Lista 3: Proste algorytmy iteracyjne

Witold Dyrka, Jakub Wojciechowski

Kwiecień 2021

Rozwiązania zadań, o ile nie wskazano inaczej, proszę przedstawić w formie kodu źródłowego w Pythonie (plik tekstowy z rozszerzeniem `.py`). Pliki z kodem źródłowym — osobne dla każdego zadania — należy umieszczać w kontenerze odbiorczym bezpośrednio (pliki spakowane nie będą akceptowane).

Każdy plik z kodem źródłowym proszę opatrzyć komentarzem informującym o autorstwie. Powyższy zapis jest traktowany jako równoważny oświadczeniu, że kod został napisany samodzielnie. W przeciwnym przypadku komentarz powinien określać rodzaj i zakres udziału zewnętrznego oraz precyzyjnie wskazywać jego źródła. Umiejętne *oraz* dobrze udokumentowane korzystanie ze źródeł zewnętrznych nie obniża wartości samego rozwiązania, jednak nadmierne lub, co gorsze, bezrefleksyjne użycie materiałów zewnętrznych może negatywnie wpłynąć na proces nauki programowania.

Kod powinien być zredagowany zgodnie z zasadami przedstawionymi na wykładzie. Dobre praktyki dotyczące redagowania kodu w Pythonie opisuje dokument *PEP8 – Style Guide for Python Code*[1].

Ponadto dla każdej z funkcji implementujących algorytm napisz funkcję testującą, która na kilku przykładach zademonstruje poprawne działanie testowanej funkcji. Funkcja testująca powinna wyświetlać czytelne komunikaty. Do automatyzacji weryfikacji poprawności warto wykorzystać asercje.

Przesyłany do oceny kod źródłowy powinien być opatrzony komentarzami. Ogólne zasady tworzenia komentarzy dokumentacyjnych w Pythonie zawiera dokument *PEP257 – Docstring Conventions*[2]. Polecamy trzymać się stylu komentarzy dokumentacyjnych Google[3] albo NumPy[4]. Porównanie obu stylów znajdziemy w dokumentacji generatora dokumentacji Sphinx[5].

[1] https://www.python.org/dev/peps/pep-0008/

[2] https://www.python.org/dev/peps/pep-0257/

[3] https://google.github.io/styleguide/pyguide.html

[4] https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard

[5] https://www.sphinx-doc.org/en/master/usage/extensions/napoleon.html

Zad. 1

Zaimplementuj algorytm sortowania przez wstawianie albo sortowania bąbelkowego. Wybór musi być jasno określony w treści rozwiązania (np. w nazwie funkcji lub komentarzu dokumentującym). Algorytmy nie zostały przedstawione na wykładzie, jednak są bardzo dobrze udokumentowane w podręcznikach i zasobach edukacyjnych internetu (choćby Wikipedii). Przedstawiając rozwiązanie, należy wskazać źródło.

a) Zacznij od projektu algorytmu (pseudokod lub schemat blokowy) oraz analizy stanów (przynajmniej jeden przykład; wzorując się na skrypcie Wykładu 5, można ograniczyć się do kontroli stanu jedynie w kluczowych punktach algorytmu).

b) Następnie zaimplementuj algorytm w postaci funkcji w języku Python oraz napisz dla niej funkcję testującą. Pamiętaj o komentarzach (w tym o komentarzach dokumentujących).

c) Uzasadnij lub zademonstruj, że złożoność czasowa algorytmu jest w najlepszym przypadku liniowa, a w najgorszym — kwadratowa.

Rozwiązania podpunktów a) i c) zadania należy przedstawić w sprawozdaniu w formacie PDF.

