Algorithm Floyd-Warshall

**אלגוריתם פלויד־וורשל** הוא אלגוריתם למציאת כל משקלי המסלולים הקלים/הקצרים ביותר בין כל זוגות הצמתים בגרף. אלגוריתם זה פועל עבור גרף מכוון משוקלל בעל משקלים (אורכי קשתות) חיוביים ושליליים, חסר מעגלים שליליים. האלגוריתם מבוסס על פרדיגמת התכנון הדינמי.

**k - קודקוד מתווך**

**k**

**i j**

**………….**

**Constructing a shortest path**

**for** (**int** k = 0; k < n; k++)

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

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

) **נוסחה (**

**נוסחה**

**if** (mat[i][k]!=*inf* && mat[k][j]!=*inf*){

**if** (mat[i][j] > mat[i][k]+mat[k][j]){

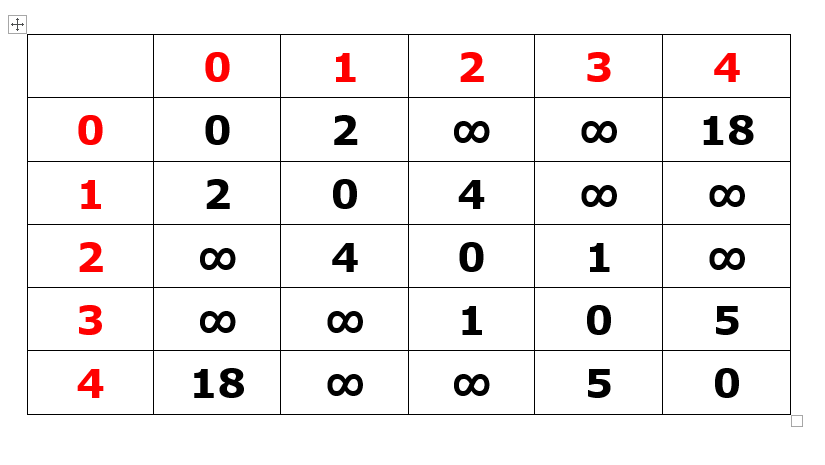
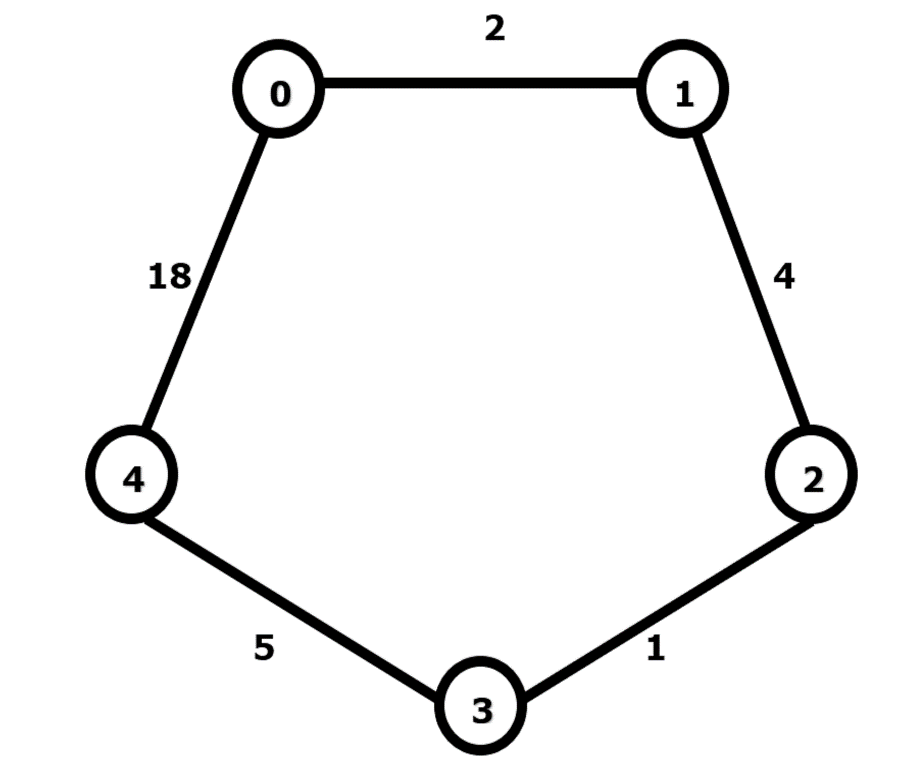
mat[i][j] = mat[i][k]+mat[k][j];

pathMat[i][j] = pathMat[i][k]+pathMat[k][j];

