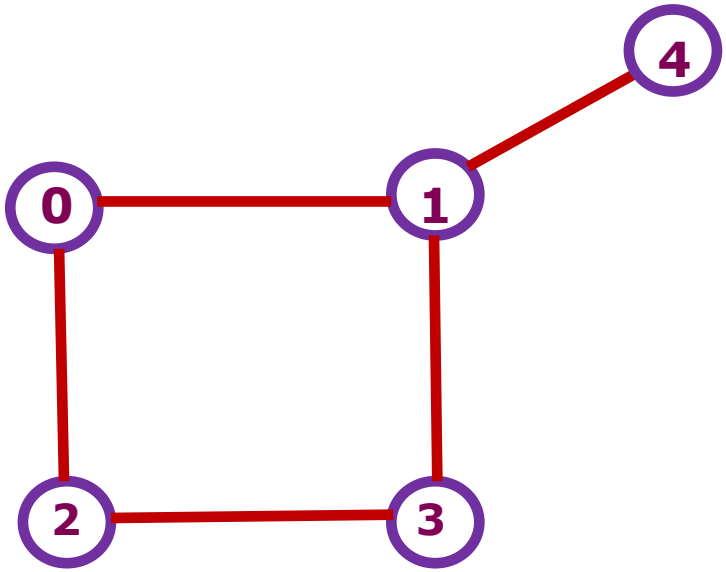
Floyd – Warshall algorithm

1. בניית מטריצת שכנות בוליאני (גרף לא מכוובן)

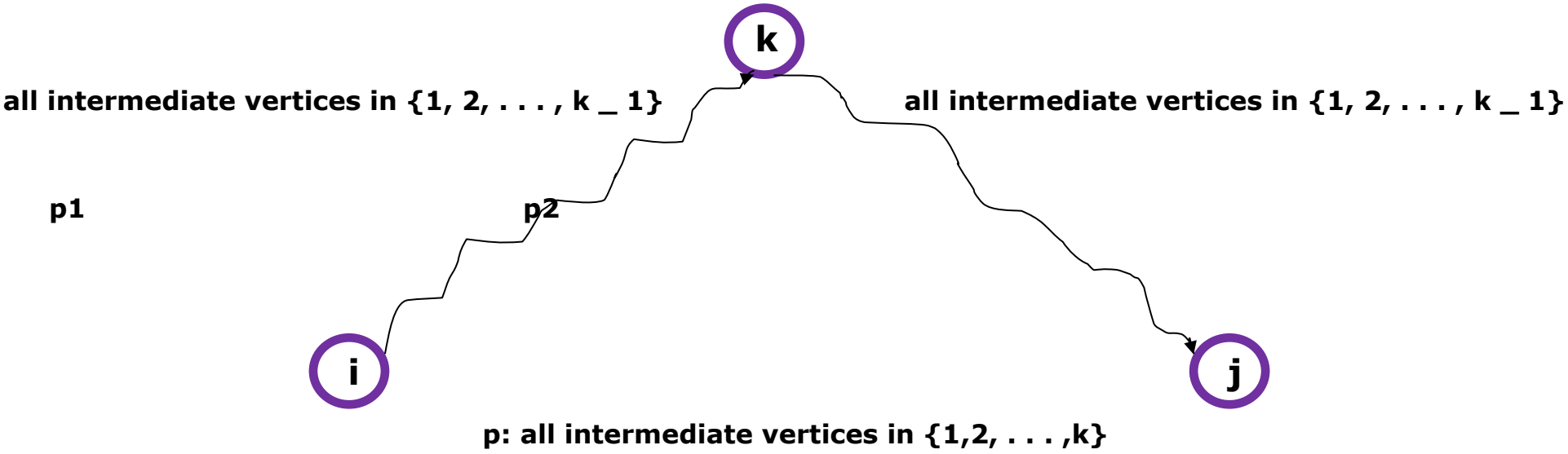


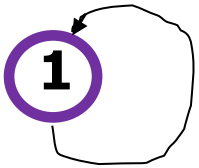
מטריצת שכנות - adjacency matrix

0	0	1	2	3	4
0	false,	**true**,	**true**,	false,	false,
1	**true**,	false,	false,	**true**,	**true**,
2	**true**,	false,	false,	**true**,	false,
3	false,	**true**,	**true**,	false,	false,
4	false,	**true**,	false,	false,	false,

