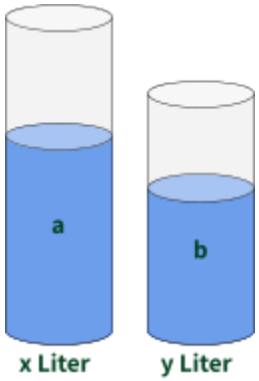
# בעיית הבקבוקים - Water Jug Problem




## תיאור הבעיה

- ☜ נתונים 2 בקבוקים בנפחים שונים, בקבוק **1** בנפח **x** ליטר ובקבוק **2** בנפח **y** ליטר.
- ☜ בקבוק 1 בתכולה **a** ובקבוק 2 בתכולה **b** ונייצג אותם ב-"טאפל" $(a,b)$ עם חשיבות לסדר.
- ☜ מה מספר הפעולות המינימלי ביותר כדי למלא את בקבוק 1 עם a ליטר ובקבוק 2 יהיה ריק.

## חוקים

1. ניתן לרוקן את הבקבוק עד סופו (לא ניתן לרוקן חלקית).
2. ניתן למלא את הבקבוק עד סופו (לא ניתן למלא חלקית).
3. ניתן למזוג מבקבוק אחד לשני עד שאחד הבקבוקים ריק או מלא.
4. אין הגבלה למצב ההתחלתי של כל בקבוק, כלומר בקבוק יכול להיות ריק או בכל מספר שלם של ליטר אפשרי.

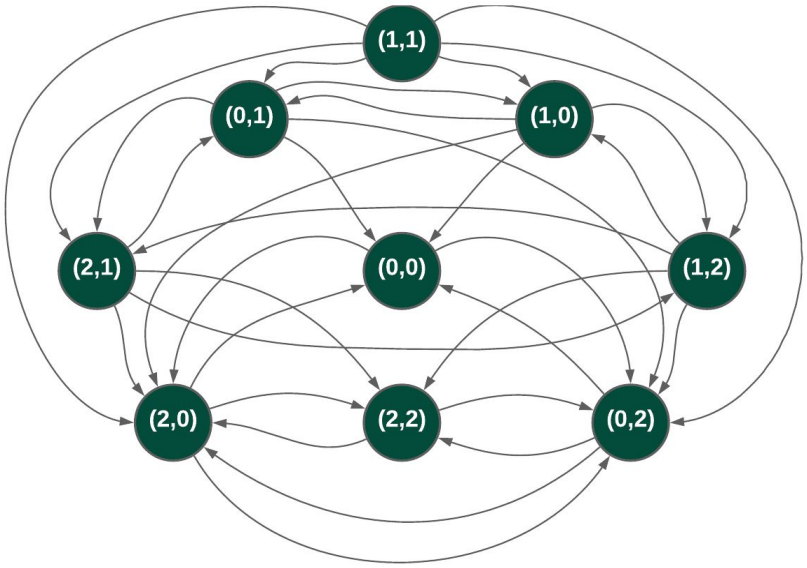
## דוגמה

נתונים 2 בקבוקים ריקים. בקבוק 1 עם $x = 5$ ליטר בקבוק 2 עם $y = 3$ ליטר.
נרצה למלא את בקבוק 1 עם $a = 4$ ליטר ובקבוק 2 יהיה ריק $b = 0$.

$$(0,0) \rightarrow (5,0) \rightarrow (2,3) \rightarrow (2,0) \rightarrow (0,2) \rightarrow (5,2) \rightarrow (4,3) \rightarrow (4,0)$$

נשים לב כי הפתרון הנל נפתר בצורה של גרף, לכן נייצג את בעיית הבקבוקים בגרף.
$B_{x,y} = (V_{x,y}\ ,\ E_{x,y})$ הוא גרף הבקבוקים, כאשר x הוא הנפח של בקבוק 1 ו-y הוא הנפח של בקבוק 2 כאשר x,y מספרים שלמים טבעיים.
הקודקודים מייצגים את כל המצבים האפשריים של הבקבוקים.
הצלעות מייצגות את כל המעברים החוקיים בין מצב קודקוד א' למצב קודקוד ב'.

ציור לדוגמה של הגרף $B_{2,2} = (V_{2,2}\ ,\ E_{2,2})$ המציג את כל המקרים של הבקבוקים:




אפשר לראות כמה הגרף לא קריא ומבולגן, ננסה לעבוד בצורה יותר נוחה.

נשאלת השאלה מה מספר הצלעות והקודקודים בגרף $B_{x,y}$?

נצייר מטריצה שתתאר לנו את כל המצבים האפשריים בגרף.