}

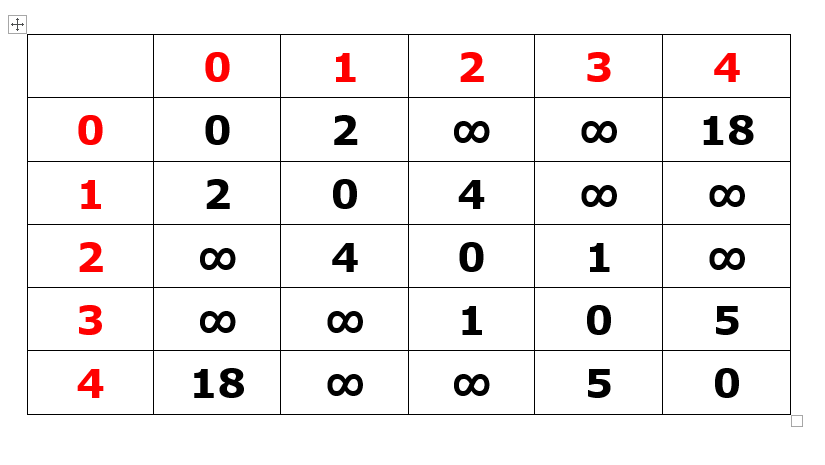
}

ממטריצת שכנות ← למטריצת מסלולים



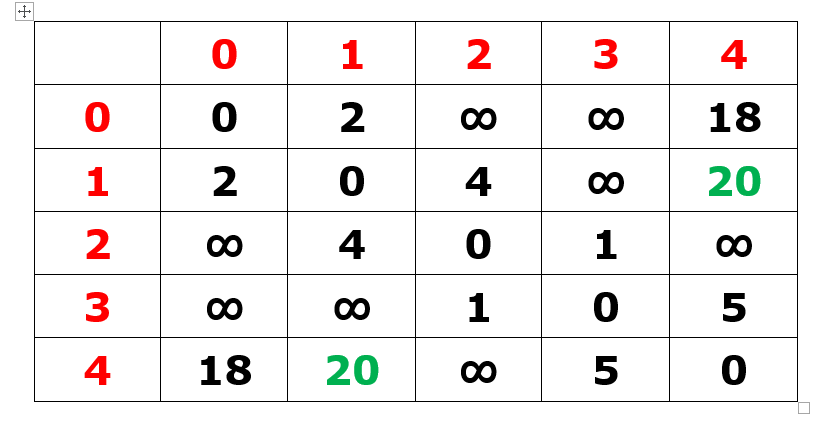
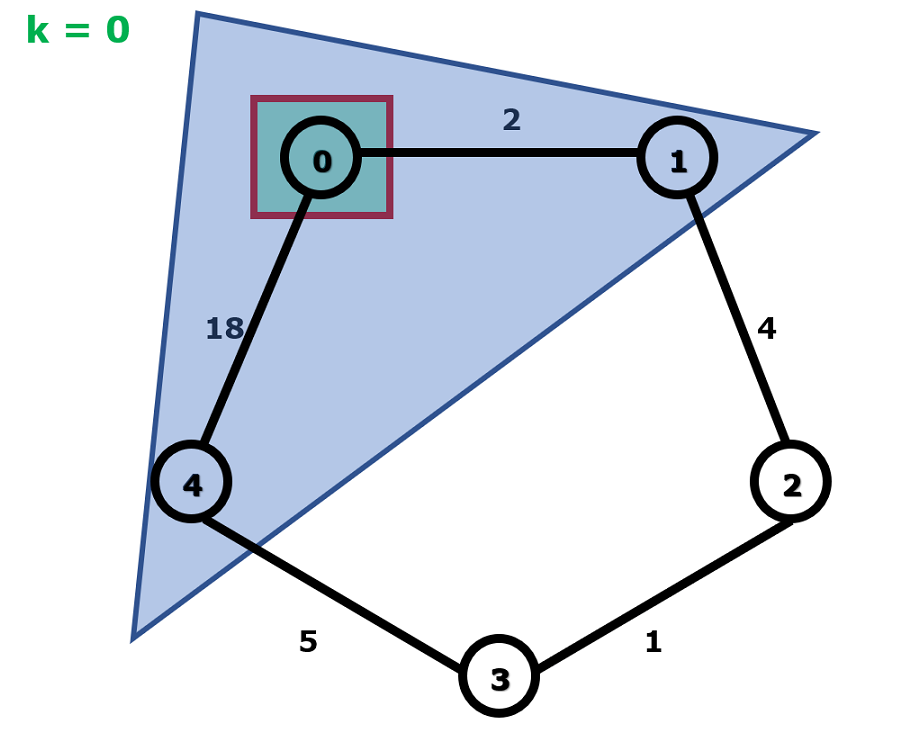
מטריצת שכנות

