרון הוא ויזואלי בלבד וכאשר מרחפים מעל אזור זה האתר יציג למשתמש דוגמית של גודל כל תא.

*במידה ויוכנסו מספרים עם שברים לשורות הנ"ל, המספר יתעגל למספר השלם הנמוך, כך, בין אם יוכנס 15.9 או 15.1, המספר שיתקבל יהיה 15.

בריבוע הכתום: הכפתור "Apply" מחיל את ההגדרות שהזנתם בשורות על לוח המשחק, והכפתור "User Guide" הוא כפתור קישורי, וישלח את המשתמש אל דף שמכיל קובץ PDF, הקובץ שיפתח הוא הקובץ הזה!

לאחר שקבענו את מימדי הלוח וכמות המוקשים לפי הלוח אותו אנו מבקשים, נעבור לשימוש בלוח עצמו.

בראשית הדף, בחלקו העליון, יופיע הטקסט הבא:

השורה הראשונה מתארת כמה מוקשים נמצאו, וכמה מוקשים קיימים בסה"כ

השורה השנייה מתארת כמה מוקשים נשארו למשתמש למצוא, כלומר כמה מוקשים חבויים עוד יש.

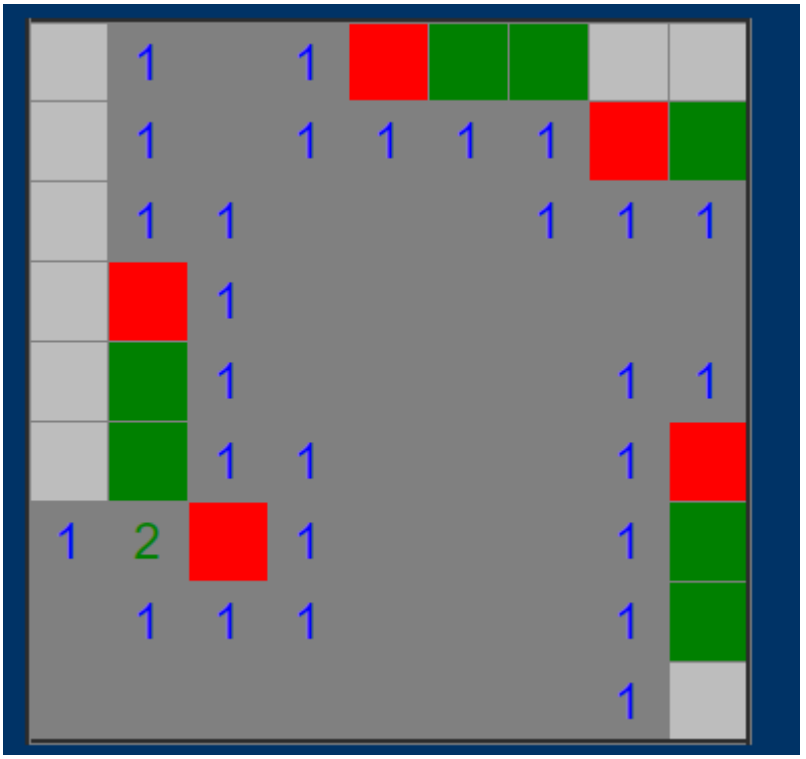
השימוש בכלי עצמו:

ועכשיו ניגש לשימוש בחלק העיקרי של האתר, לוח הפתירה.

כפי שצוין בתחילת המדריך, השימוש באתר מומלץ שיעשה במקביל למשחק שולה מוקשים.

נתחיל מלחיצה אקראית על לוח המשחק:

לאחר הלחיצה התקבל הלוח הזה, נזין את נתוני הלוח לאתר ונלחץ על כפתור Find Mines



לאחר הזנת הנתונים במלואה, כאשר מתאימים את המשבצות במשחק עצמו למשבצות בלוח, ולאחר לחיצת הכפתור, נקבל נתונים, באדום, מסומנים מוקשים, את נקודות אלה נעביר ללוח שולה המוקשים, לוח שולה המוקשים יראה כך:

ועכשיו, נלחץ על מקומות שאוששו שפנוים:

את הנתונים החדשים נזין ללוח המשחק וכך חלילה, ישנם x סוגים של משבצות שהמחשב יסמן:

ירוק – משבצת שבטוח שאין מוקש עליה

אדום – משבצת שבטוח שמכילה מוקש

ירוק בהיר* – משבצת שרוב הסיכויים שאינה מכילה מוקש

ורוד* – משבצת שרוב הסיכויים שכן מכילה מוקש

צהוב* – משבצת רנדומלית לחלוטין

*מקרים מיוחדים:

במקרה שהמשחק יסמן משבצת ירוקה בהירה, על השחקן ללחוץ על המקבילה לה בלוח המשחק, כאשר רוב הסיכויים שהמשחק ימשך.

