

## Praca domowa 06 – graph

Termin zwrotu : 2 grudnia godz. 23.00

Zadanie uznaje się za zaliczone, gdy praca oceniona zostanie na co najmniej 6 pkt.

Niech  $G=(V,E)$  będzie grafem nieorientowanym, w którym  $V$  reprezentuje zbiór wierzchołków etykietowanych kolejnymi liczbami  $1,2,\dots,n$  gdzie  $n=|V|$  oraz  $E$  definiuje zbiór  $m=|E|$  krawędzi grafu reprezentowanych przez pary  $(u,v)$  gdzie  $u \in \{1,2,\dots,n\}$  oraz  $v \in \{1,2,\dots,n\}$ .

W oparciu o technologię EJB należy zbudować komponent *Graph* organizujący niezbędne do przechowywania opisu grafu struktury danych oraz implementujący metody potrzebne do tworzenia oraz przeszukiwania grafu. Bezpośrednie odwołania do elementów struktury są niedozwolone. Dostęp winien odbywać się wyłącznie z użyciem odpowiednio zaimplementowanych metod ( np. przykładowo *addEdge(...)*, *getEdge(...)*, *getNextEdge(...)* itp. ). Zaproponowane metody winny być udostępnione z wykorzystaniem interfejsu *IGraphRemote*.

Należy zbudować komponent *Search* implementujący algorytm obliczania ilości spójnych składowych w grafie. Metoda służąca obliczaniu składowych winna być udostępniona z wykorzystaniem interfejsu *ISearchRemote*. Ewentualne wszelkie inne operacje związane z operowaniem grafem realizowane być mogą wyłącznie w oparciu o metody udostępnione interfejsem *IGraphRemote*.

Należy napisać program klienta *AppClient* umożliwiający wczytanie zbioru krawędzi z tekstowego pliku wejściowego (wskazywanego przez parametr `<input_file>` programu) zawierającego w kolejnych liniach ciąg liczb, z których każde dwie kolejne traktowane jako para  $u$  i  $v$  opisują krawędzie grafu. Ilość krawędzi grafu określona jest wyłącznie poprzez ilość zapisów w pliku wejściowym. Długość pojedynczej linii ani ilość liczb w linii nie jest w jakikolwiek sposób określona (format swobodny). Aplikacja – korzystając z operacji udostępnionych interfejsem *IGraphRemote* - winna utworzyć właściwą strukturę danych opisujących graf, a następnie korzystając z udostępnionej interfejsem *ISearchRemote* metody wyznaczyć ilość spójnych składowych grafu.

Rozwiązanie nie może korzystać z bibliotek zewnętrznych. Wynik końcowy (w strumieniu wyjściowym nie powinny pojawiać się jakiegokolwiek inne elementy – np. wydruki kontrolne) działania programu musi zawierać wyłącznie pojedynczą wartość wskazującą ilość składowych badanego grafu.

Ilość składowych : 2

Program ma być zapisany w plikach : *IGraphRemote.java* zawierającym definicję interfejsu komponentu zdefiniowanego w pliku *Graph.java* , *ISearchRemote.java* zawierającym definicję interfejsu komponentu zdefiniowanego w pliku *Search.java* , oraz kod programu *AppClient.java*. Poszczególne elementy rozwiązania nie mogą korzystać z bibliotek zewnętrznych innych niż niezbędne moduły serwera (jak np. *gf-client.jar*, *javaee.jar* itp.).

Proces kompilacji musi być możliwy z użyciem komendy

```
javac -cp <app-server-modules> -Xlint AppClient.java Graph.java IGraphRemote.java \
Search.java ISearchRemote.java
```

Rozwiązanie testowane będzie w środowisku serwera aplikacyjnego GlassFish 4. Podczas weryfikacji program wykonywany będzie kolejno dla kilku różnych zestawów danych testowych.

#### Wymagania :

- Komponent umożliwiający tworzenie i przeszukiwanie grafu winien być umieszczony w pliku `Graph.java`.
- Interfejs udostępniający zaprojektowane metody realizowane przez komponent `Graph.java` winien zostać zdefiniowany w pliku `IGraphRemote.java`.
- Algorytm umożliwiający wyznaczanie ilości składowych grafu winien być umieszczony w pliku `Search.java`.
- Interfejs udostępniający zaprojektowane metody realizowane przez komponent `Search.java` winien zostać zdefiniowany w pliku `ISearchRemote.java`.
- Aplikacja klienta wprowadzająca dane oraz organizująca proces obliczeń winna być umieszczona w pliku `AppClient.java`
- W pliku `README.pdf` winien być zawarty szczegółowy opis zastosowanych do przechowywania grafu struktur danych, proponowanych metod dostępu/przeszukiwania oraz algorytmu wyznaczania składowych spójnych.
- Proces obliczenia rozwiązania winien się kończyć w czasie nie przekraczającym 1 min (orientacyjnie dla typowego notebooka). Po przekroczeniu limitu czasu zadanie będzie przerywane, i traktowane podobnie jak w sytuacji błędów wykonania (czyli nie podlega dalszej ocenie).

#### Sposób oceny :

- 1 pkt – **Weryfikacja** : czy program jest skompletowany i spakowany zgodnie z ogólnymi zasadami przesyłania zadań.
- 1 pkt – **Kompilacja** : każdy z plików winien być kompilowany bez jakichkolwiek błędów lub ostrzeżeń (w sposób omówiony wyżej)
- 1 pkt – **Wykonanie** : program powinien wykonywać się bez jakichkolwiek błędów i ostrzeżeń (dla pliku danych wejściowych zgodnych z wyżej zamieszczoną specyfikacją) z wykorzystaniem omówionych wyżej parametrów linii komend.
- 2 pkt – **README** : plik `README.pdf` dokumentuje w sposób kompletny i właściwy algorytm poszukiwania rozwiązania.
- 1 pkt – **Styl kodowania** : czy funkcje i zmienne posiadają samo-wyjaśniające nazwy ? Czy podział na funkcje ułatwia czytelność i zrozumiałość kodu ? Czy funkcje eliminują (redukują) powtarzające się bloki kodu ? Czy wcięcia, odstępy, wykorzystanie nawiasów itp. (formatowanie kodu) są spójne i sensowne ?
- 4 pkt – **Poprawność algorytmu** : czy algorytm został zaimplementowany poprawnie a wynik odpowiada prawidłowej (określonej zbiorem danych testowej) wartości.