מטריצת שכנות



מטריצת מסלולים

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** |
| **0** | **0🡪0** | **0🡪1** |  |  | **0🡪4** |
| **1** | **1🡪0** | **1🡪1** | **1🡪2** |  |  |
| **2** |  | **2🡪1** | **2🡪2** | **2🡪3** |  |
| **3** |  |  | **3🡪2** | **3🡪3** | **3🡪4** |
| **4** | **4🡪0** |  |  | **4🡪3** | **4🡪4** |



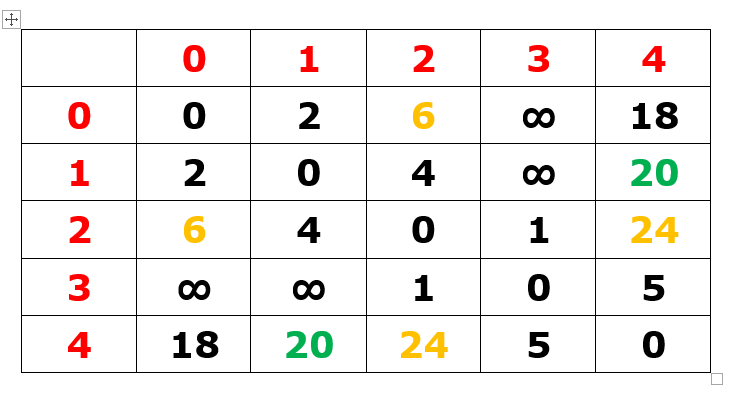
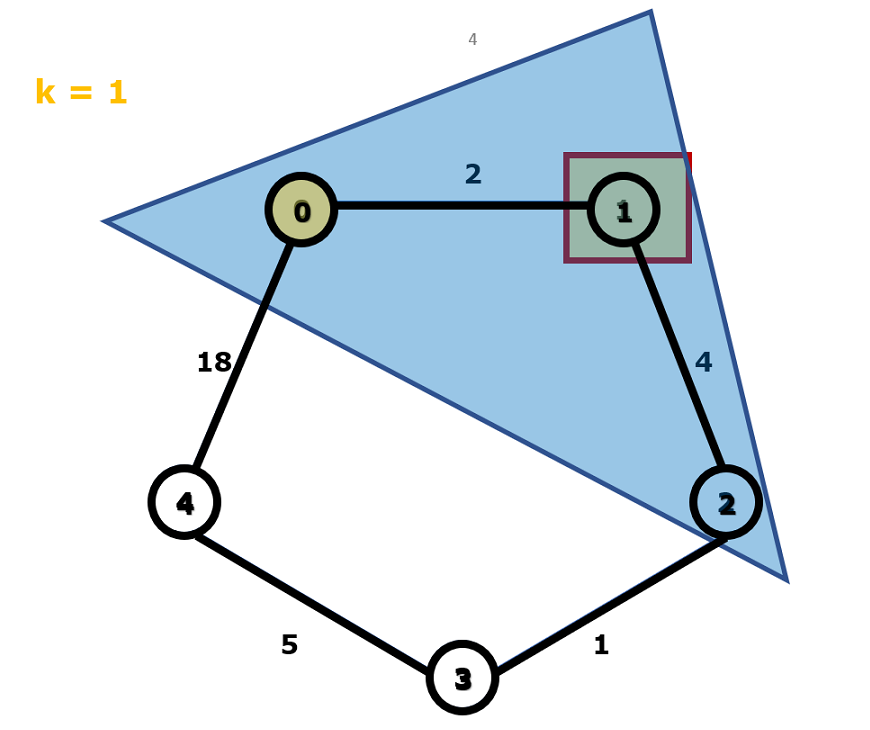
**K = 0**

