Praca domowa 1 – levenshtein

Termin zwrotu : 18 października godz. 23.00
Zadanie uznaje się za zaliczone, gdy praca oceniona zostanie na co najmniej 6 pkt.

W pliku tekstowym zawarta jest bliżej nieokreślona ilość zapisów (linii). Zawartość każdej z linii pliku zgodna jest z wzorcem opisanym wyrażeniem regularnym postaci :

```
^[\s]*([a-zA-Z]+[\s]+){1,2}[a-zA-Z]+[\s]*$
```

Każda z linii traktowana być może jako kolejny wpis w spisie osobowym. Określona wyżej składnia dopuszcza zatem w oczywisty sposób zarówno wpisy zawierające nazwisko poprzedzające jedno bądź dwa imiona, lub alternatywnie nazwisko poprzedzone jednym lub dwoma imionami.

Dla zadanego łańcucha testowego (zgodnego z opisanym wyżej wzorcem) korzystając z miary Levenshteina (miary podobieństwa dwóch łańcuchów znaków) należy wyznaczyć w zadanym zestawie danych testowych linię zawierającą zapis najbardziej podobny (o najmniejszej wartości odległości określonej w.w. miarą). W przypadku gdy więcej niż jeden zapis jest odległy o tę sama wartość, jako rozwiązanie należy przyjąć pierwszy z nich.

Program ma być zapisany wyłącznie w dwóch plikach : `Levenshtein.java` zawierającym implementację algorytmu obliczającego odległośc dwóch łańcuchów znaków, oraz `AppClient.java` – zawierającym program główny realizujący zasadniczy algorytm. Program nie może korzystać z jakichkolwiek bibliotek zewnętrznych.

Proces kompilacji musi być możliwy z użyciem komendy

```
javac –Xlint Levenshtein.java AppClient.java
```

Uruchomienie programu winno być możliwe z użyciem komendy

```
java AppClient <file> ”<test_string>”
```

gdzie parametr `<file>` określa lokalizację pliku danych a `<test_string>` to analizowany (szukany) łańcuch tekstowy. Wynik końcowy (w strumieniu wyjściowym nie powinny pojawiać się jakiekolwiek inne elementy – np. wydruki kontrolne) działania programu musi zawierać pojedynczą liczbę całkowitą określającą numer linii pliku wejściowego spełniającą warunki niniejszego zadania.

```
Linia : 652
```

© 2017 dr inż. Jerzy R. Jaworowski, Instytut Teleinformatyki, Politechnika Krakowska im. Tadeusza Kościuszki

Wymagania :

- Klasa implementująca problem winna zostać zdefiniowana w pliku `Levenshtein.java`
- Klasa implementująca mechanizm program główny (metoda main) winny być zdefiniowane w pliku `AppClient.java`
- W pliku README.pdf winien być zawarty szczegółowy opis organizacji struktur danych oraz szczegółowy opis zastosowanego mechanizmu (metody) poszukiwania ścieżki o minimalnym koszcie transportu.
- Proces poszukiwania rozwiązania dla zestawu danych o rozmiarze n rzędu 10^4 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 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.

© 2017 dr inż. Jerzy R. Jaworowski, Instytut Teleinformatyki, Politechnika Krakowska im. Tadeusza Kościuszki