Praca domowa 07 – classification

Termin zwrotu : 15 grudnia godz. 23.00
Zadanie uznaje się za zaliczone, gdy praca oceniona zostanie na co najmniej 6 pkt.

W pliku wejściowym zapisano pewną ilość n+1 elementowych wektorów $v(x_1,v_2,....v_{n+1},)$. Składowe wektorów v_i przyjmują wartości rzeczywiste. Wartość n określona jest parametrem zewnętrznym programu `<n-size>`. Plik wejściowy (wskazywany przez parametr `<input_file>` programu) zawiera w kolejnych liniach ciąg liczb, z których każde n+1 kolejnych traktowane jako współrzędne pewnego wektora. Ilość wektorów 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).

Zadanie polega na zaproponowaniu mechanizmu klasyfikacji, przy czym algorytm w pierwszej fazie (w pierwszym etapie) będziemy uczyć. Proces nauczania polega na podaniu pewnej liczby wektorów, o których dokładnie wiemy, do której z klas każdy z nich należy. Ilości klas nie znamy (określona ona jest przez dane wejściowe). W procesie nauczania element $1 <= v_1 <= k$ (v_1 jest zawsze nieujemną liczbą całkowitą) że oznacza numer klasy do której należy n-wymiarowy wektor $(v_2,....v_{n+1})$. A więc np. jeżeli parametr `<n-size>` określono jako 3 a w pliku `<input_file>` napotkamy ciąg liczb

2 5 -1 3 4 0 2 -1 2 ……..

to pierwsza czwórka liczb (2, 5, -1, 3) interpretowana jest następująco : liczba 2 wskazuje, iż występujący po niej trójwymiarowy wektor (5, -1, 3) traktować należy jako punkt w przestrzeni trójwymiarowej należący do klasy 2. Kolejna czwórka liczb (4, 0,2,-1) wskazuje, że punkt (0, 2, -1) należy do klasy 4 itd. Podając pewną – dość dużą ilość elementów ciągu uczącego dla każdej z klas, uzyskujemy w procesie uczenia wiedzę o cechach charakterystycznych każdej z klas.

W pewnym momencie w pliku wejściowym pojawi się pierwszy z wektorów, którego współrzędna $v_0 = 0$. Oznacza to, że proces uczenia należy uznać za zakończony, i przechodzimy do fazy rozpoznawania. A więc np.

…..0 3 -1 3 …..

Oznacza, że musimy rozpoznać (sklasyfikować) do której z k klas wcześniej poznanych wektor (3, -1, 3) najbardziej pasuje.

Program ma być zapisany wyłącznie w dwóch plikach `Classification.java` zawierającym implementację zadania, oraz `Main.java` – zawierającym programem główny. Program nie może korzystać z jakichkolwiek bibliotek zewnętrznych.

Proces kompilacji musi być możliwy z użyciem komendy

```
javac –Xlint Classification.java Main.java
```

Uruchomienie programu winno być możliwe z użyciem komendy

```
java Main <input_file> <n-size>
```

Wynik końcowy (w strumieniu wyjściowym nie powinny pojawiać się jakiekolwiek inne elementy – np. wydruki kontrolne) działania programu musi odpowiadać ściśle zdefiniowanym warunkom: Jeżeli w zbiorze wejściowym pojawiło się m wektorów wymagających sklasyfikowania to wynik jest pojedynczą (!!!) m-cyfrową liczbą, której każda z cyfr określa kolejno klasy przynależności dla każdego z klasyfikowanych wektorów. Jeżeli sklasyfikować należało cztery wektory, które wg programu zaliczone zostały odpowiednio do klas 5, 6, 1, 3 to wynik winien mieć postać jak niżej

```
Wyniki klasyfikacji : 5613
```

Wymagania :

- Klasa implementująca problem winna zostać zdefiniowana w pliku `Classification.java`
- Klasa implementująca mechanizm program główny (metoda main) winny być zdefiniowane w pliku `Main.java`
- W pliku README.pdf winien być zawarty szczegółowy opis organizacji struktur danych oraz szczegółowy opis zastosowanego mechanizmu uczenia i klasyfikacji.
- Proces poszukiwania 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 – **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 minimalizacji strat (odpadów).
- 1 pkt – **Komentarze wewnętrzne** : czy program jest skomentowany w sposób zapewniający zrozumienie jego działania, oraz wyjaśniający warunki, które muszą zachodzić przed i po wykonaniu każdej z funkcji.
- 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 , przy czym za mechanizm uczenia się i klasyfikacji (winny to być dwie oddzielne metody) można otrzymać dwa punkty, a za wybór, poprawność implementacji i skuteczność wykorzystanej metody klasyfikacji kolejne dwa..