**Distance (4🡪0🡪1) = 18 + 2 = 20**

**Distance (1🡪0🡪4) = 18 + 2 = 20**

מטריצת מסלולים

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** |
| **0** | **0🡪0** | **0🡪1** |  |  | **0🡪4** |
| **1** | **1🡪0** | **1🡪1** | **1🡪2** |  | **1🡪0🡪4** |
| **2** |  | **2🡪1** | **2🡪2** | **2🡪3** |  |
| **3** |  |  | **3🡪2** | **3🡪3** | **3🡪4** |
| **4** | **4🡪0** | **4🡪0🡪1** |  | **4🡪3** | **4🡪4** |

****

**K = 1**

**Distance (0🡪1🡪2) = 2 + 4 = 6**

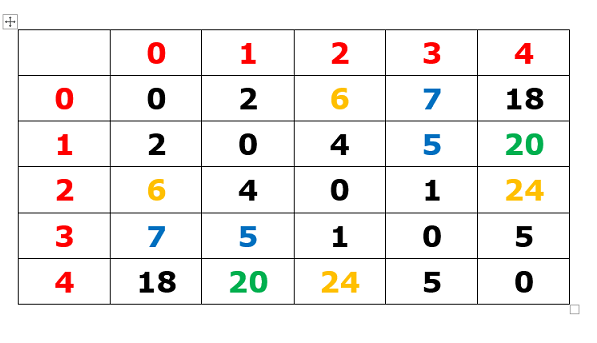
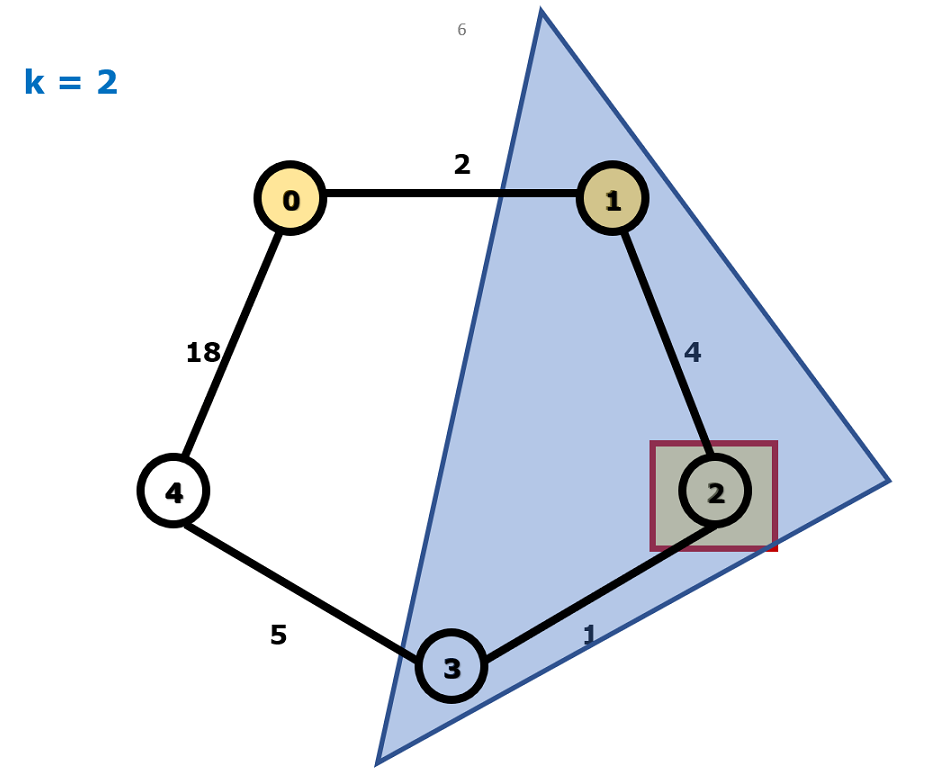
**Distance (2🡪1🡪0) = 4 + 2 = 6**

