

הסבר תרגיל 2 אימות פורמלי:

כדי להכניס קשת (b,a) נציין אותה כ- a b כאשר כל קשת תופר עם פסיקים בדיוק כמו שנכתב במסמך המאפיין של Ex2 כלומר:

```
Please enter your graph or EXIT if you want to exit the software
1 2,3 4,4 5
```

הפלט יראה כך:

```
['x1'], ['x2'], ['x3'], ['x4'], ['x5'],
```

שינוי אחד הוא שלא ניתן להוסיף 0 כצומת מאחר ש-0 משמש להכנסת צומת שאין לא קשתות כלומר:

```
Please enter your graph or EXIT if you want to exit the software
0 1,0 2,3 0,4 0
['x1'], ['x2'], ['x3'], ['x4'],
```

0 a וההפך

בעצם אומר שצומת a הוא חלק מהגרף גם אם אין לו קשתות בכלל כמובן שצירוף של 0 a וההפך לא יפגע ב- a אם המשתמש יכניס קשתות ש- a משתתף בהם לדוגמא:

```
Please enter your graph or EXIT if you want to exit the software
1 0,2 1,1 2
['x1', 'x2'],
```

ליציאה הקלד EXIT

```
Please enter your graph or EXIT if you want to exit the software
EXIT
```

כמה דברים על הקוד:

1) הקוד מממש את הפוסאדו קוד שנראה בהרצאה 3 אחד לאחד כלומר בניתי 2 פונקציות DFS-A שמבצע על גרף G את אלגוריתם DFS, כמו כן בניתי את DFS-B שמבצע את האלגוריתם DFS על הגרף G ההפוך.

2) בניתי 2 מחלקות:

1. vertex - מייצגת צמת ומכילה את התכונות:

א..שם של הצומת -הערך המספרי שלה.

ב.הקשתות בגרף G - לדעת את השכנים שלו בגרף G .

ג.הקשתות בגרף G ההפוך - לדעת את השכנים בגרף G ההפוך.

ד..צומת חדשה/ישנה תכונה זאת תשמש לדעת אם עברנו על הצומת במהלך ה- dfs .

2. edge - מייצגת קשת ומכילה את התכונות:

א.המספר של הצומת ממנה יוצאת.

ב.המספר של הצומת אליה נכנסת.

3)הקוד מתחיל בכך שהוא מקבל את string שהמשתמש הכניס בודק אם תקין מבחינת התבנית שהכנסנו, אם כן מפרק אותו לחלקים באמצעות פסיקים לאחר מכן עובר על כל מקטע ובונה את הקשתות והצמתים הרלוונטים שלנו,

לאחר מכן מבצע dfs על G כמו הפוסאדו קוד בהרצא:

```
DFS-A ( $G(V, E)$ ,  $h$ )
for every  $e \in E$  mark  $e$  "new"
for every  $u \in V$  do {
     $f(u) \leftarrow \text{NULL}$ 
    mark  $u$  "new"
}
 $i \leftarrow 1$ 
```

Strongly Connected Components

```
while there are new vertices do {
    let  $v$  be such a vertex
    mark  $v$  "old"
    while  $v$  has new outgoing edges or  $f(v) \neq \text{NULL}$  do {
        if  $v$  has new outgoing edges then do {
            let  $v \xrightarrow{e} w$  be a new edge
            mark  $e$  "old"
            if  $w$  is new then do {
```

Strongly Connected Components

```
        mark  $w$  "old"
         $f(w) \leftarrow v$ 
         $v \leftarrow w$ 
    }
}
else ( $f(v) \neq \text{NULL}$ ) then do {
     $h(v) \leftarrow i$ 
     $i \leftarrow i + 1$ 
     $v \leftarrow f(v)$ 
}
```

Strongly Connected Components

```
}
 $h(v) \leftarrow i$ 
 $i \leftarrow i + 1$ 
}
DFS-A labels vertices  $h(v)$  by finishing
visit order
```

אחר מכן מבצע dfs על G ההפוך כמו הפוסאדו קוד בהרצאה

```

DFS-B ( $G(V, E^R), h$ )
for every  $e \in E^R$  mark  $e$  "new"
for every  $u \in V$  do {
     $f(u) \leftarrow \text{NULL}$ 
    mark  $u$  "new"
}

```

Strongly Connected Components

```

while there are new vertices do {
    let  $v$  be a new vertex for which  $f(v)$  is maximal
     $S \leftarrow \{v\}$ 
    mark  $v$  "old"
    while  $v$  has new outgoing edges or  $f(v) \neq \text{NULL}$  do {
        if  $v$  has new outgoing edges then do {
            let  $v \xrightarrow{e} w$  be a new edge
            mark  $e$  "old"
            if  $w$  is new then do {

```

Strongly Connected Components

```

        mark  $w$  "old"
         $f(w) \leftarrow v$ 
         $S \leftarrow S \cup \{w\}$ 
         $v \leftarrow w$ 
    }
    else ( $f(v) \neq \text{NULL}$ ) then do {
         $v \leftarrow f(v)$ 
    }
    print set  $S$  is an SCC
}

```

לאחר מציאת רכיבי הקשירות אנו קוראים לפונקציה שמדפיסה את הפלט, היא מוסיפה X לכל השמות.

(4) כל הקוד מכיל הערות להבנה של היישום.