# Przeszukiwanie grafów

Pseudokod według podręcznika [1], trochę uproszczony.

Oznaczenia:

`Adj(u)` to zbiór wierzchołków, do których jest krawędź z wierzchołka `u`

`u.kolor` to kolor wierzchołka: `BIAŁY` lub `SZARY`

`u.pi` to taki wierzchołek `v`, że dotarliśmy do `u` krawędzią `(v,u)`. Takie krawędzie tworzą drzewo (lub las drzew) spinające przeszukiwany graf. Korzeń (korzenie) tego drzewa (lasu) to te wierzchołki `u` dla których `u.pi == NIL`

### Przeszukiwanie w głąb

```
DFS(G)
// G - graf
  dla każdego wierzchołka u
    u.kolor=BIAŁY
    u.pi=NIL
  dla każdego wierzchołka u
     if u.kolor==BIAŁY
       DFS(G,u)

DFS(G,u)
// G - graf
// u - wierzchołek
// przeszukujemy graf w głąb rozpoczynając od wierzchołka startowego u
// wierzchołki już odwiedzone mają kolor SZARY, u jest BIAŁY
   u.kolor=SZARY
   dla każdego v należącego do Adj(u)
     if v.kolor==BIAŁY
        v.pi=u
        DFS(G,v)
```

### Przeszukiwanie wszerz

```
BFS(G)
// G - graf
  dla każdego wierzchołka s
    s.kolor=BIAŁY
    s.pi=NIL
  dla każdego wierzchołka s
     if s.kolor==BIAŁY
       BFS(G,s)
BFS(G,s)
// G - graf; s - wierzchołek
// przeszukujemy graf wszerz rozpoczynając od wierzchołka startowego s
// wierzchołki już odwiedzone mają kolor SZARY, s jest BIAŁY
   s.kolor=SZARY
   enqueue(Q,s)
   while notEmpty(Q)
     u=dequeue(Q)
     dla każdego v należącego do Adj(u)
       if v.kolor==BIAŁY
          v.kolor=SZARY
          v.pi=u
          enqueue(Q,v)
```

**Silnie spójne składowe grafu skierowanego**

Rozszerzamy najpierw funkcję `DFS(G,u)` o liczenie kolejności *odwiedzania* i *zakończenia przetwarzania* wierzchołków grafu.

```
DFS(G)
// G - graf
  dla każdego wierzchołka u
    u.kolor=BIAŁY
    u.pi=NIL
  dla każdego wierzchołka u   // pętla główna
     if u.kolor==BIAŁY
       DFS(G,u)

DFS(G,u)
// G - graf
// u - wierzchołek
// przeszukujemy graf w głąb rozpoczynając od wierzchołka startowego u
// wierzchołki już odwiedzone mają kolor SZARY, u jest BIAŁY
   time = time+1   // zwiększenie licznika "czasu"
   u.d = time      // zapamiętujemy czas odwiedzenia wierzchołka u
   u.kolor=SZARY
   dla każdego v należącego do Adj(u)
     if v.kolor==BIAŁY
        v.pi=u
        DFS(G,v)
    time = time+1   // zwiększenie licznika "czasu"
    u.f = time      // zapamiętujemy czas przetworzenia wierzchołka u
```

Teraz algorytm znajdowania silnie spójnych składowych grafu skierowanego.

```
Strongly-Connected-Components(G)
   - wykonaj DFS(G) // wylicza u.f dla każdego wierzchołka
   - skonstruuj graf GT będący transpozycją grafu G
   - wykonaj DFS(GT) przebiegając pętlę główną w kolejności malejących u.f
                     (u.f wyliczone w DFS(G) powżej)
   - uzyskane drzewa przeszukiwania w głąb to silnie spójne składowe
     grafu G; wypisz wierzchołki tych drzew
```

**Stwierdzenie.** Złożoność pesymistyczna algorytmu wyznaczającego silnie spójne składowe w grafie skierowanym $G = (V, E)$ jest $\Theta(|V| + |E|)$.

**Stwierdzenie.** Algorytm `Strongly-Connected-Components(G)` poprawnie wyznacza silnie spójne składowe w grafie skierowanym.

# Literatura

[1] Cormen, Leiserson, Rivest, Wprowadzenie do algorytmów, WNT 1998.