במידה והמשבצת אדומה בהירה, על השחקן להציב בה את הצבע האדום וללחוץ על כפתור Find Mines, ומשם להמשיך את המשחק, לאחר שימשיך, במידה ולא נפסל, המשבצת תיחשף עבורו בהמשך בצבע אדום.

מדריך למתכנת

טבלת פונקציות:

פונקציה	טענת כניסה	טענת יציאה
BuildBoard()	תקבל או תשתמש במשתנים שיכילו את גודל המשבצות בלוח, רוחב לוח, אורך לוח. קריאה מתבצעת בטעינת הדף, או מפונקציית שינוי ערכי הלוח. משתנים גלובליים: num, textTableTag, CellSize	הפונקצייה בונה את לוח ומערך, כאשר הלוח בנוי בצורת טבלה ועליו יתבצעו השינויים של המשתמש, שיוכתבו למערך שנוצר.

Restart()	שימוש ברוחב ואורך הלוח משתנים גלובלים: Row, Board, Col .	תחדש את לוח המשחק ע"מ שיוכל לשמש למשחק חוזר
ChangeValues()		מחלצת מדף הHTML נתונים לבניית הלוח, נתונים אלו כוללים אורך, רוחב, כמות מוקשים וגודל כל תא בלוח.
CurrentMines()	רוחב ואורך הלוח, הקריאה מתבצעת מפונקציית FindMinesMain()	סופרת את כמות המוקשים בלוח ומציגה זאת למשתמש
MinesLeft(CurrentMines)	כמות המוקשים סה"כ בלוח, הקריאה לפונקציה מתבצעת מפונקציית CurrentMines()	מחשבת ומציגה למשתמש את כמות המוקשים שנשארו לו ע"מ לנצח.

FindMinesPattern()	מקבלת את המשתנים Row, Board, Col , Findingmines.	בדיקה של מוקשים או משבצות "בטוחות" ע"י חיפוש של דפוסים שהוזנו מראש לתוכנה וידועים ככאלה שניתן להסיק מהם מסקנות בטוחות לגבי הלוח.
SwitchNum(currentThis)	מיקום לחיצה על הלוח(משבצת). מקבל את num- הid של המשבצת.	מחליפה את המספר המוצג על המשבצת ומחליף גם במערך (Board).
FindMinesMain()	נעשה שימוש במערך Board, בגודל ורוחב הלוח משתנים גלובליים:,Row, Col Findingmines.	פונקציה מרכזית, מרכזת בתוכה את הקריאות לשאר הפונקציות שעליהן מוטלת המשימה למצוא וודאות לגבי משבצות
CheckTopRow(c) CheckBottomRow(c) CheckLeftCol(c) CheckRightCol(c) CheckUpLeftCorner(c) CheckUpRightCorner(c)	מקבלות מיקום ותוכן של משבצת במסגרת הלוח משתנים גלובליים:,Board mines, Notmines, Findingmines.	מנסה לחפש וודאות לגבי משבצות שמקיפות את המשבצת שהתקבלה בטענת הכניסה

CheckDownLeftCorner(c) CheckDownRightCorner(c) CheckTopRow(c) CheckBottomRow(c) CheckLeftCol(c) CheckRightCol(c) FindMinesCenter()		
CheckRandom()	נעשה שימוש במערך Board, בגודל ורוחב הלוח משתנים: Board, FindingMinesRandom, Col, Row.	הצבה של מוקש במשבצת מסוימת, ובדיקה של הלוח כאשר הוא מוצב, כלומר, האם הלוח יסתדר לוגית במידה וקיים מוקש במשבצת זו.
RandomCheck2() RandomCheckTopRow() RandomCheckBottomRow() RandomCheckLeftCol() RandomCheckRightCol()	נעשה שימוש במערך Board, בגודל ורוחב הלוח משתנים גלובליים: Row, Col, CheckBoard, Notmines, mines	חיפוש וסימון של משבצות שרוב הסיכויים שלא קיימות בהן מוקשים

random()	בוחר משבצת מבין המשבצות הריקות משבצת וצובע אותה בצבע צהוב. משתנים גלובליים:Row, Col.	בוחר משבצת מבין המשבצות הריקות משבצת וצובע אותה בצבע צהוב.
PaintGreen() PaintRed() PaintPink()	נעשה שימוש במערך Board, בגודל ורוחב הלוח ובמשתנים Row, Col.	צובעות את הלוח בצבעים רלוונטים בהתאם לקריאה שנעשתה לפונקציה

אלגוריתמים מרכזיים:

אלגוריתם ראשון: בניית הלוח

בניית הלוח מוגדרת על פי שני משתנים, Row וCol, שניהם יקבעו, בהתאם לשמם, את גודל הלוח, על פי ברירת מחדל, גודל הלוח יהיה 9 על 9, שהוא גודל הלוח הקטן ביותר במשחק שולה המוקשים הקלאסי.