**Distance (4🡪0🡪1🡪2) = 18 + 2 + 4 = 24**

**Distance (2🡪1🡪0🡪4) = 4 + 2 + 18 = 24**

מטריצת מסלולים

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** |
| **0** | **0🡪0** | **0🡪1** | **0🡪1🡪2** |  | **0🡪4** |
| **1** | **1🡪0** | **1🡪1** | **1🡪2** |  | **1🡪0🡪4** |
| **2** | **2🡪1🡪0** | **2🡪1** | **2🡪2** | **2🡪3** | **2🡪1🡪0🡪4** |
| **3** |  |  | **3🡪2** | **3🡪3** | **3🡪4** |
| **4** | **4🡪0** | **4🡪0🡪1** | **4🡪0🡪1🡪2** | **4🡪3** | **4🡪4** |



**K = 2**

**Distance (1🡪2🡪3) = 4 + 1 = 5**

**Distance (3🡪2🡪1) = 1 + 4 = 5**

**Distance (0🡪1🡪2🡪3) = 2 + 4 + 1 = 7**

**Distance (3🡪2🡪1🡪0) = 1 + 4 + 2 = 7**

**Distance (4🡪0🡪1🡪2🡪3) = 18 + 2 + 4 + 1 = 25**

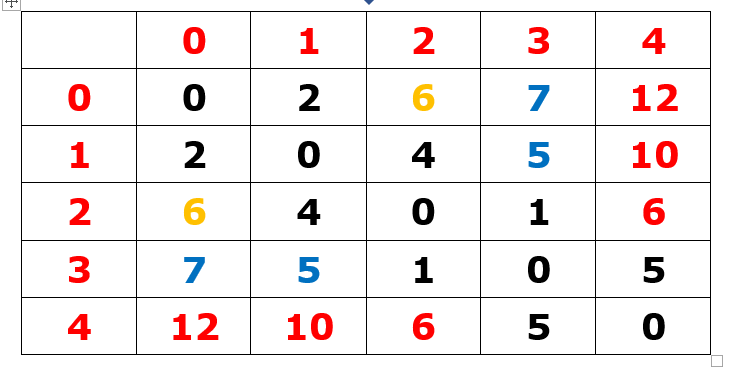
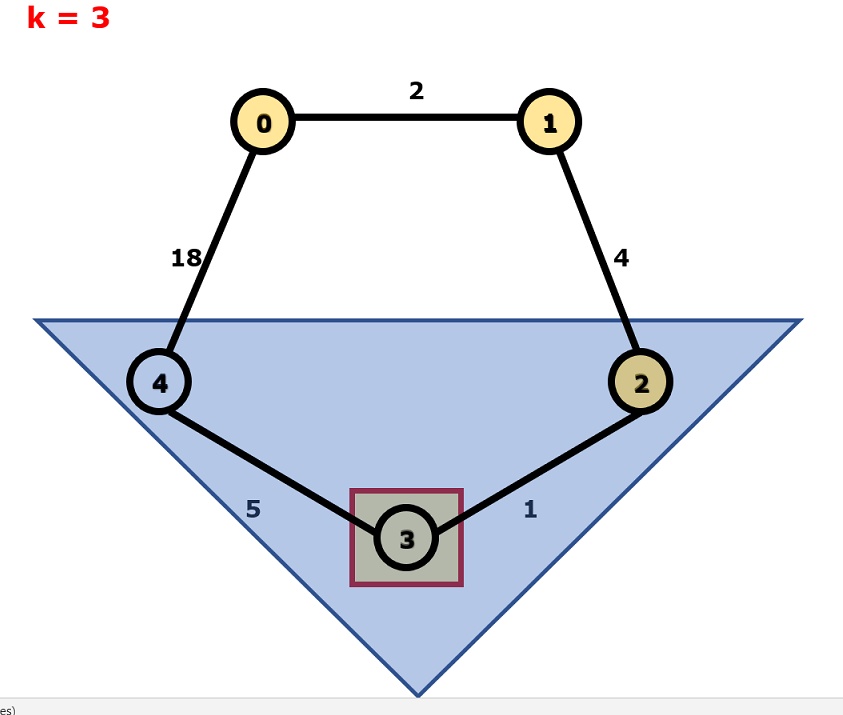
**Distance (3🡪2🡪1🡪0🡪4) = 1 + 4 + 2 + 18 = 25**

