Algorytmy grafowe programik 02: Spójność.

A Program do napisania

Proszę o przesłanie w odpowiednim zadaniu w MSTeams

- plików o zindywidualizowanej nazwie **02NazwiskoImie.py** albo **02NazwiskoImieNieDziala.py** (jeśli podjęli Państwo próbę zrobienia, ale nie działa);

- **Proszę:**
  - nazwisko pierwsze, bez polskich znaków;
  - nie wysyłać niekompletnych programów bez dopisku NieDziala.
- Proszę o wpisanie w programie $'leifman.txt'$ a nie odwołania do pliku, które Państwo wykorzystywali.
- Proszę nie wysyłać mi pliku tekstowego z grafem.

Zadanie A.1. W pliku leifman.txt zapisana jest macierz następników digrafu. Dla Państwa wygody w tej macierzy znaki „0" zostały zastąpione znakiem „-", żeby wygodnie skorzystać bez zbytnich przeróbek z zadania z poprzednich zajęć. We wczytanym grafie wierzchołki powinny być numerowane kolejno liczbami naturalnymi zaczynając od 1 (zgodnie z kolejnością wierszy).
Napisz program, który wykorzystując algorytm Leifmana znajduje:

- PLAN MINIMUM (za 2/3 punktów): zbiory V_{ij} **wykorzystując algorytm BFS** zaczynając od wierzchołka o numerze 1 (ignorujemy wyszukiwanie wierzchołków o stopniu wejścia/wyjścia równym 0).
- PLAN MAKSIMUM (za wszystkie punkty): wszystkie składowe silnej spójności danego grafu, etykiety nadajemy też **wykorzystując algorytm BFS**.

W wyjściu powinno się znaleźć:

- PLAN MINIMUM: wypisane zbiory V_{ij} wyznaczone w pierwszej iteracji.
- PLAN MAKSIMUM wszystkie stany L, C (po każdej zmianie) oraz wszystkie kolejno utworzone zbiory V_{ij} (w momencie ich wyznaczenia). Uwaga: tutaj nie ignorujemy żadnego etapu algorytmu.

WSKAZÓWKI (**Tylko dla tych, którzy sami nie umieją sobie poradzić.** Nie jest to „jedyny właściwy" sposób podejścia do tego zadania, ale niektórym pomoże):

a. Można na przykład zrobić dwa słowniki list (jak poprzednio słownik z listami następników) jeden dla list następników jak w poprzednim zadaniu a drugi dla list poprzedników (WYKORZYSTAJ POPRZEDNI PROGRAM) + na każdym z nich niezależnie zadziałać algorytmem BFS;

b. W BFS zamiast NUMBER tutaj zastosować etykiety p() i l() (nie będzie potrzebny licznik „i" - po prostu odkryte wierzchołki dostaną etykietę równą 1, gdy zostaną odkryte + początkowy wierzchołek ma też etykietę 0 na początku + ignorujemy też dodawanie krawędzi do TREE i OTHER - interesują nas tylko etykiety)

c. Etykiety p() i l() można na przykład zapisać w dwóch oddzielnych słownikach lub w jednym (klucze - wierzchołki, wartości: listy dwuelementowe z etykietami lub jak Państwo wolą, słowniki z dwoma kluczami l i p).

d. Nie trzeba się spinać na plan maksimum. 2/3 punktów to i tak dużo i jeszcze będą prostsze programy.

PRZYKŁADOWE WEJŚCIE:
- 1 - - - - 1 - - - - - - - - - - - -
- - 1 - - - - - - - - - - - - - - - -
- 1 - 1 - - - - - - - - - - - - - - -
- - - - 1 - - - - - - - - - - - - - -
- - - 1 - - - - - - - - - - - - - - -
1 - - - - - 1 - - - - - - - - - - - -
- 1 - - - - - 1 - - - - - - - - - - -
- - 1 1 - - - - 1 - - - 1 - - - - - - -
- - - 1 - - - - - 1 - - - - - - - - - -
- - - - 1 - - - - - - - - - 1 - - - - -
- - - - - 1 - - - - - - - - - - - - -

- - - - - 1 1 - - - 1 - 1 - - 1 - - - -
- - - - - - 1 - - - - - - - - - - - - -
- - - - - - - - 1 - - - 1 - 1 - - - 1 1
- - - - - - - - 1 - - - - - - - - - - -
- - - - - - - - - - 1 - - - - - 1 - - -
- - - - - - - - - - - 1 1 - - - - 1 - -
- - - - - - - - - - - - 1 1 - - - - - -
- - - - - - - - - - - - - - - - - - 1 - -
- - - - - - - - - - - - - - 1 - - - 1 -

PLAN MINIMUM PRZYKŁADOWE WYJŚCIE:
V11= empty
V10=[2, 3, 4, 5, 7, 8, 9, 10, 13, 15]
V01=[6, 11, 12, 16, 17]
V00=[1, 14, 18, 19, 20]

PLAN MAKSIMUM PRZYKŁADOWE WYJŚCIE (Oczywiście kolejność rozpatrywania zbiorów z L i struktury zapisanych danych nie muszą być identyczne):

Rozpatrujemy: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
V11= empty
V10=[2, 3, 4, 5, 7, 8, 9, 10, 13, 15]
V01=[6, 11, 12, 16, 17]
V00=[1, 14, 18, 19, 20]
C= [[1]]
L= [[2, 3, 4, 5, 7, 8, 9, 10, 13, 15], [6, 11, 12, 16, 17], [14, 18, 19, 20]]
Rozpatrujemy: [2, 3, 4, 5, 7, 8, 9, 10, 13, 15]
V11=[2, 3]
V10=[4, 5]
V01=[7, 8, 13]
V00=[9, 10, 15]
C= [[1], [2, 3]]
L= [[6, 11, 12, 16, 17], [14, 18, 19, 20], [4, 5], [7, 8, 13], [9, 10, 15]]
Rozpatrujemy: [6, 11, 12, 16, 17]
Wierzcholek o stopniu wyjscia 0
C= [[1], [2, 3], [6]]
Wierzcholek o stopniu wyjscia 0
C= [[1], [2, 3], [6], [11]]
V11=[12, 16, 17]
V10= empty
V01= empty
V00= empty
C= [[1], [2, 3], [6], [11], [12, 16, 17]]
L= [[14, 18, 19, 20], [4, 5], [7, 8, 13], [9, 10, 15]]
Rozpatrujemy: [14, 18, 19, 20]
V11=[14, 18, 19, 20]
V10= empty
V01= empty
V00= empty
C= [[1], [2, 3], [6], [11], [12, 16, 17], [14, 18, 19, 20]]
L= [[4, 5], [7, 8, 13], [9, 10, 15]]
Rozpatrujemy: [4, 5]
V11=[4, 5]
V10= empty
V01= empty
V00= empty
C= [[1], [2, 3], [6], [11], [12, 16, 17], [14, 18, 19, 20], [4, 5]]
L= [[7, 8, 13], [9, 10, 15]]
Rozpatrujemy: [7, 8, 13]
V11=[7, 8, 13]
V10= empty

V01= empty
V00= empty
C= [[1], [2, 3], [6], [11], [12, 16, 17], [14, 18, 19, 20], [4, 5], [7, 8, 13]]
L= [[9, 10, 15]]
Rozpatrujemy: [9, 10, 15]
V11=[9, 10, 15]
V10= empty
V01= empty
V00= empty
C= [[1], [2, 3], [6], [11], [12, 16, 17], [14, 18, 19, 20], [4, 5], [7, 8, 13], [9, 10, 15]]
L= []