הפונקציה BuildBoard היא שבונה את הלוח לראשונה, ועל כן במידה והמשתמש יבחר לשנות את גודל הלוח, היא גם תהיה זו שתשנה אותו.

הקריאה לפונקציה מתבצעת בתחילת דף הJS, ודרך פונקציית ChangeValues().

משתני עזר

Row – כמות השורות בלוח, 9 על פי ברירת מחדל

Col – כמות הטורים בלוח, 9 על פי ברירת מחדל

CellSize – גודל התאים בלוח, 40*40 פיקסלים על פי ברירת מחדל

```
function BuildBoard() {
    var num = 0;
    textTableTag = "<table
style='background-color:#BDBDBD;'border='3'>";
    for (sh = 0; sh < Row; sh++) {
        textTableTag += "<tr>";
        for (am = 0; am < Col; am++) {
            textTableTag = textTableTag + "<td id='" + num +
                "' onclick='SwitchNum(this);' style='width:" +
CellSize + "px; height:" + CellSize + "px;'>  </td>";
            num++;
```

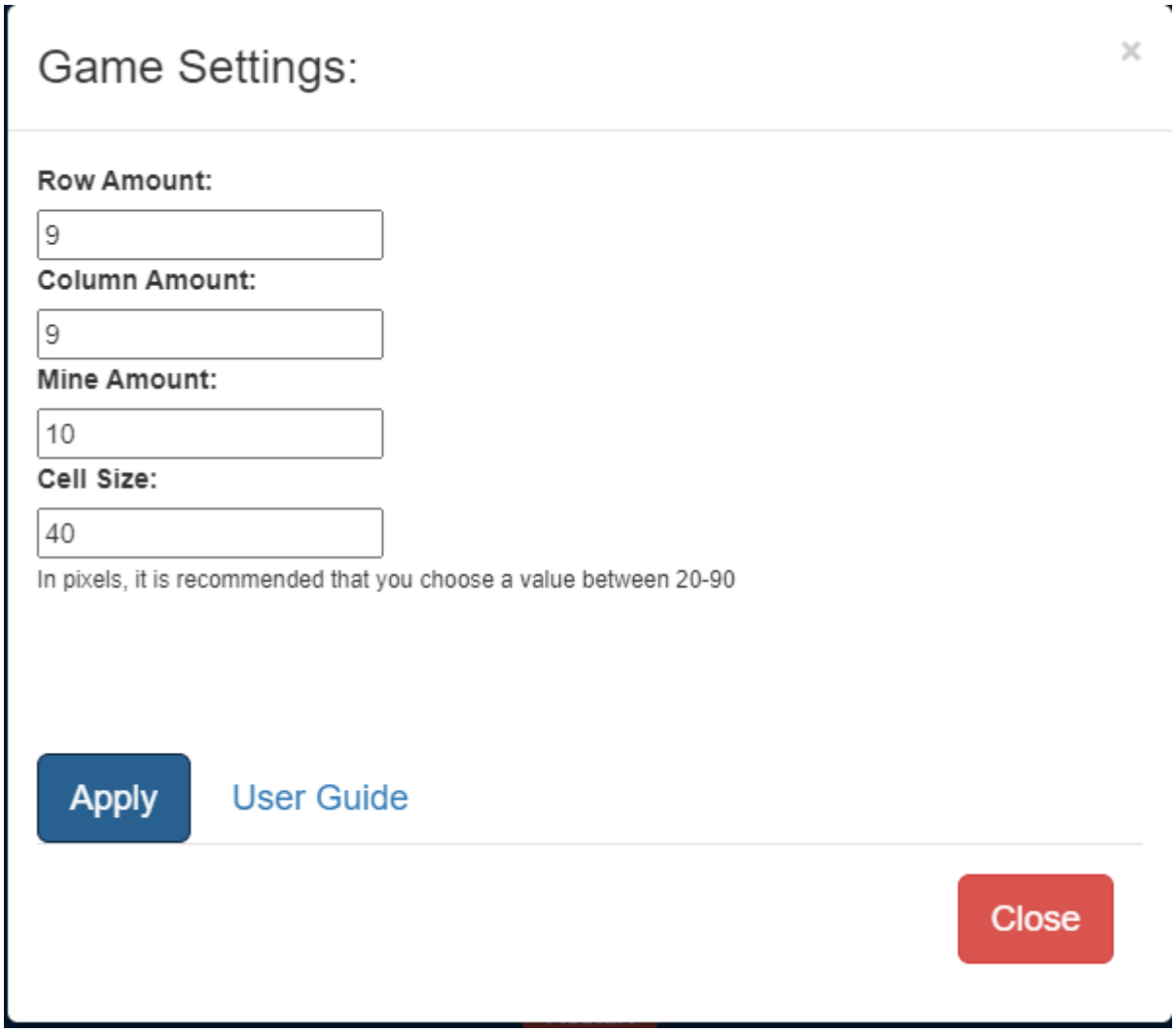
```
        }
        textTableTag += "</tr>";
    }
    textTableTag += "</table>";
    document.getElementById("Board").innerHTML = textTableTag;

    Board = new Array(Row);
    for (var i = 0; i < Board.length; i++) {
        Board[i] = new Array(Col);
    }
    for (var i = 0; i < Row; i++) {
        for (var j = 0; j < Col; j++) {
            Board[i][j] = -1;
        }
    }
    CurrentMines();
}
```

