המסלול הקצר ביותר

הבעיה: בנבנו אלגוריתם המחשב את כמות המסלולים מנקודת (0,0) ל (m,n) (לכל הצלעות מחיר אחיד) ?

פתרון:

1. מגדירים מטריצה חבשה [n][m].
2. מאותחלים את הראשון ב-0.
3. מאותחלים שורה ועמודה ראשונה ב-1.
4. מחברים את התא מעל'ו עם התא השמאלי' הנמוך (לפ' הציור).
5. מחזירים את התא [n-1][m-1] - כמות המסלולים עבור [n][m].

סיבוכיות:

O(n * m).

<u>קוד:</u>

<u>הקוד הוא למטריצה ריבועית, זה לא בהכרח חייב להיות ריבוע.</u>

```
public static int numberOfEqualPaths(int n){
            int mat[][] = new int[n][n];
            for (int i=0; i<n; i++){ // אתחול
                  mat[i][0] = 1;
                  mat[0][i] = 1;
            }

            for (int i=1; i<n; i++){ // חישוב כמות המסלולים
                  for (int j=i; j<n; j++){
                        mat[i][j] = mat[i-1][j]+mat[i][j-1];
                        mat[j][i] = mat[i][j];
                  }
            }
            return mat[n-1][n-1];
}
```

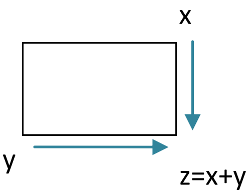
~~~~~~

<u>הבעיה:</u> למצוא אלגוריתם המחשב את כמות המסלולים מנקודת (0,0) ל (m,n) (<u>לכל צלע מחיר שווה</u>) ?

<u>פתרון:</u>

1. מגדירים מטריצה חדשה [m][n].
2. מאתחלים את הראשון ב-0.
3. מאתחלים שורה ועמודה ראשונה ב-1.
4. אם (התוצאות שווה)

Z מקבל את הסכום של 2 המסלולים.



אחרת

Z מקבל את הסכום של המסלול הקטן מבין ה-2.

מחזירים את התא [m-1][n-1] - כמות המסלולים עבור [m][n].
~~~~~~

קוד:

```
public int numberOfCheepestPaths(){
        mN[0][0].entry = 0;
        for (int i=1; i<n; i++){
            mN[0][i].entry = mN[0][i-1].entry +  mN[0][i-1].x;
            mN[0][i].numOfPaths = 1;
            mN[i][0].entry = mN[i-1][0].y+  mN[i-1][0].entry;
            mN[i][0].numOfPaths = 1;
        }
        for (int i=1; i<n; i++){
            for (int j=1; j<n; j++){
                int x = mN[i-1][j].entry+mN[i-1][j].y;
                int y = mN[i][j-1].entry+mN[i][j-1].x;
                if (x<y){
                    mN[i][j].entry = x;
                    mN[i][j].numOfPaths = mN[i-1][j].numOfPaths;
                }
                else      if (x>y) {
                        mN[i][j].entry = y;
                        mN[i][j].numOfPaths = mN[i][j-1].numOfPaths;
                    }
                    else{ //x==y
                        mN[i][j].numOfPaths =
                            mN[i][j-1].numOfPaths+mN[i-1][j].numOfPaths;
                        mN[i][j].entry = x; //x==y
                    }
            }
        }
        return mN[n-1][n-1].numOfPaths;
    }
```

סיבוכיות:

O(n * m).