# מציאת תת-מטריצה ריבועית גדולה ביותר המורכבת מאחדות בלבד

# נתונה מטריצה ובה אפסים או אחדות, יש למצוא את הריבוע הגדול ביותר של אחדות.

## **1.** בהינתן מערך המורכב מאפסים ואחדות בלבד יש למצוא את **תת-קטע (interval) ארוך ביותר המורכב מאחדות**

בלבד.

**דוגמה:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | address |
| 1 | 0 | 1 | 1 | 0 | **1** | **1** | **1** | 0 | 0 | 1 | 1 | 0 | 1 | 0 | array |

אורכו של מקטע אחדות הארוך ביותר הוא 3, מיקום האיבר הראשון במקטע הוא .5

**אלגוריתם חמדני:**

גודל המקטע שמתקבל באלגוריתם חמדני הוא 2, מיקום האיבר הראשון במקטע הוא 2, מסקנה: האלגוריתם

החמדני לא עובד כי 2<3.

**חיפוש שלם:**

נצטרך לעבור על כל התת-קטעים של המערך: נסמן ב[i,j]- קטע שמיקום האיבר הראשון הוא i ומיקום האיבר

האחרון הוא j , אורכו של הקטע הוא . j-i+1

[1,1] [1,2] . . . . . . . [1,n-1][1,n]

[2,2] . . . . . . . [2,n-1] [2,n]

. . . . . . . . . . . . . . . . . . . . . . . . . . . .

[n-1,n-1] [n-1,n]

[n,n]

מספר תתי-קטעים לפי סידרה חשבונית:

היא סיבוכיות מעבר על תת-קטע.

סיבוכיות של חיפוש שלם: , כאשר

**תכנון דינאמי:**

נגדיר מערך עזר help *באורך* ,n כך שכל איבר מייצג מספר אחדות הנמצאות בקטע שמסתיים במיקום האיבר:

דוגמה:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | address |
| 0 | 0 | 1 | 1 | 0 | **1** | **1** | **1** | 0 | 0 | 1 | 1 | 0 | 1 | 0 | array |
| 0 | 0 | 1 | 2 | 0 | **1** | **2** | **3** | 0 | 0 | 1 | 2 | 0 | 1 | 0 | help |

help[0] = array[0]

for i=1 to n-1

if (array[i]==1)

help[i] = help[i-1] + 1

else help[i] = 0

end-for

אורכו של מקטע אחדות הארוך ביותר maxLength יהיה שווה:

## maxLength = *max*(help[0], . . . , help[n-1]) = help[index]

בדוגמה לעיל 3 = *,*maxLength 7 = ,index טווח המקטע הוא

## .[index - maxLength+1, index]

**2**. בהינתן מערך דו-ממדי )מטריצה( mat[n,n] המורכב מאפסים ואחדות בלבד יש למצוא את **תת-המטריצה**

**הריבועית הגדולה ביותר המורכבת מאחדות** בלבד:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 0 | 0 |
| 1 | **1** | **1** | **1** | 0 |
| 0 | **1** | **1** | **1** | 1 |
| 1 | **1** | **1** | **1** | 0 |
| 1 | 1 | 0 | 0 | 1 |

**אלגוריתם חמדני:** לא עובד.

**חיפוש שלם:** נייצר את כל תתי הריבועים ונבדוק עבור כל ריבוע אם כולו מלא אחדות, אם כן, נבדוקהאם גודלו הכי גדול שמצאנו ואם כן נשמור את התשובה.

נטצרך לעבור על כל תתי-מטריצה ריבועיות.

**שיטה 1:**

מספר תת-מטריצות ריבועיות בגודל 1\*1 הוא **,**

מספר תת-מטריצות ריבועיות בגודל **2\*2 הוא ,**

מספר תת-מטריצות ריבועיות בגודל **3\*3 הוא ,**

**. ...**

. ...

מספר תת-מטריצות ריבועיות בגודל(n-1)\*(n-1) הוא ,

מספר תת-מטריצות ריבועיות בגודל n\*n הוא .

סה"כ מספר תתי-מטריצות ריבועיות הוא:

בדיקה האם תת-מטריצה ריבועית מורכבת מאחדות בלבד (לעבור על כל תת-ריבוע) לוקחת ,

**סה"כ סיבוכיות של חיפוש שלם** היא

**נכונות השיטה**: בודקים את כל האפשרויות ולכן בהכרח נגיע גם לתשובה הנכונה.

**PSEDO-CODE:**

**boolean checkForOne( [][]mat, iStart, jStart, iEnd, jEnd)**

boolean ans = true

for (int i=iStart; ans && i<=iEnd; i++)

for (int j = jStart; ans && j<=jEnd; j++) if (mat[i][j] == 0) ans = false

end-for end-for return ans

**end-checkForOne**

**///Exhaustive Search**

**int exhaustiveSearch([][]mat)**

n = mat.length, maxDim = 0 for (int i=0; i<n; i++)

for (int j=0; j<n; j++) if (mat[i][j]==1)

k = 1, p=i+k, q=j+k while (p<n && q<n)

if(*checkForOne*(mat,i,j,p,q))