הסבר - הפונקצייה בונה את הלוח, תחילה, היא מגדירה משתנה בשם num, המשתנה בהמשך ישמש להגדרה של id ייחודי לכל תא בלוח, לאחר מכן, הפונקציה תגדיר בעבור כל הטבלה את צבע הרקע ואת עובי הגבול שלה, הפונקציה תמשיך ללולאה, הלולאה תרוץ Row פעמים, כאשר בתוכה, קיימת עוד לולאה, מה שהלולאה בעצם עושה, זה יוצרת שורות בטבלה, והלולאה הפנימית, יוצרת Col טורים בשורה הרלוונטית, הלולאה הפנימית, גם קובעת את המאפיינים של כל תא את הid שלו, ואת מימדיו.

המשתנה num עולה +1 בעבור כל איטרציה של הלולאה הפנימית, לאחר ששתי הלולאות מסיימות לרוץ, מוזרק הקוד לשטח שיועד לו בדף הHTML, ונוצר מערך משחק, מערך זה הוא השלד של הלוח הויזואלי, והשינויים שהמשתמש יעשה בלוח שבדף הHTML יועברו על מערך זה, המערך ישמש על מנת למצוא היכן קיימים מוקשים, לאחר כל שינוי בגודל הלוח או לאחר בנייתו הראשונית, כל משבצות הלוח יהיו ריקות, וערכן יהיה -1, עוד על מה מספר של כל משבצת אומר, בהמשך.

חלק מאלגוריתם בניית הלוח, היא פונקציה משנית בשם ChangeValues, הפונקציה שולפת מקופסת מודל שקיימת בדף הHTML, פונקצייה זו מקבלת מדף הHTML את כמות השורות, כמות העמודות וגודל כל תא בלוח.

הפונקצייה נקראת מתוך דף הHTML, בפועל, זה אומר שבשהמשתמש ירצה לשנות את מימדי הלוח או את כמות המוקשים בו, הוא ילחץ על כפתור במודל שיקרא לפונקציה:



<u>הכפתור Apply הוא זה שיקרא לפונקציה, כך זה נראה בדף הHTML:</u>

```
<button type="button" class="btn btn-primary btn-lg" onclick="ChangeValues()" style="outline:none;">Apply</button>
```

הפונקציה מחולקת לארבעה החלקים עיקריים, כל אחד מהם יקבע משהו אחר, הראשון, יקבע את כמות השורות בלוח, הפונקצייה תיגש תקבל את מימדיה מהשורה הרלוונטית במודל, ולאחר מכן תבצע שאילתה שתוודא שלא הוכנסו נתונים לא חוקיים, נתונים לא חוקיים, הם מספרים שאינם חיוביים ואינם שלמים, כלומר, מספרים חיוביים שהם גדולים מ0 ולא כוללים שבר.

התהליך זהה לחלוטין גם עבור קביעה של כמות טורים.

הפונקצייה גם מקבלת את כמות המוקשים שיש על הלוח, גם פה הפונקצייה תמיר את סוג המידע המתקבל לאינטג'ר, ותוודא שהוא מספר שלם וחיובי, אך פה, בניגוד לשני השאילתות הקודמות,

הפונקציה תוודא גם שמספר המוקשים לא עולה על מספר המשבצות בלוח, כלומר שלא יהיה מצב בו ישנם יותר מוקשים מאשר מקומות בלוח.

החלק האחרון הוא שליפה של גודלי התאים, למשתמש מוצגת הדגמה של גודלי תא במימדים של בין 20-90 פיקסלים, אך הוא יכול לקבוע כל גודל, בתנאי שזה מספר חיובי.

במידה ומוכנסים נתונים לא חוקיים, הפונקצייה תציב את נתוני ברירת המחדל ההתחלתיים.

