Algorytmy do zestawu nr 2[1]

1. Algorytm sprawdzający, czy sekwencja liczb jest ciągiem graficznym

Dane wejściowe: sekwencja n liczb naturalnych: $A = \{A[0], A[1], A[2], \ldots, A[n-1]\}$

Algorytm 1: `degree_seq`(A, n)

```
Posortuj tablicę A nierosnąco
while TRUE do
    if ∀_{i=0,1,...,n−1} A[i] = 0 then              ▷ Jeśli tablica składa się z samych zer,
        return TRUE                                  ▷ to ciąg wejściowy był graficzny.
    end if
    if A[0] ⩾ n  OR ∃_{i=0,...,n−1} A[i] < 0 then
        return FALSE
    end if
    for (i ← 1; i ⩽ A[0]; i ← i + 1) do
        A[i] ← A[i] − 1
    end for
    A[0] ← 0
    Posortuj tablicę A nierosnąco
end while
```

Dodatkowo: jeśli w wejściowej sekwencji liczba wszystkich nieparzystych elementów jest nieparzysta, to ciąg na pewno nie jest graficzny, ponieważ **liczba wierzchołków o nieparzystym stopniu w grafie prostym jest parzysta.**

2. Algorytm oznaczający spójne składowe

Wynikiem uruchomienia procedury `components`(G) jest tablica *comp*, gdzie *comp*$[v]$ oznacza numer spójnej składowej, do której należy wierzchołek v. W trakcie działania algorytmu *comp*$[v] = -1$ oznacza, że wierzchołek v jeszcze nie został odwiedzony.

Procedura rekurencyjna `components_R`$(nr, v, G, comp)$ ma na celu przeszukanie w głąb, zaczynając od danego wierzchołka v. df

[1] Na podstawie: Cormen Thomas H., Leiserson Charles E., Rivest Ronald L., *Wprowadzenie do algorytmów*, Wyd. 4, Warszawa, Wydawnictwo Naukowo – Techniczne, 2001, ISBN 83-204-2665-0.

Algorytm 2: `components(G)`

```
nr ← 0                                        ▷ nr - numer spójnej składowej.
for każdy wierzchołek v należący do grafu G do
    comp[v] ← −1                              ▷ Wszystkie wierzchołki są nieodwiedzone.
end for
for każdy wierzchołek v należący do grafu G do
    if comp[v] = −1 then
        nr ← nr + 1
        comp[v] ← nr        ▷ Oznaczamy v jako odwiedzony i należący do składowej nr.
        components_R(nr, v, G, comp)    ▷ Dalsze odwiedzanie: przeszukiwanie w głąb.
    end if
end for
return comp   ▷ Tablica numerów: wierzchołek v należy do składowej numer comp[v].
```

Algorytm 3: `components_R(nr, v, G, comp)`

```
for każdy wierzchołek u ∈ G będący sąsiadem v do
    if comp[u] = −1 then
        comp[u] ← nr
        components_R(nr, u, G, comp)
    end if
end for
```

3. Cykl Eulera[2]

Cykl Eulera to zamknięta ścieżka zawierająca każdą krawędź grafu dokładnie jeden raz. Graf spójny jest eulerowski (czyli posiada cykl Eulera) wtedy i tylko wtedy, gdy stopień każdego z jego wierzchołków jest liczbą parzystą. **Mostem** nazywamy taką krawędź, której usunięcie spowoduje utratę spójności.

Twierdzenie 1. (Poszukiwanie cyklu Eulera: algorytm Fleury'ego) *Niech G będzie grafem eulerowskim. Wtedy następująca konstrukcja jest wykonalna i daje w wyniku cykl Eulera w grafie G.*

Zacznij cykl w dowolnym wierzchołku u i przechodź krawędzie w dowolnej kolejności, dbając jedynie o zachowanie następujących zasad:

1. *usuwaj z grafu przechodzone krawędzie i wierzchołki izolowane powstające w wyniku usuwania tych krawędzi;*
2. *w każdym momencie przechodź przez most tylko wtedy, gdy nie masz innej możliwości.*

4. Cykl Hamiltona

Cykl Hamiltona to zamknięta ścieżka zawierająca każdy wierzchołek grafu dokładnie jeden raz (za wyjątkiem pierwszego wierzchołka – on pojawia się dwukrotnie). Warunek konieczny: graf musi być

[2] Źródło twierdzenia 1.: Robin J. Wilson, *Wprowadzenie do teorii grafów*, Wyd. drugie, Warszawa, Wydawnictwo Naukowe PWN, 1998, ISBN 83-01-12641-8.

spójny. Warunek konieczny i wystarczający dla dowolnego grafu wejściowego nie istnieje[3] (a przynajmniej nie został jeszcze odkryty). Poszukiwanie cyklu Hamiltona polega na metodzie typu *brute force*: przeszukujemy graf w głąb, odwiedzając wierzchołki dodajemy je na stos. Startujemy od dowolnego wierzchołka.

1. Jeśli stos zawiera wszystkie wierzchołki, to tworzy on ścieżkę Hamiltona: należy jeszcze sprawdzić, czy w grafie jest połączenie między wierzchołkiem startowym a ostatnim na stosie:
   (a) Jeśli tak, to ścieżka jest cyklem Hamiltona, czyli graf jest hamiltonowski.
   (b) Jeśli nie, to usuwamy ostatni wierzchołek ze stosu, oznaczamy go jako nieodwiedzonego i wycofujemy się na wyższy poziom rekurencji.
2. Jeśli stos nie zawiera wszystkich wierzchołków, to rekurencyjnie wywołujemy procedurę dla nieodwiedzonych sąsiadów ostatniego ze stosu. Jeśli po zakończeniu pętli stos nadal nie zawiera wszystkich wierzchołków grafu, to usuwamy ostatni wierzchołek ze stosu, oznaczamy go jako nieodwiedzonego i wycofujemy się na wyższy poziom rekurencji.

Jeżeli powyższa metoda siłowa nie zwróci cyklu Hamiltona, to znaczy, że graf nie jest hamiltonowski.

[3]Dla zainteresowanych – niektóre warunki wystarczające, by graf był hamiltonowski: https://www.math.uni-hamburg.de/home/diestel/books/graph.theory/preview/Ch10.pdf. *Uwaga: proszę pamiętać, że – skoro są to warunki wystarczające – jeśli graf ich nie spełnia, to nadal nie wiadomo, czy jest hamiltonowski, czy nie.*