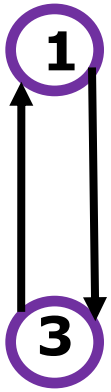
2. אלגוריתם FloydWarshall

k - קודקוד מתווך





false



true

```
for (int k = 0; k < n; k++)
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            . . . . . . . . .        ( נוסחה )
```

(נוסחה)

mat[i,j] ← mat[i,j] or (mat(i,k) and mat(k,j))

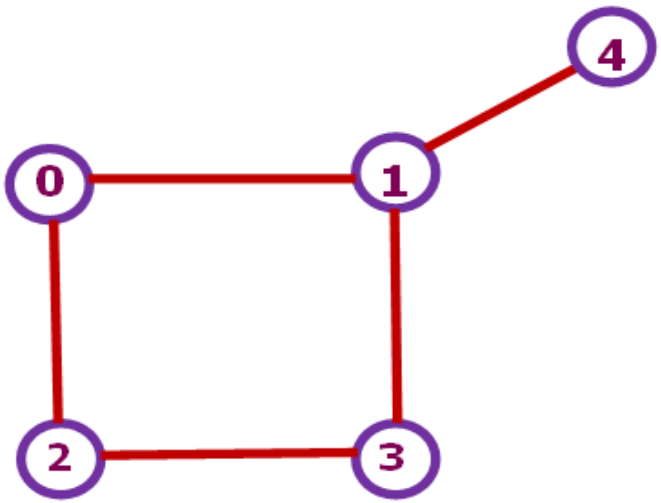
מטריצה שכנות בוליאני אחרי FloydWarshall

(מטריצה קשירות)

0	0	1	2	3	4
0	true,	true,	true,	true,	true,
1	true,	true,	true,	true,	true,
2	true,	true,	true,	true,	true,
3	true,	true,	true,	true,	true,
4	true,	true,	true,	true,	true,

3. בניית מטריצת מסלולים

n = 5



0	0	1	2	3	4
0	false,	**true**,	**true**,	false,	false,
1	**true**,	false,	false,	**true**,	**true**,
2	**true**,	false,	false,	**true**,	false,
3	false,	**true**,	**true**,	false,	false,
4	false,	**true**,	false,	false,	false,

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

	0	1	2	3	4
0	"",	"",	"",	"",	"",
1	"",	"",	"",	"",	"",
2	"",	"",	"",	"",	"",
3	"",	"",	"",	"",	"",
4	"",	"",	"",	"",	"",

מטריצת מסלולים (אתחול)

קלט

0	0	1	2	3	4
0	false,	**true**,	**true**,	false,	false,
1	**true**,	false,	false,	**true**,	**true**,
2	**true**,	false,	false,	**true**,	false,
3	false,	**true**,	**true**,	false,	false,
4	false,	**true**,	false,	false,	false,

0	1	2	3	4	
0	" ",	[0→1],	[0→2],	" ",	" ",
1	[1→0],	" ",	" ",	[1→3],	[1→4],
2	[2→0],	" ",	" ",	[2→3],	" ",
3	" ",	[3→1],	[3→2],	" ",	" ",
4	" ",	[4→1],	" ",	" ",	" ",

מטריצת מסלולים (אלגוריתם FloydWarshall)