נעזר בגרף מהדוגמה הקודמת $B_{2,2} = (V_{2,2}\ ,\ E_{2,2})$:

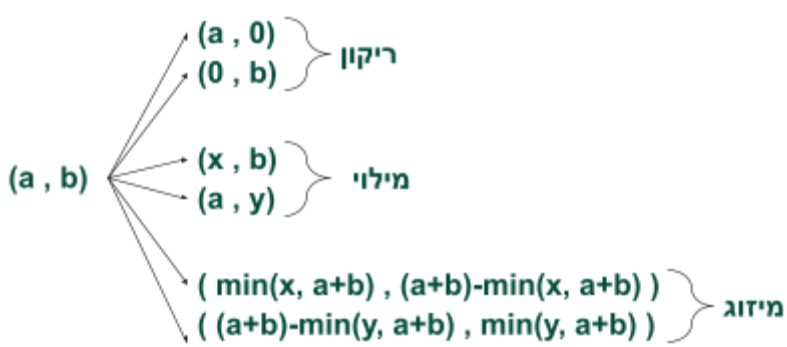
| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | (0,0) | (0,1) | (0,2) |
| 1 | (1,0) | (1,1) | (1,2) |
| 2 | (2,0) | (2,1) | (2,2) |

לפי המטריצה אפשר לראות כי מספר הקודקודים בגרף הוא: $\left|V_{2,2}\right| = 9$.

אם כך, עבור המקרה הכללי נקבל: $\left|V_{x,y}\right| = (x+1)\cdot(y+1)$.

מה מספר הצלעות בגרף?

נסתכל על קודקוד כלשהו ונבחן איזה צלעות אפשרויות יכולות לצאת ממנו $\deg_{\rightarrow}(a,b)$:



למשל עבור מקרה של מזיגה מבקבוק 1 ל-2 נבצע: $( (a+b) - min(y, a+b),\ min(y, a+b) )$
כאשר $a+b$ מייצג את כמות המים הכוללת של 2 הבקבוקים יחדיו.
עבור הבקבוק השני נבחר $min(y, a+b)$ וזאת כדי להבטיח שלא יקרה מקרה בו $y < a+b$ מאחר ו-y הוא הנפח המקסימלי בבקבוק השני.
עבור הבקבוק הראשון נבחר $(a+b) - min(y, a+b)$ כלומר כל מה שנשאר לנו בבקבוק הראשון לאחר המזיגה אל הבקבוק השני.

נחזור לשאלה מה מספר הצלעות בגרף? - ראינו 6 מקרים עבור קודקוד $(a, b)$, אבל לא בהכרח שמכל קודקוד בגרף יצאו 6 צלעות (לדוגמה הציור של הגרף בעמוד הקודם), יכול להיות קודקוד בדרגה יוצאת נמוכה יותר מ-6.
לכן נובע ש $\deg_{\rightarrow}(a, b) \le 6$ ומכאן:

$$\left|E_{x,y}\right| \le 6 \cdot \left|V_{x,y}\right| = 6 \cdot (x+1) \cdot (y+1)$$

את הגרף $B_{x,y}$ אפשר לייצג במחשב בכמה סוגים של מבני נתונים ושיטות פתרון.
נציג כעת את מטריצת השכנויות של הגרף $B_{2,2} = (V_{2,2}\ ,\ E_{2,2})$ בגודל $\left|V_{2,2}\right| x \left|V_{2,2}\right|$.
המטריצה מכילה אפסים כאשר כל עמודה ושורה מייצגת את הקשר בין 2 קודקודים באופן בינארי (True or False).
אם קיים קשר ישיר בין 2 קודקודים נסמן ב-1.

| | (0,0) | (0,1) | (0,2) | (1,0) | (1,1) | (1,2) | (2,0) | (2,1) | (2,2) |
|---|---|---|---|---|---|---|---|---|---|
| (0,0) | 0 | | 1 | | | | 1 | | |
| (0,1) | 1 | 0 | 1 | 1 | | | | 1 | |
| (0,2) | 1 | | 0 | | | | 1 | | 1 |
| (1,0) | 1 | 1 | | 0 | | 1 | 1 | | |
| (1,1) | | 1 | 1 | 1 | 0 | 1 | 1 | 1 | |
| (1,2) | | | 1 | 1 | | 0 | | 1 | 1 |
| (2,0) | 1 | | 1 | | | | 0 | | 1 |
| (2,1) | | 1 | | | | 1 | 1 | 0 | 1 |
| (2,2) | | | 1 | | | | 1 | | 0 |

מספר האחדות בעמודה הוא הדרגה הנכנסת (מספר הצלעות המצביעות) עבור הקודקוד המייצג את העמודה.
מספר האחדות בשורה הוא הדרגה היוצאת (מספר הצלעות היוצאות) עבור הקודקוד המייצג את השורה.

אנחנו משתמשים במטריצה דו מימדית, מה נשים באינדקסים של המטריצה? הם לא יכולים לכלול 2 קואורדינטות כמו שהצגנו כאן, אז איך נבצע המרה?

נגדיר:
i - גובה המים של בקבוק 1.
j - גובה המים של בקבוק 2.
col - מספר העמודות כלומר (y+1).
ומתקיים: $k = (col + 1) \cdot i + j$ כאשר k מייצג את התא.