לסיום, הפונקציה שולחת קריאה לפונקציית BuildBoard, הלוח נבנה מחדש, הפעם עם הנתונים המחודשים.

```
function ChangeValues() {
    Row = parseInt(document.getElementById("Row").value);
    if (!Number.isInteger(Row) || Row < 1) {
        Row = 9;
    }
    Col = parseInt(document.getElementById("Col").value); //Cast
to int
    if (!Number.isInteger(Col) || Col < 1) { // Prevent Floats
        Col = 9;
    }
    TotalMines = parseInt(document.getElementById("Mines").value);
    if (!Number.isInteger(TotalMines) || TotalMines > Col*Row ||
TotalMines < 1) {
        TotalMines = 10;
    }
    CellSize =
parseInt(document.getElementById("CellSize").value);
    if (!Number.isInteger(CellSize) || CellSize < 1)
        CellSize = 40;

    BuildBoard();
}
```

אלגוריתם שני: שינוי מספר בעת לחיצה

```
function SwitchNum(currentThis) {
    var id = currentThis.id;
    firstRow = Math.floor(id / Col);
    firstCol = id % Col;
    if (Board[firstRow][firstCol] == -1) {
        document.getElementById(id).style.backgroundColor =
"#808080";
        Board[firstRow][firstCol] = 0;
    }
    else {
        if (Board[firstRow][firstCol] == 0) {
            document.getElementById(id).innerHTML = 1;
            document.getElementById(id).style.color = "black";
            document.getElementById(id).style.backgroundColor =
"#808080";
            Board[firstRow][firstCol] = 1;
        }
        else {
            if (Board[firstRow][firstCol] == 1) {
                document.getElementById(id).innerHTML = 2;
                document.getElementById(id).style.color = "green";
                document.getElementById(id).style.backgroundColor
= "#808080";
                Board[firstRow][firstCol] = 2;
            }
```

*קטע זה חוזר על עצמו עד הספרה 9, בהגיעו ל- 9, הקוד מפנה מחדש אל המספר 1-.

משתני עזר:

firstRow – מייצג מספר שורה.

firstCol – מייצג מספר עמודה.

מערך מספרים Board- מערך זה מכיל את סדר הלוח.

Id - מספר המשבצת במערך.

הסבר -

אלגוריתם זה הוא חלק מהפונקציה - SwitchNum(), תפקיד פונקציה זו הוא לשנות את המספר במערך ובלוח המשחק כאשר מתקבלת לחיצה על אחת מן המשבצות.

ראשית הפונקציה לוקחת את מספר הid של המשבצת שנלחצה. לאחר מכן, דרך הid, הפונקציה תחלץ את מספר השורה והעמודה של המשבצת שנלחצה, הפונקציה תבדוק מה המספר המופיע במקום המקביל במערך, ותוסיפו לערכו 1, לאחר מכן, תעדכן את לוח המשחק בשינוי המספר ותשנה את צבע המספר המופיע בלוח בהתאם למספר החדש.

אלגוריתם שלישי: בדיקה בשיטת חיסור לגבי מיקוש או אי-מיקוש של משבצת

אלגוריתם זה הוא הפונקציה – FindMinesCenter()

פונקציה זו בודקת האם ניתן להסיק מנתוני הלוח את מיקומו של מוקש או מיקומה של משבצת שאינה מוקש, במידה ומתקיים אחד משני המצבים, תצבע הפונקציה את המשבצת הרלוונטית באדום או ירוק, בהתאם למה שמכילה המשבצת.

את האלגוריתם ניתן לפצל לשלושה חלקים:

חלק ראשון – בדיקת המשבצות מסביב:

```
function FindMinesCenter(c) {
    if (-1 < Board[sh - 1][am - 1] && Board[sh - 1][am - 1] < 9) {
        Notmines++;
    } else {
        if (Board[sh - 1][am - 1] == 9) {
            mines++;
        }
    }
```

הקטע חוזר עבור כל משבצת מסביב לפונקציה, המקרה הזה בודק את המשבצת משמאל למעלה

משתני עזר:

sh – מייצג מספר שורה.

am – מייצג מספר עמודה.

מערך מספרים Board – מערך שמכיל את לוח המספרים של התכנית

Notmines – סופר כמה משבצות מסביב למשבצת מסוימת נמצאו כלא מוקש.

mines – סופר כמה משבצות מסביב למשבצת מסוימת נמצאו כמוקש.

הסבר – הקטע עובר על כל המשבצות מסביב למשבצת מסוימת וסופר כמה מהמשבצות סביבה מוקשים, וכמה לא מוקשים (בוודאות).

חלק שני: בדיקת היתכנות מוקשים סביב המשבצת

```
if (8 - Notmines == c) {
        if (-1 < !Board[sh - 1][am - 1] && Board[sh - 1][am - 1] <
!9) {
            Board[sh - 1][am - 1] = 9;
            document.getElementById((sh - 1) * Col + am -
1).style.backgroundColor = "Red";
            Findingmines = true;
        } if (-1 < !Board[sh - 1][am] && Board[sh - 1][am] < !9) {
            Board[sh - 1][am] = 9;
            document.getElementById((sh - 1) * Col +
am).style.backgroundColor = "Red";
            Findingmines = true;
```

משתני עזר:

sh – מייצג מספר שורה.