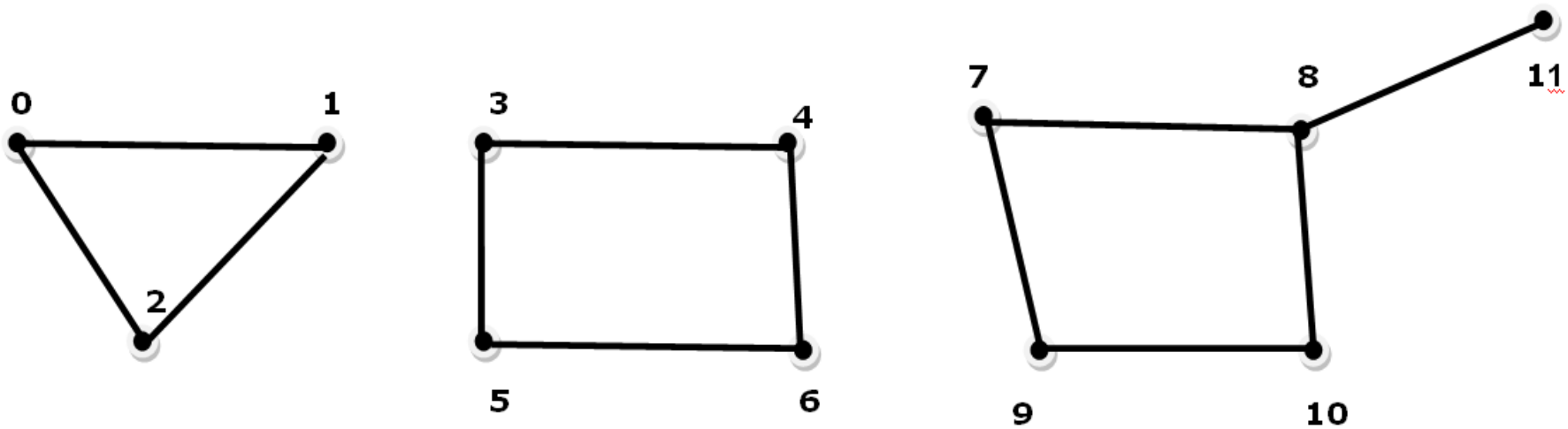
פלט

	0	1	2	3	4
0	[0->1 1->0],	**[0→1]**,	**[0→2]**,	**[0->1 1->3]**,	[0->1 1->4]
1	**[1→0]**,	**[1->0 0->1]**,	[1->0 0->2],	**[1→3]**,	**[1→4]**
2	**[2→0]**,	**[2->0 0->1]**,	[2->0 0->2],	**[2→3]**,	[2->0 0->1 1->4]
3	[3->1 1->0],	**[3→1]**,	**[3→2]**,	**[3->1 1->3]**,	[3->1 1->4]
4	[4->1 1->0],	**[4→1]**,	[4->1 1->0 0->2],	**[4->1 1->3]**,	[4->1 1->4]

שלב 1 - בניית מסלולים למטריצה שחנות

שלב 2 - בניית מסלולים למטריצה קשירות (FloydWarshall)

4. מספר רכיבי קשירות



גרף אחד : 12 קודקודים

12 קשתות (צלעות)

קלט

	0	1	2	3	4	5	6	7	8	9	10	11
0	F	T	T	F	F	F	F	F	F	F	F	F
1	T	F	T	F	F	F	F	F	F	F	F	F
2	T	T	F	F	F	F	F	F	F	F	F	F
3				F	T	T	F	F	F	F	F	F
4				T	F	F	T	F	F	F	F	F
5				T	F	F	T	F	F	F	F	F
6				F	T	T	F	F	F	F	F	F
7								F	T	T	F	F
8								T	F	F	T	T
9								T	F	F	T	F
10								F	T	T	F	F
11								F	T	F	F	F

פלט

	0	1	2	3	4	5	6	7	8	9	10	11
0	T	T	T	F	F	F	F	F	F	F	F	F
1	T	T	T	F	F	F	F	F	F	F	F	F
2	T	T	T	F	F	F	F	F	F	F	F	F
3				T	T	T	T	F	F	F	F	F
4				T	T	T	T	F	F	F	F	F
5				T	T	T	T	F	F	F	F	F
6				T	T	T	T	F	F	F	F	F
7								T	T	T	T	T
8								T	T	T	T	T
9								T	T	T	T	T
10								T	T	T	T	T
11								T	T	T	T	T

	0	1	2	3	4	5	6	7	8	9	10	11				connectComp[]
0	T	T	T	F	F	F	F	F	F	F	F	F			1	
1	T	T	T	F	F	F	F	F	F	F	F	F			1	
2	T	T	T	F	F	F	F	F	F	F	F	F			1	
3				T	T	T	T	F	F	F	F	F			**2**	
4				T	T	T	T	F	F	F	F	F			**2**	
5				T	T	T	T	F	F	F	F	F			**2**	
6				T	T	T	T	F	F	F	F	F			**2**	
7								T	T	T	T	T			**3**	
8								T	T	T	T	T			**3**	
9								T	T	T	T	T			**3**	
10								T	T	T	T	T			**3**	
11								T	T	T	T	T			**3**	

numComponents = 1

5. בדיקת מטריצה
(אם מטריצה רכיב קשירות אחד או לא)

אם מטריצה - רכיב קשירות אחד אז כל האיברים של מטריצה חייבים להיות **TRUE**, אחרת יש קודקודים שאין מסלולים בין קודקודים האלה.

זה אומר שבבדיקת מטריצה בוליאני **FALSE** הראשון נותן תשובה שלילית לשרלה.