if (k+1>maxDim) maxDim = k+1 end-if

k = k + 1, p = i + k, q = j + k end-while

end-if end-for

end-for

**end-exhaustiveSearch**

**שיטה 2:**

**נפעיל אלגוריתם חמדני על כל תא במטריצה:**

מספר תאים במטריצה הוא **.**

בדיקה עבור כל תא האם תת-מטריצה ריבועית מורכבת מאחדות בלבד לוקחת ,

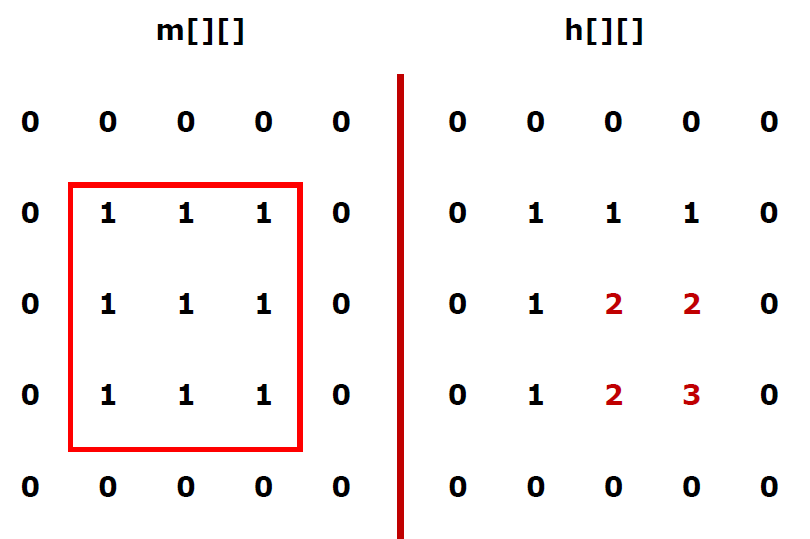
סה"כ סיבוכיות של חיפוש שלם היא

**תכנון דינאמי:**

נייצר מטריצה שבה בכל תאi,j יישמר אורך תת הריבוע שמסתיים ב-( i,j) כך שאורך תת הריבוע הגדול ביותר יישמר באחד מהתאים במטריצה – זו תהיה התשובה.

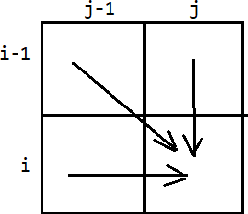
נבנה מטריצת עזר ,help[n,n] כאשר help[i,j] מהווה גודל של תת-מטריצה ריבועית המורכבת מאחדות שהפינה

הימנית התחתונה של התת-מטריצה נמצאת בתא עם קואורדינטות [[i,j



כדי לחשב את help[i,j] צריך לדעת את הערכים ,help[i-1,j-1] help[i,j-1],help[i-1,j]

בשורה הראשונה help[0,j] ובעמודה הראשונה help[i,0] גודל של ריבוע אחדות שווה ל- 1 כאשר mat[0,j]=1 או .mat[i,0]=1 לכן את השורה הראשונה ואת העמודה הראשונה של help זהות לשורה הראשונה ולעמודה הראשונה ב-.mat

תא [i,j] יכול להיות פינה ימנית תחתונה של התת-מטריצה

רק אם mat[i,j]=1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

במקרה זה גודל הריבוע לא יכול להיות גדול

מ-,1+help[i-1,j] מ ,1+help[i-1,j-1] -

מ ,1+help[i,j-1] –

כלומר help[i,j] = min(help[i-1,j-1], help[i-1,j], help[i,j-1]) + 1

למה min – כי גודל תת הריבוע יגדל רק אם 3 השכנים שלו היו 1 כדי שנקבל ריבוע מלא ואז התא

החדש יתוסף לריבוע שבו משתתפים כל 3 השכנים שלו ולכן ניקח את הריבוע עם הגודל המינימאלי .

כדי לקבל את הריבוע עצמו– נשמור תוך כדי המילוי את הגודל הגדול ביותר ואת מיקום התא שבו

נמצא המקסימום וזה הקודקוד הימני העליון של הריבוע.

**דוגמה:**

mat[i,j]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

help[i,j]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 2 |
| 0 | 0 | 1 | 2 | 2 |
| 0 | 1 | 1 | 2 | 3 |
| 1 | 1 | 2 | 2 | 3 |

**PSEDO-CODE:**

**int subMatrixOf1(int [][] m)**

n = mat.length, maxDim = 0 ,

iMax = 0, jMax = 0

help[n,n]

**//copy first row**

for (int j = 0; j < n; j++)

help [0,j] = mat[0,j]

end-for

**//copy first column**

for (int i = 0; i < n; i++)

help [I,0] = mat[i,0]

end-for

**// get maxDim**

for (int i = 1; i < n ; i++)

for (int j = 1; j < n ; j++)

if (mat[i,j] == 1){

help [i,j] = *min3*(help [i-1][j-1], help [i-1][j], help [i][j-1]) + 1

if (h[i,j] > maxDim)

maxDim = help [i,j]

iMax = i

jMax = j

end-if

end-for

end-for

return maxDim

**end-subMatrixOf1**

**// get min of three numbers**

**int min3(a, b, c)**

min = a

if (min > b) min = b

if (min > c) min = c

return min

**end-min3**

**סיבוכיות של תכנון דינאמי:**