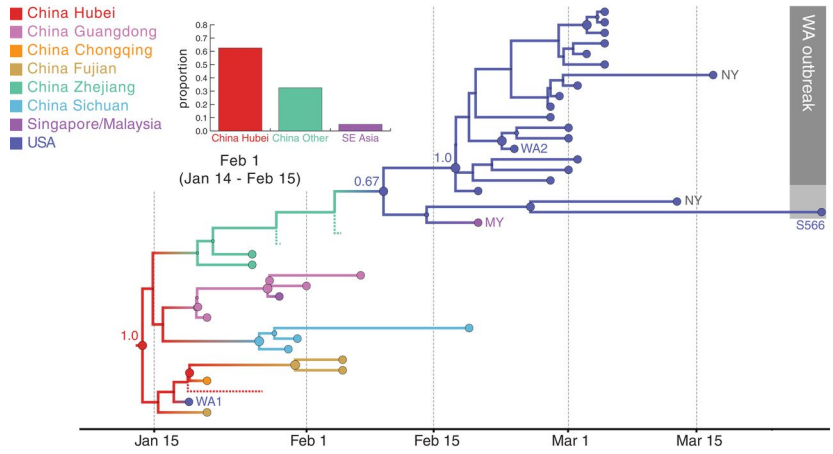
Algorytmy i struktury danych (bioinformatyka, 2021)

Duży projekt Termin: niedziela, 30.05.2021, 23:59

Drzewo filogenetyczne

W tym zadaniu interesują nas *drzewa filogenetyczne* o następującym kształcie: są to drzewa o dowolnym rozgałęzieniu (każdy wierzchołek może mieć dowolnie wiele dzieci), w którym każdy wierzchołek zawiera pewną sekwencję nukleotydów (napis składający się ze znaków `ACTG`) oraz pewne dodatkowe informacje na temat próbki, z których ona pochodzi (identyfikator próbki, kraj oraz data pobrania próbki).

Takie drzewo ma odzwierciedlać zależności ewolucyjne – relacja bycia przodkiem w drzewie miałaby odpowiadać wyewoluowaniu jednego organizmu z drugiego. Interesują nas dość wąskie ramy czasowe (rzędu kilku lat); za przykład może posłużyć wirus SARS-CoV-2 i jego mutacje.



Źródło: *The emergence of SARS-CoV-2 in Europe and North America* (Worobey et al., 2020).

Przyjmujemy założenie, że jeśli wierzchołek x jest rodzicem wierzchołka y w drzewie, to data pobrania próbki odpowiadającej x musi być wcześniejsza niż data pobrania próbki odpowiadającej y. Dla uproszczenia załóżmy, że daty pobrania próbek są różne (żadne dwie próbki nie zostały pobrane tego samego dnia).

Odległość edycyjna

Prostą miarą podobieństwa dwóch sekwencji jest *odległość edycyjna*. Jest to minimalna liczba *insercji*, *delecji* i *substytucji* (operacji dodania dowolnego znaku, usunięcia dowolnego znaku lub zamiany dowolnego znaku na inny) potrzebnych do otrzymania jednej sekwencji z drugiej.

Koszt drzewa filogenetycznego definiujemy następująco: dla każdego wierzchołka (oprócz korzenia) liczymy odległość edycyjną między nim a jego rodzicem i sumujemy wszystkie te wartości.

Dla danej listy próbek i sekwencji, *optymalne* drzewo filogenetyczne to drzewo o najmniejszym koszcie, w którym występują wszystkie sekwencje z listy i data pobrania próbki rodzica jest zawsze wcześniejsza niż dziecka.

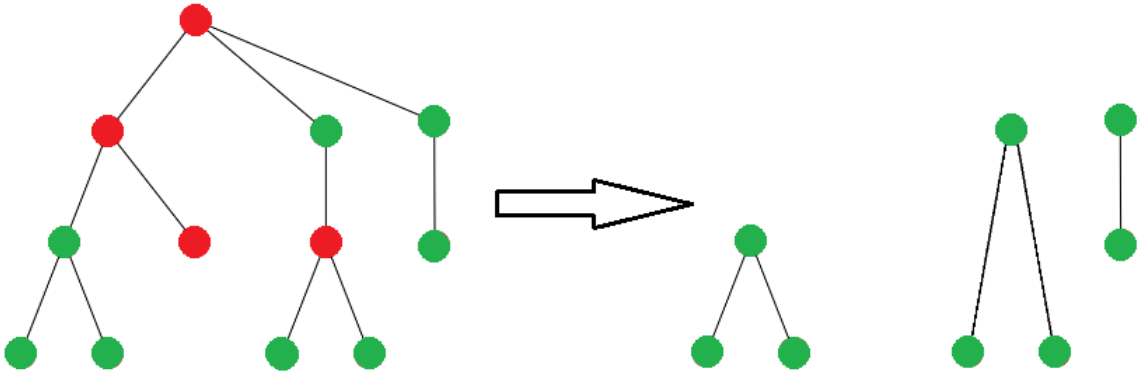
Zadanie

Zadanie polega na napisaniu funkcji umożliwiających następujące główne operacje:

- wczytanie informacji o próbkach i sekwencjach z pliku `.fasta`;
- stworzenie optymalnego drzewa filogenetycznego z podanej listy próbek i sekwencji;
- „przefiltrowanie" drzewa według kraju pobrania próbki (opis niżej);
- szybką konstrukcję drzewa o niskim koszcie (ale niekoniecznie optymalnego; opis niżej).

