

Mini Projekt - Mapa

Rafał Byczek

2 stycznia 2017

Przesyłam Panu rozwiązanie zadania Mapa. Uwaga numer 1 - proszę używać pythona2, bo do tej wersji dostosowałem moje rozwiązanie oraz przed uruchamianiem **python2 main.py** zaleca się używanie aplikacji z konta roota, żeby w paru miejscach nie trzeba podawać hasła podczas wykonywania się linii w kodzie basha z prefiksem sudo :) Na początek mały opis co się tutaj wogóle znajduje:

- **./geopy** - jest to wspólna biblioteka pythonowa pobrana z strony <https://pypi.python.org/pypi/geopy>, dzięki uprzejmości tej biblioteki dostałem możliwość dostawiania współrzędnych geograficznych jakichś krajów, ulic itp, które są używane w poniższych rozwiązaniach
- **./data/data01.txt** - to jest plik udostępniony przez Pana pod nazwą *data-used-autnums* zawierający informacje o numerach AS.
- **./data/data02.txt** - to jest plik udostępniony przez Pana pod nazwą *data-raw-table*, w którym są podane powiązania numerów IP z numerami AS.
- **./data/data03.txt** - to jest plik mapujący pełne nazwy krajów na skróty dwuznakowe.
- **./databse.py** - to jest plik, który należy uruchomić na początku. zadaniem tego programu jest odpowiednie przetworzenie plików z folderu *data*, do formatu, który mi potem jest pomocny. I tak plik *./data/data01.txt* ewoluuje do pliku **./data02.in**, plik *./data/data02.txt* do **./data03.in** oraz plik *./data/data03.txt* do **./data01.in**.
- **./data01.in** trzyma wiersze postaci *skrot_kraju%pelna_nazwa_kraju*
- **./data02.in** trzyma wiersze postaci *numer_as%skrot_kraju%dodatkowe_informacje_o_numerze_as*
- **./data03.in** korzystając z informacji z pliku wiążącego ip z as trzyma wiersze następującej postaci *numer_ip_normalnie%maska_normalnie%numer_ip_binarnie%maska_binarnie*

No i teraz mamy doczynienia z głównym plikiem **./main.py**. W nim się dzieje cała reszta. Najpierw w liniach od 7-29 następuje załadowanie informacji z plików *./data01.in*, *./data02.in*, *./data03.in* do odpowiednich słowników. Funkcja **def check(ip)** znajdująca się w liniach od 31 do 44 ma za zadanie sprawdzać czy w wczytanych danych istnieje nasz numer IP, bo te pliki nie zawierają wszystkiego. W głównej pętli programy prosimy użytkownika o podanie numeru IP. Gdy dostajemy numer IP w liniach od 63 do 91 próbujemy wycisnąć z naszych plików wszystko co można i stosowne informacje są wyświetlane na ekranie użytkownikowi.

Druga część rozwiązania korzysta z linuxowego polecenia **whois**. W liniach od 49 do 57 jest ono używane, a za pomocą grepa do katalogu temp są wyodrębniane do plików interesujące nas informacje na temat danego numeru IP, jego numeru AS i organizacji, która zarządza tym numerem AS. Potem te dane są przetwarzane w liniach od 99 do 160 i wypisywane na ekran użytkownikowi.

Trzecia część zaś zawiera rozwiązanie korzystające z vpn i pingowania. Mamy tutaj program **./getPingTime.sh**, który uruchamiamy w postaci **./getPingTime.sh miasto numer_ip**. Za pomocą programu **./getPingTime.sh** najpierw tworzę vpn za pomocą pliku dostarczonego przez Pana nam zawierającego miasta *bombay, california, frankfurt, saopaulo, sydney, tokyo, virginia*. I tak dla każdego miasta robię w tablicy routingu małe przekierowania przez podane miasto, i potem pingujemy i dla



każdego miasta mierzymy średni czas przesłania pakietów. Potem każde połączenie czyści za sobą i przywraca tablice routingu do pierwotnej postaci. Zaś w pliku `./main.py` w liniach od 197 do 220 następuje przeparsowanie tego co wypisał program ping i policzenie średniego czasu na przesłanie pakietu i wypisanie użytkownikowi stosownej informacji dla każdego miasta.

Używając biblioteki **geopy** i znajdującego się tam pliku `./distance.py`, w którym jest między innymi zaimplementowane liczenie najkrótszej odległości na sferze zwanej **Ortodroma**. Za pomocą testów wykonanych w wykomentowanych liniach w okolicach linii 18-24, 256-259, 280-281, które brały lokalizację mojego domu i liczyły odległości od miast *bombay*, *california*, *frankfurt*, *saopaulo*, *sydney*, *tokyo*, *virginia* i czasy trwania pingów. Na podstawie tak zebranych danych uzyskałem coś na kształt średniej prędkości *internetu w powietrzu*, uzyskując około **20165129.2815 m/s**. Tej prędkości będę używał w dalszych obliczeniach. Teraz znając prędkość przepływu zapisaną w zmiennej **SREDNIA_PREDKOSC** oraz mając czasy pingów z poszczególnych miast, jestem w stanie wyznaczyć plus minus odległość interesującego mnie numeru IP od danego miasta, co też potem ładuje w słowniku **odleglosc**, który to słownik trzyma teraz dla mnie pod kluczem nazwa miasta - przypuszczalna odleglosc od niego wyrażoną w kilometrach.

A więc mam teraz jakiś zbiór punktów na kuli ziemskiej wraz z odległościami od szukanego miejsca. Więc ja teraz traktuję odwrotności tych odległości jako wagi tych punktów i będę szukał środka masy dla tych punktów na kuli ziemskiej. Posłuży się w tym celu algorytmem liczenia środka masy dla punktów na kuli ziemskiej znajdującym się tutaj <http://www.geomidpoint.com/calculation.html>. Algorytm ten został zaimplementowany w pliku `main.py` w liniach od 303 do 350. Z wyliczonego środka masy za pomocą biblioteki **geopy** uzyskuję co to za miejsce na kuli ziemskiej ma takie współrzędne geograficzne jak środek masy moich punktów.