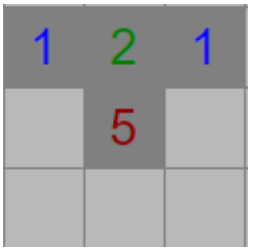
am – מייצג מספר עמודה.

מערך מספרים Board – מערך זה מכיל את סדר הלוח.

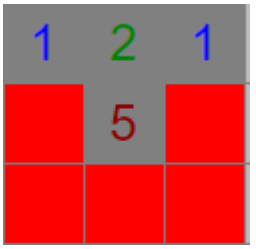
Notmines – סופר כמה משבצות סביב המשבצת ברדיוס של אחד נמצאו כלא מוקש.

mines – סופר כמה משבצות סביב המשבצת ברדיוס של אחד נמצאו כמוקש.

על מנת לבדוק האם יש בכלל טעם להציב מוקשים, תבדוק הפונקציה האם מספר המשבצות שמסביב למשבצת (8) פחות מספר המשבצות בהן לא קיים מוקש שווה למספר המופיע במשבצת הנבדקת, במידה ומתקיים תנאי זה, הפונקציה תמשיך, מטרת שאילתה זו הוא לבדוק האם יש מצב שבו למשבצת יש את מסביבה כמות משבצות סגורה שתואמת את כמות המוקשים ש"חסרים" מסביבה, לדוגמא:



במצב זה, הפונקציה תסיק כי חסרים חמישה מוקשים סביב המשבצת, ומכיוון שקיימות חמש משבצות ריקות סביבה, הרי שכולן גם מוקשים, ולכן, תסמן הפונקציה את המשבצות האלו באדום.


1 2 1
5

אלגוריתם רביעי : הצבה של מוקש במערך

```
function CheckRandom() {
    CheckBoard = Board;
    for (sh2 = 0; sh2 < Row; sh2++) {
        for (am2 = 0; am2 < Col; am2++) {
            if (CheckBoard[sh2][am2] == -1) {
                CheckBoard[sh2][am2] = 9;
                RandomCheckTopRow();
                RandomCheckBottomRow();
                RandomCheckLeftCol();
                RandomCheckRightCol();
                RandomCheck2();

                if (FindingMinesRandom == true) {
                    CheckBoard[sh2][am2] = -1;
                    document.getElementById(sh2 * Col + am2).style.backgroundColor = "#c03fab";
                    Board = CheckBoard;
                    break;
                } else {
                    CheckBoard[sh2][am2] = -1;
                }
            }
        }
        if (FindingMinesRandom == true) {
            break;
        }
    }
    if (FindingMinesRandom == false) {
        random();
    }
    FindingMinesRandom = false;
}
```

משתני עזר:

CheckBoard – מערך עזר.

מערך מספרים Board– מערך זה מכיל את סדר הלוח.

sh2 – מייצג מספר שורה.

am2 – מייצג מספר עמודה.

FindingMinesRandom – מספר המשבצת.

Id – מספר המשבצת במערך.

הסבר –

אלגוריתם זה הוא קטע מהפונקציה – CheckRandom().

פונקציה זו מציבה מוקש במשבצת ריקה ועוברת על המערך החדש שנוצר, היא תחפש נתונים חדשים בהתאם, במידה ונמצא מצב אפשרי לוגית, והצבת המוקש נכונה, כלומר שקיים סיכוי

שהמשבצת שעליה הוצב המוקש באמת מכילה מוקש אמיתי יסמן הקוד את המשבצת בצבע ורוד (מוקש פוטנציאלי).

תחילה, מבוצעת בפונקציה הפנייה של CheckBoard אל Board, המצב הזה, דה פקטו, יוצר עבורנו שני מערכים זהים, על אף ששניהם מפנים אל אותו עצם, במידה ונערוך את אחד מהם, השני לא יושפע.

העבודה בפונקציה זו תתבצע על המערך CheckBoard, זאת משום שכך נוכל לבצע שינויים בלוח (שנחוצים לפונקציה זו) ללא פגיעה בלוח המקורי, כלומר, יצירה של מערך עזר. לאחר קטע זה, תתחיל איטרציה של 2 לולאות, תפקידן לעבור על כל המערך, ולבדוק האם המשבצת הרלוונטית לאיטרציה זו ריקה, במידה וכן תתבצע הסבה של המשבצת לכזו שכן מכילה מוקש ותתחיל סדרת בדיקות, מהן נדע האם התקבלו תובנות חדשות לגבי הלוח ומה הן, קיימת בדיקה שבודקת האם קיימת משבצת אפשרית לפתיחה.

בקוד, בדיקה זו היא בעלת כמה חלקים שונים, שכן ישנם מקומות בלוח שדורשים בדיקות שבנויות אחרת, לדוגמא, מכיוון שמספר המשבצות סביב המשבצת הנבדקת יכול להשתנות בהתאם למיקומה בלוח, שכן משבצת בפינת הלוח או במסגרתו, תהיה בעלת פחות משבצות מקיפות אל מול משבצת במרכז הלוח, לראות אם הוספת המוקש נותנת לנו נתונים חדשים על מצב המשחק.