Filtrowanie

Mając drzewo filogenetyczne ze wszystkimi próbkami, chcielibyśmy móc ograniczyć się tylko do próbek z konkretnego kraju. Definiujemy w tym celu operację filtrowania wierzchołków drzewa ze względu na kraj. Intuicyjnie: tworzymy nową kopię wierzchołków, których próbki zostały zebrane w odpowiednim kraju, i odpowiednio łączymy krawędziami przodków i potomków. Zwykle nie otrzymamy w ten sposób jednego drzewa, tylko wiele drzew. Trochę bardziej formalnie: wynikiem ma być lista drzew; jeśli w oryginalnym drzewie wierzchołek x jest przodkiem wierzchołka y i oba odpowiadają próbkom zebranym w podanym kraju, to na wynikowej liście musi pojawić się drzewo, w którym (kopia) x jest przodkiem (kopii) y.



Na przykładzie powyżej filtrowanie zielonych wierzchołków tworzy trzy drzewa. Zauważ, że w drugim z tych drzew obecne są krawędzie, których nie było w oryginalnym drzewie.

Szybszy algorytm konstrukcji drzewa

Tworzenie optymalnego drzewa filogenetycznego może trwać dość długo. Należy zaimplementować szybszy algorytm, który konstruuje drzewo o względnie niskim koszcie. Do przykładowych testów (`nazwa_testu.fasta`) dołączony jest orientacyjny wymagany koszt wyprodukowanego drzewa i czas działania rozwiązania wzorcowego (w pliku `nazwa_testu.txt`).

Wzorcowe rozwiązanie używa następującego algorytmu. Drzewo tworzymy stopniowo, zaczynając od korzenia i po kolei dodając wierzchołki odpowiadające kolejnym próbkom.

- Korzeń drzewa odpowiada próbce o najwcześniejszej dacie pobrania.
- Dla każdej z pozostałych sekwencji x, od najstarszej do najnowszej daty pobrania próbki, wybieramy jej rodzica w następujący sposób:

- losujemy dowolny wierzchołek v drzewa skonstruowanego do tej pory;
- niech d oznacza odległość edycyjną między x a v (dokładniej: między sekwencją x a sekwencją przechowywaną w wierzchołku v); jeśli wszyscy sąsiedzi v (rodzic i dzieci) mają odległość edycyjną do x co najmniej d, v staje się kandydatem; w przeciwnym razie przechodzimy do sąsiada o najmniejszej odległości edycyjnej do x i powtarzamy operację.

Powyższe poszukiwanie kandydata (losowanie i przechodzenie w stronę mniejszej odległości edycyjnej, aż znajdziemy kandydata na rodzica) powtarzamy kilkukrotnie (3 razy) i wybieramy najlepszego spośród kandydatów. Warto się upewnić, że nie liczymy wielokrotnie odległości edycyjnej między tą samą parą sekwencji.

Wymagania

Wymagane jest zdefiniowanie klasy `Tree` z następującymi metodami:

- `edges()` – zwraca listę krawędzi w drzewie; każda krawędź jest parą napisów (`tuple`) postaci `(id_próbki_rodzica, id_próbki_dziecka)`;
- `filter(country)` – zwraca listę drzew powstałych przez operację filtrowania.

Główną częścią zadania jest napisanie następujących funkcji (w nawiasach podane są orientacyjne liczby punktów za daną funkcję).

- `read_data(filename)` – wczytuje dane o próbkach z pliku FASTA o podanej nazwie i sortuje je po dacie pobrania próbki (dokładniejszy opis znajduje się w komentarzu w pliku `phylogeny.py`). **(3 pkt)**
- `construct_optimal_tree(samples)` – tworzy optymalne drzewo filogenetyczne dla podanej listy próbek. **(8 pkt)**
- `Tree.filter(self, country)` – filtruje drzewo; zwraca listę drzew. **(9 pkt)**
- `construct_approximate_tree(samples)` – tworzy (szybko) drzewo filogenetyczne o względnie niskim koszcie. **(10 pkt)**

Oczekiwane złożoności poszczególnych funkcji są opisane w pliku `phylogeny.py`.
Można korzystać z kodów z wykładów i laboratoriów.