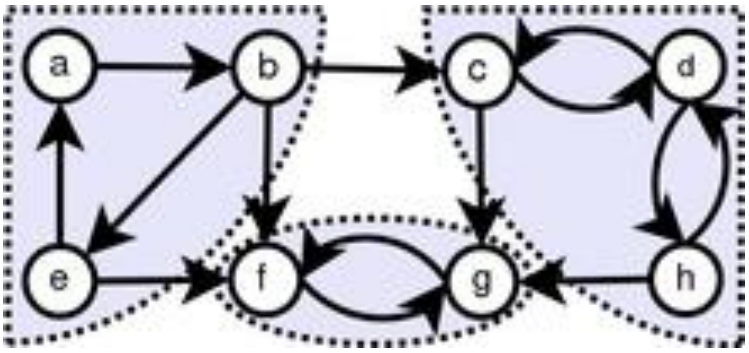
Tarjan's strongly connected components algorithm

רכיב קשיר היטב - גרף מכוון נקרא **קשיר היטב (או קשיר בחוזקה)** אם קיים מסלול מכל צומת שבו אל כל צומת אחר.

עבור גרף **G=(V,E)** מכוון כללי, ניתן תמיד לפרק את הגרף לרכיבים קשירים היטב -תתי-גרפים מקסימליים של הגרף המקורי (גם: רק"ח - רכיבי קשירות חזקה), שכל אחד מהם הוא גרף קשיר היטב בפני עצמו. פירוק זה מהווה חלוקה של הגרף למחלקות זרות - שני רכיבים שונים לא יכולים להכיל צומת משותף.



אלגוריתם נאיבי

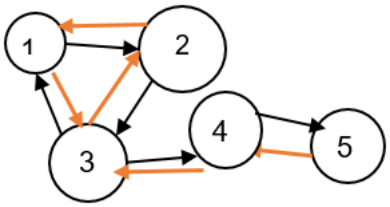
כדי למצוא את הרכיב הקשיר היטב שמכיל את קודקוד v, אנו צריכים לבנות קבוצת A של קדקודים הנגישים מקדקוד v. אחר כך נבנה גרף חדש **G′** - נהפוך את הכיוונים של כל צלעות גרף G – ונבנה קבוצת B של קדקודים הנגישים מקדקוד v בגרף **G′**.

קבוצת $A \cap B$ היא קבוצת הקדקודים המהווים את רכיב קשיר היטב של גרף G המכיל את קדקוד v. סיבוכיות של אלגוריתם במקרה הגרוע היא O(|E|*|V|) כוון שלכל קדקוד עוברים על כל צלעות הגרף.

דוגמה: א) ניקח v = 1, A={2,3,4,5}, B={3,2,} , $A \cap B$={2,3},
component(1)= {1,2,3}

ב) ניקח v = 4, A={5}, B={3,2,1} , $A \cap B$={∅}

component(4)= {4}

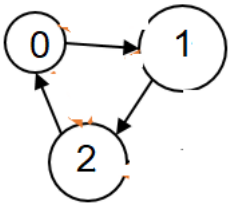


אלגוריתם Tarjan למציאת רכיבים קשירים היטב:

```
for (int u = 0; u < n; u++)
      if (!visited[u])
            dfs(u);
return components;


for (int v : graph[u]) {
      if (!visited[v])
            dfs(v);
      if (lowlink[u] > lowlink[v]) {
            lowlink[u] = lowlink[v];
            isComponentRoot = false;
      }
}
if (isComponentRoot) {
      List<Integer> component = new ArrayList<>();
      while (true) {
            int x = stack.pop();
            component.add(x);
            lowlink[x] = Integer.MAX_VALUE;
            if (x == u)
                  break;
      }
      components.add(component);
}
```

- כמו DFS עם כמה שינויים.
- מוסיפים קודקוד חדש וממנו צלע לכל קודקוד בגרף, ועוברים באמצעות DFS על הגרף. כך מובטח שנעבור על כל הקודקודים.
- הרכיבים הקשירים הם תתי-עצים ב-T, כאשר הוא עץ הנוצר ע"י DFS.
- על מנת לגלות אותם צריך לחתוך קשתות מסוימות ומה שנשאר הם הרכיבים .
- נקבע ששורש של תת-עץ הוא הקדקוד העליון ביותר (צריך לקבוע אם V מסוים הוא שורש כזה).

דוגמה1



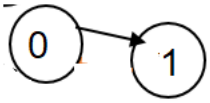

```
u = 0
lowlink: [0, 0, 0]
visited: [true, false, false]
stack: [0]
uIsComponentRoot = true
```

```
u = 1
lowlink: [0, 1, 0]
visited: [true, true, false]
stack: [0, 1]
uIsComponentRoot = true

u = 2
lowlink: [0, 1, 2]
visited: [true, true, true]
stack: [0, 1, 2]
uIsComponentRoot = true
```

יציאה מרקורסיה:

```
u = 2, v = 0
uIsComponentRoot = false

u = 1, v = 2
uIsComponentRoot = false
[[2, 1, 0]]
```

דוגמה 2




```
u = 0
lowlink: [0, 0]
visited: [true, false]
stack: [0]
uIsComponentRoot = true

u = 1
lowlink: [0, 1]
visited: [true, true]
stack: [0, 1]
uIsComponentRoot = true
components: [[1]]

components: [[1], [0]]
[[1], [0]]
```