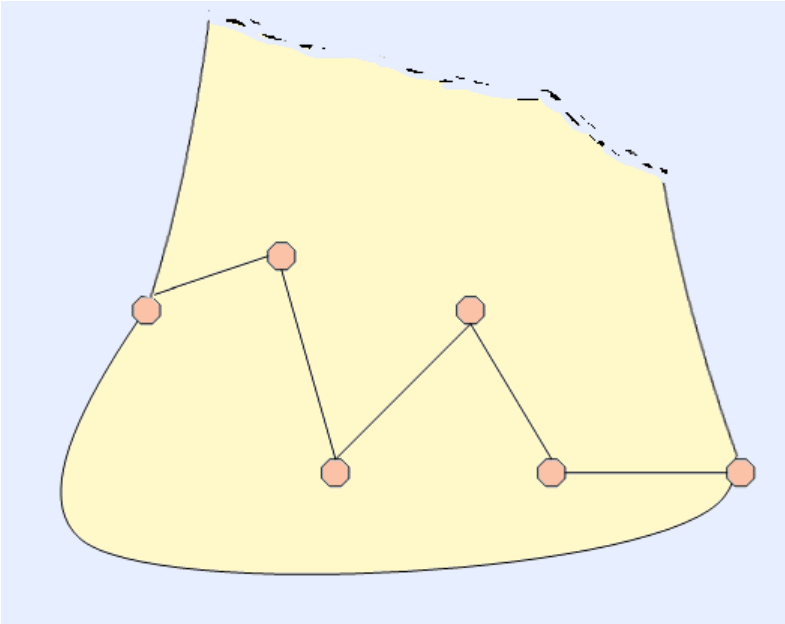
# Praca domowa 03 – highway

Termin zwrotu : 5 listopada godz 23.00
Zadanie uznaje się za zaliczone, gdy praca oceniona zostanie na co najmniej 6 pkt.

Dwa miasta leżące odpowiednio na zachodnim oraz wschodnim brzegu półwyspu połączone zostały drogą przechodzącą przez wszystkie *n* miast leżących na półwyspie w taki sposób, iż droga przechodzi przez każde z nich, wyznaczając równocześnie ich numerację (czyli miasto nr *1* na wybrzeży zachodnim, miasto oznaczone jako *n* na wschodnim).



Planuje się budowę k autostrad łączących z góry wskazane pary miast ze sobą. Nowo budowane autostrady nie mogą się wzajemnie przecinać, ani też przecinać istniejącej już drogi. Mogą być zatem prowadzone zależnie od potrzeb północną lub południową stroną istniejącej drogi.

Dane wejściowe umieszczone są w SQL'owej bazie danych. Połączenie do SQL-owej bazy danych (dostęp do bazy) realizowany jest z wykorzystaniem driverów JDBC poprzez wykonanie metody

```
String database = <conn>;  // parametr z URL'a
Connection conn = DriverManager.getConnection(database);
```

Z bazy danych należy wczytać pewną ilość kolejnych par liczb *i, j* gdzie *i, j* oznaczają numery miast, które winny zostać połączone autostradą. Miasto o numerze 1 położone jest na zachodnim wybrzeżu, miasto o największym występującym w danych numerze *n* – na wybrzeżu wschodnim. Wykaz planowanych do budowy autostrad przechowywany jest w SQL-owym repozytorium danych (w bazie danych) w tabeli o nazwie HData. Struktura tabeli utworzona została z wykorzystaniem instrukcji

```
CREATE TABLE [dbo].[HData](
      [id] [int] NOT NULL,
      [i] [int] NOT NULL,
      [j] [int] NOT NULL,
CONSTRAINT [PK_Data] PRIMARY KEY CLUSTERED
      ([id] ASC) ON [PRIMARY] )
```

Wynikiem działania programu jest wartość 1 – jeżeli planowana sieć połączeń jest możliwa do realizacji, 0 w przeciwnym przypadku.

Program ma być zrealizowany w postaci servletu `Highway.java` – zawierającego implementację algorytmu. Rozwiązanie nie może korzystać z jakichkolwiek bibliotek zewnętrznych oraz nie może być zależne od jakiegokolwiek dialektu SQL.

Proces kompilacji musi być możliwy z użyciem komendy

```
javac –Xlint Highway.java
```

Niezbędny do nawiązania połączenia z bazą danych parametr `<conn>` przekazywany jest w URL'u uruchamiającym servlet.

Zawartość pliku web.xml, który używany będzie w trakcie uruchamiania i testowania komponentu podano niżej :

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://java.sun.com/xml/ns/javaee"
       xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
       xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
       id="WebApp_ID" version="3.0">
  <servlet>
    <servlet-name>servletNNNNN</servlet-name>
    <servlet-class>Highway</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>servletNNNNN</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Przykładowy wynik końcowy :

```
Wynik : 1
```

Wymagania :

- Klasa implementująca algorytm winna być zdefiniowana w pliku `Highway.java`
- W pliku README.pdf winien być zawarty opis organizacji struktur danych, szczegółowy opis algorytmu oraz analiza/dyskusja złożoności obliczeniowej zaproponowanego rozwiązania.
- Proces poszukiwania rozwiązania dla zestawu danych o ilości rzędu n=100 winien się kończyć w czasie nie przekraczającym 3 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 struktury danych, oraz opis przyjętej koncepcji algorytmu
- 1 pkt – **Styl kodowania** : czy funkcji 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.