במידה והתקבלו נתונים חדשים, יתבצע שינוי של המשבצת בה הצבנו מוקש למשבצת ריקה (על מנת שלא לפגוע במערך המקורי), ואז אותה משבצת תצבע בצבע ורוד שמסמל מוקש בסבירות יחסית גבוהה, לאחר מכן תתבצע יציאה מהלולאה שמובילה ליציאה גם מהלולאה השניה.

במידה ולא התקבל נתון חדש על לוח המשחק, תחזיר הפונקציה את המשבצת שהוצב עליה מוקש למצב ריק, ותעבור להצבה במשבצת הבאה.

במידה ולא נמצאו שום תובנות חדשות מאף אחת מההצבות, תפנה הפונקציה אל פנוקצייה Random

אלגוריתם חמישי: רנדומלי

```
function random() {
    var RandomArray = [];
    for (var RaRow = 0; RaRow < Row; RaRow++) {
        for (var RaCol = 0; RaCol < Col; RaCol++) {
            if (Board[RaRow][RaCol] == -1) {
                var ClearId = (RaRow * Col + RaCol);
                RandomArray.push(ClearId);
            }
        }
    }
    var RandomNumber = RandomArray[Math.floor(Math.random() *
RandomArray.length)];
    document.getElementById(RandomNumber).style.backgroundColor =
"#f39f0c";

}
```

משתני עזר:

RandomArray – מערך עזר.

RaRow – מייצג מספר שורה.

RaCol – מייצג מספר עמודה.

מערך מספרים Board– מערך זה מכיל את סדר הלוח.

ClearId – מספר המשבצת.

RandomNumber – מייצג מספר רנדומלי במערך.

הסבר –

אלגוריתם זה הוא הפונקציה – random()

הפונקציה היא פונקציית הסוף של התוכנית, והגישה אליה תתבצע במקרי קיצון, בהם אף אחת מהפונקציות הקודמות לא הצליחו לייצר תובנות שיעזרו להבין יותר טוב היכן נמצאים או לא נמצאים מוקשים.

הפונקציה תיצור מערך חד מימדי, ואליו, תכניס את הID של כל משבצת שנמצאה שלא הוכנס בה מספר כלשהו, כלומר משבצות שלא נפתרו עדיין, מבין כל אלה, תגריל הפונקציה באופן אקראי

אחד מאיברי המערך, הפונקציה תצבע את המשבצת שהוגרלה בצבע צהוב, שיסמן עבור המשתמש שהוגרלה משבצת בצורה רנדומלית, ועליה עליו ללחוץ.

אלגוריתם שישי – מציאת מוקשים ע"י דפוסים קבועים מראש:

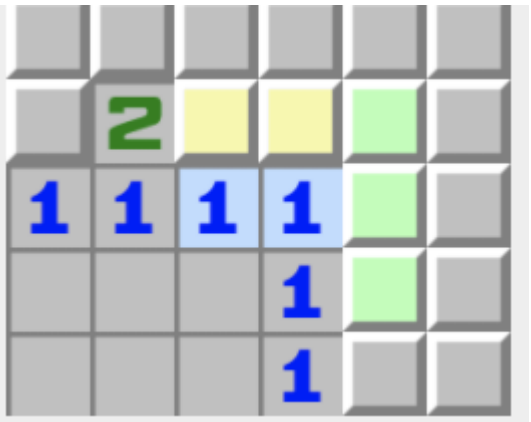
ראשית, על מנת להסביר ביצד אלגוריתם זה עובד, יש להבין מה הרעיון מאחוריו.
ישנם כמה דרכים ע"מ לדעת היכן ממוקמים מוקשים במשחק שולה מוקשים, לפי דרכים אלו, התוכנה נבנתה, כאשר ניתן לחשב לפי חיסור משמונה (האלגוריתם השלישי שהוצג), לפי הצבה ובדיקה לוגית (האלגוריתם הרביעי) או לפי ניחוש אקראי (האלגוריתם החמישי). קיימת עוד דרך שבה ניתן לפתור לוח, והיא לפי שינון דפוסים ומה ניתן להסיק מהם, אחד מהיתרונות של התוכנה, היא שהיא לא צריכה לשנן, מה שכותבים לה זה מה שהיא יודעת, ולכן, אלגוריתם זה עובד בצורה אפקטיבית בה.
הבסיס מאחוריו, הוא הסקה של מסקנות (ודאיות, או חצי וודאיות) על הלוח, לפי מציאה של דפוסים שמבעוד מועד אנחנו יודעים מה הם עושים.
לדוגמא:

הדפוס המופיע פה הוא גם הדפוס שמופיע בקוד למעלה, אך לפני שיוסבר הקוד עצמו, צריך להבין את הרעיון שעומד מאחורי הדפוס.
שמו של הדפוס הזה הוא +1-1.