**Distance (4🡪3) = 5**

**Distance (3🡪4) = 5**

מטריצת מסלולים

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** |
| **0** | **0🡪0** | **0🡪1** | **0🡪1🡪2** | **0🡪1🡪2🡪3** | **0🡪4** |
| **1** | **1🡪0** | **1🡪1** | **1🡪2** | **1🡪2🡪3** | **1🡪0🡪4** |
| **2** | **2🡪1🡪0** | **2🡪1** | **2🡪2** | **2🡪3** | **2🡪1🡪0🡪4** |
| **3** | **3🡪2🡪1🡪0** | **3🡪2🡪1** | **3🡪2** | **3🡪3** | **3🡪4** |
| **4** | **4🡪0** | **4🡪0🡪1** | **4🡪0🡪1🡪2** | **4🡪3** | **4🡪4** |



**Distance (2🡪3🡪4) = 1 + 5 = 6**

**Distance (4🡪3🡪2) = 5 + 1 = 6**

**K = 3**

**Distance (1🡪2🡪3🡪4) = 4 + 1 + 5 = 10**

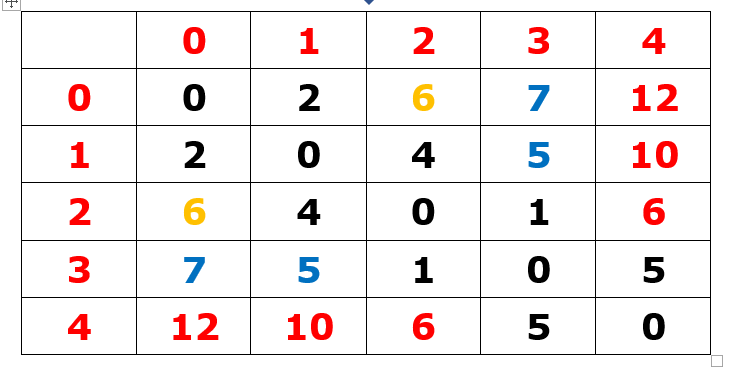
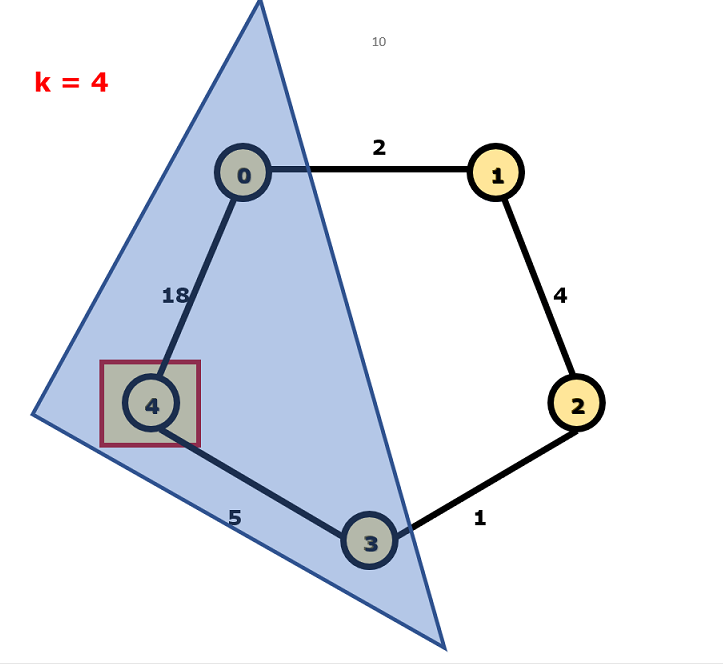
**Distance (4🡪3🡪2🡪1) = 5 + 1 + 4 = 10**