**איך ולמה k?**

אנחנו ניצור בקוד מטריצה מהצורה:

```
this.matrix = new int[(x+1)*(y+1)][(x+1)*(y+1)];
```

נתבונן במטריצה לדוגמה שהצגנו, עבור השורה החמישית:

| index | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| .... | case | (0,0) | (0,1) | (0,2) | (1,0) | (1,1) | (1,2) | (2,0) | (2,1) | (2,2) |
| 4 | (1,1) | | 1 | 1 | 1 | 0 | 1 | 1 | 1 | |

אנחנו לא יכולים לבקש מהמטריצה את המקרה $(4, 6)$ כי הוא לא קיים בכלל בלולאה מאחר ו- x=2 וגם y=2.
נסתכל על המטריצה "דמיונית" הנראית כך:

| | | |
|---|---|---|
| (0,0) = 0 | (0,1) = 1 | (0,2) = 2 |
| (1,0) = 3 | (1,1) = 4 | (1,2) = 5 |
| (2,0) = 6 | (2,1) = 7 | (2,2) = 8 |

כל תא במטריצה מאפיין תרחיש חוקי בבעיה.
נרוץ בלולאה קנונית כך:

```
for(int i = 0; i <= this.x ; i++) {
  for(int j = 0 ; j <= this.y; j++) {
```

כל שורה במטריצה המקורית מייצגת לנו תרחיש (קודקוד בגרף) שממנו יוצאות צלעות ומסומנות ב-1 במידה ויש קשר ישיר, לכן לכל תרחיש במטריצה "הדימיונית" שאיתה אנו רצים עם הלולאות נפעיל עליה את כל 6 האפשרויות לצלעות יוצאות שהצגנו מקודם.

לכן בכל איטרציה בבניית המטריצה שלנו נחשב תחילה מול איזה תרחיש אנחנו עומדים `k = getIndex(i,j)`

כאשר getIndex מחזיר: `(this.y+1)*i + j` כאשר במקרה שלנו נקבל $k = (3+1)\cdot 2 + 0 = 6$

לאחר מכן נציב 1 בכל 6 המקרים לצלעות יוצאות עבור תרחיש במיקום ה-k הזה.

```
// Empty the 1st bottle
matrix[index][getIndex(0,j)] = 1;
// Empty the 2nd bottle
matrix[index][getIndex(i,0)] = 1;
```

וכן הלאה....

אפשר להסיק כי:

$i = k/(col + 1)$ כי זה נותן לנו בדיוק את השלם במקרה שלנו $i = 6/(2+1) = 6/3 = 2$

$j = k\%(col + 1)$ כי זה נותן לנו את שארית החלוקה במקרה שלנו $j = 6\%(2+1) = 6\%3 = 0$

מימוש בגיטהאב

**מימוש פתרון הבעיה**

```
public class WaterJug {

    int x, y, max, num_nodes;
    int[][] matrix;

    public WaterJug(int b1, int b2) {
        this.x = b1;
        this.y = b2;
        this.num_nodes = (b1+1)*(b2+1);
        this.matrix = new int[num_nodes][num_nodes];
        generate();
    }

    private int getIndex(int i, int j) {
        return (this.y+1)*i + j;
    }

    private void generate() {
        // java init arrays filling with '0'.
        for(int i = 0; i <= this.x ; i++) {
            for(int j = 0 ; j <= this.y; j++) {
                /*
                This index represents the tuple (i,j) of a scenario (node).
                 */
                int index = getIndex(i,j);

                /*
```

```
            Each scenario (node) could have at most 6 out-degrees edges.
             */

            // Empty the 1st bottle
            matrix[index][getIndex(0,j)] = 1;
            // Empty the 2nd bottle
            matrix[index][getIndex(i,0)] = 1;

            // Filling the 1st bottle
            matrix[index][getIndex(x,j)] = 1;
            // Filling the 2nd bottle
            matrix[index][getIndex(i,y)] = 1;

            // i+j means the amount of water of the 2 bottles together.
            // Pouring 1st to the 2nd
            matrix[index][getIndex((i+j)-Math.min(y,i+j), Math.min(y,i+j))] = 1;
            // Pouring 2nd to the 1st
            matrix[index][getIndex(Math.min(x,i+j), (i+j)-Math.min(x,i+j))] = 1;
        } // end inner for
    } // end main for

    /* To deny self-point of each node */
    for(int i = 0 ; i < this.num_nodes; i++) {
        this.matrix[i][i] = 0;
    }
}

public void printMatrix() {
    System.out.print("      ");
    for (int i = 0; i <= x; i++) {
        for (int j = 0; j <= y; j++) {
            System.out.print("(" + i + "," + j + ") ");
        }
    }
    System.out.println();
    int k = 0, l = 0;
    for (int i = 0; i < this.num_nodes; i++) {
        System.out.print("(" + k + "," + l + ") ");
        for (int j = 0; j < this.num_nodes; j++) {
            System.out.print("  " + matrix[i][j] + "   ");
        }
        System.out.println();
        if(l == y) {
            k++;
            l = 0;
        }
        else {
```

```
                l++;
            }
        }

    }

    public static void main(String[] args) {
        int bottle1 = 1;
        int bottle2 = 2;

        WaterJug waterJug = new WaterJug(bottle1,bottle2);
        waterJug.printMatrix();
    }
}
```