נסתכל על האחד המסומן, לפי חוקי המשחק, אחד משני המוקשים שמעליו בטוח מוקשים:

עכשיו נסתכל על האחד שמימנו:

מכיוון ששני המספרים הם אחד, וישנה חפיפה בין המשבצות, ניתן להסיק שאחת משני המשבצות מכילה מוקש, הפרט היותר חשוב, הוא ששלוש המשבצות מימין לאחד בטוח לא מכילה מוקש, מכיוון שהמשבצות סביבו יכולות להכיל רק מוקש אחד. לכן, ניתן לסמן אותן בצבע ירוק:



העקרון חוזר על עצמו עם דפוסים אחרים, כאשר לפעמים מסיקים דברים לא וודאים אך כאלו שנותנים למשתמש אפשרות יותר סבירה לנצח, או כאלו שיתנו לנו מוקשים וודאיים ולא משבצות ירוקות.

בעת, ניתן לחזור לקוד:

```
function FindMinesPattern() {
    for (sh = 1; sh < Row - 1; sh++) {
        for (am = 1; am < Col - 1; am++) {
```

```
            if (Board[sh][am] == 1) {
                //#region 1-1+ DownLeft
                if (Board[sh + 1][am] == 1 && Board[sh + 2][am] == 1 &&
Board[sh + 2][am - 1] == 1 &&
                    Board[sh + 2][am - 2] == 1 && Board[sh + 2][am - 3]
== 1 && Board[sh + 2][am - 2] == 2) {
                    Board[sh + 1][am + 1] = 10;
                    Board[sh + 2][am + 1] = 10;
                    Board[sh + 3][am + 1] = 10;
                }
                //#endregion
```

האלגוריתם חוזר בצורה דומה עבור דפוסים אחרים, וגם עבור תצורות אחרות של הדפוס, כלומר, במידה והוא הפוך או מוסט לאחד הצדדים

הפונקציה תעבור על כל משבצת לא פינתית בלוח (ולכן תתחיל לולאת הפור מ-1), ותבדוק האם עבור המשבצות שמקיפות אותה מתקיים הדפוס, במידה וכן, תסמן את המשבצות הרלוונטיות במספר 10.

הפונקציה בקטעים הבאים בקוד, תבדוק את הדפוס גם על סיבוביו (סיבובים של 90 מעלות).

כך, כל שאילתה בפונקציה FindMinesPattern בודקת region שונה, כלומר, הפונקציה מחולקת לפי שאילתות, שכל אחת בודקת דפוס, כאשר חלק מן הפונקציות מבוססות על אותו דפוס, רק לאחר שהוא סובב.

קיים סוג דומה של דפוס, שבמקום מסקנות על אי-מיקוש, יסיק מסקנות על מיקוש:

בדומה מאוד לדפוס הקודם, הפעם ניתן להסיק שאחת מהמשבצות מעל הארבע היא מוקש, והשלושה שמשמאלו הם בטוח מוקשים. גם הדפוס הזה מופיע בקוד בצורה דומה לדפוס הקודם, אך הפעם הוא יסמן משבצות אדומות במקום ירוקות.

```
//#region 1-2+ UpLeft
if (Board[sh + 1][am - 1] == 1 && Board[sh + 1][am] == 1 &&
Board[sh + 1][am + 1] == 1 &&
                        Board[sh + 1][am + 2] == 4 && Board[sh +
2][am + 2] == 2 && Board[sh + 3][am + 2] == 1) {
                        Board[sh][am + 3] = 12;
                        Board[sh + 1][am + 3] = 12;
                        Board[sh + 2][am + 3] = 12;
                    }
```

לפי צורה דומה, קיימים גם דפוסים שיתנו לנו משבצות לא וודאיות, אך שכבל הנראה מכילות מוקש.

בסוף הפונקציה, לאחר שנבדקו כל הדפוסים, הפונקציה תשלח קריאה לשלוש פונקציות צבע (PaintGreen, PaintRed, PaintPink), פונקציה שצובעת בורוד מספרים שסומנו ב11, פונקציה שצובעת בירוק מספרים שסומנו ב10 (ואז מחזירה את הסימון שלהם ל-1) ופונקציה שצובעת באדום מספרים שסומנו ב9.
בסוף הפונקציה קיימת שאילתה, לאחר השליחה לפונקציות הצביעה, במידה ונצבעה בצורה בטוחה איזושהי משבצת, הפונקציה לא תשלח קריאה לפונקציה שבודקת בצורה רנדומלית, כלומר, התוכנה לא תגיע לפונקציות שבודקות בצורה אי וודאית במידה וניתן להסיק איזשהו נתון בצורה וודאית.