**Distance (0🡪1🡪2🡪3🡪4) = 2 + 4 + 1 + 5 = 12**

**Distance (4🡪3🡪2🡪1🡪0) = 5 + 1 + 4 + 2 = 12**

מטריצת מסלולים

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** |
| **0** | **0🡪0** | **0🡪1** | **0🡪1🡪2** | **0🡪1🡪2🡪3** | **0🡪1🡪2🡪3🡪4** |
| **1** | **1🡪0** | **1🡪1** | **1🡪2** | **1🡪2🡪3** | **1🡪2🡪3🡪4** |
| **2** | **2🡪1🡪0** | **2🡪1** | **2🡪2** | **2🡪3** | **2🡪3🡪4** |
| **3** | **3🡪2🡪1🡪0** | **3🡪2🡪1** | **3🡪2** | **3🡪3** | **3🡪4** |
| **4** | **4🡪3🡪2🡪1🡪0** | **4🡪3🡪2🡪1** | **4🡪3🡪2** | **4🡪3** | **4🡪4** |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** |
| **0** | **0🡪0** | **0🡪1** | **0🡪1🡪2** | **0🡪1🡪2🡪3** | **0🡪1🡪2🡪3🡪4** |
| **1** | **1🡪0** | **1🡪1** | **1🡪2** | **1🡪2🡪3** | **1🡪2🡪3🡪4** |
| **2** | **2🡪1🡪0** | **2🡪1** | **2🡪2** | **2🡪3** | **2🡪3🡪4** |
| **3** | **3🡪2🡪1🡪0** | **3🡪2🡪1** | **3🡪2** | **3🡪3** | **3🡪4** |
| **4** | **4🡪3🡪2🡪1🡪0** | **4🡪3🡪2🡪1** | **4🡪3🡪2** | **4🡪3** | **4🡪4** |

מטריצת מסלולים

**static** **int** *inf* = Integer.***MAX\_VALUE***;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **static** **int**[][] initInt(){

**int** [][] mat = {{ 0, 2, *inf*, *inf*, 18},

{ 2, 0, 4, *inf*, *inf*},

{ *inf*, 4, 0, 1, *inf*},

{ *inf*, *inf*, 1, 0, 5},

{ 18, *inf*, *inf*, 5, 0}};

**return** mat;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **static** **void** printIntMatrix(**int**[][] mat){

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

**for**(**int** i=0; i<mat.length; i++){

**for**(**int** j=0; j<mat[0].length; j++){

**if** (mat[i][j] == *inf*)

System.***out***.format("%6s,","inf");

**else**

System.***out***.format("%6d,",mat[i][j]);

}

System.***out***.println();

}

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **static** **void** printPath(String [][]pathMat, **int**[][] mat){

System.***out***.println();

**for**(**int** i=0; i<pathMat.length; i++){

**for**(**int** j=0; j<pathMat[0].length; j++){

**if** (mat[i][j] == *inf*) System.***out***.print("\*, ");

**else** System.***out***.printf("%25s",pathMat[i][j]+"; ");

}

System.***out***.println();

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **static** String [][] buildPathMatrix(**int** [][] mat){

**int** n = mat.length;

// path matrix initialization

String [][] pathMat = **new** String[n][n];

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

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

**if** (mat[i][j] != *inf*)

pathMat[i][j] = " "+**i**+"**->**"+**j**+" ";

**else**

pathMat[i][j] = "";

}

}

// path matrix building

........

........

........

{

**משימה:**

לייצר כל מסלולים בין כל זוגות הצמתים בגרף לפי אלגוריתם פלויד־וורשל – להשלים את הפונקציה:

**public** **static** String [][] buildPathMatrix(**int** [][] mat)