Zad. 2

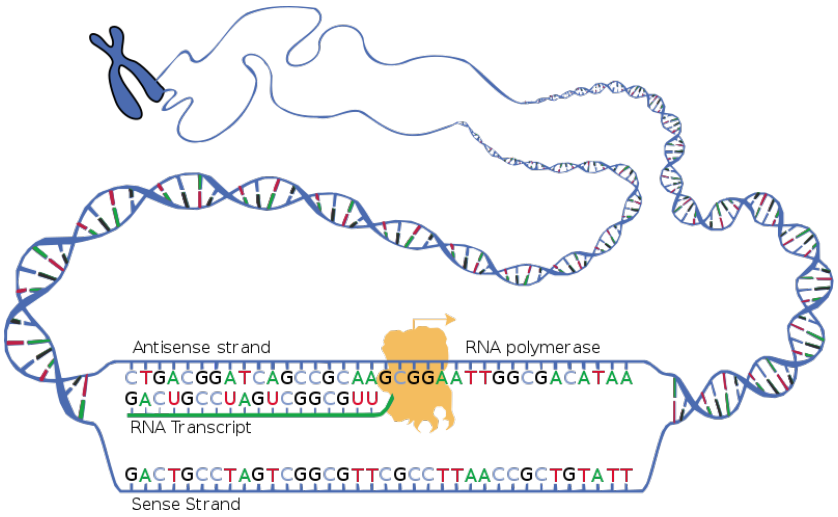
Zaimplementuj w Pythonie algorytm z Zad. 5 z Listy 1 — sprawdzanie przynależności podanego łańcucha tekstu (napisu) do języka $a^n b^n$. Napisz funkcję testującą. Pamiętaj o komentarzach dokumentujących.

Zad. 3

Zaprojektuj i zaimplementuj algorytmy rozwiązujące poniższe problemy z zakresu bioinformatyki.

Transkrypcja (Rys. 1) jest początkowym etapem ekspresji genu, podczas którego na bazie cząsteczki DNA syntezowana jest cząsteczka RNA. Cząsteczka DNA składa się z dwu nici: kodującej (nici sensownej), biegnącej w kierunku 5' do 3', oraz komplementarnej do niej nici matrycowej (antysensownej), biegnącej w kierunku przeciwnym. Podczas transkrypcji białko zwane polimerazą RNA odczytuje sekwencję nici niekodującej (biegnącej w kierunku 3' do 5') i na jej podstawie syntezuje nową cząsteczkę RNA

(w kierunku 5' do 3'). Zatem kolejność zasad w nowo powstającym łańcuchu jest odwrócona względem kolejności w cząsteczce stanowiącej matrycę i taka sama jak w nici kodującej.



Rysunek 1: Schemat procesu transkrypcji DNA. Źródło: https://commons.wikimedia.org/wiki/File:DNA_transcription.svg (domena publiczna).

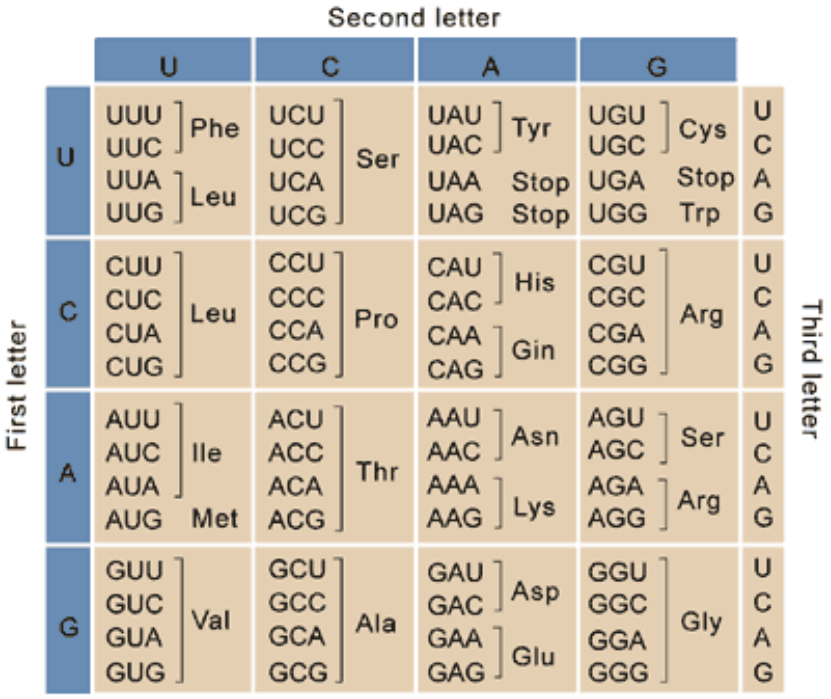
a) Dla sekwencji nici kodującej DNA znajdź sekwencję nici matrycowej.

b) Dla sekwencji nici matrycowej DNA znajdź sekwencję RNA.

Kolejnym etapem ekspresji genu jest *translacja*, czyli synteza białka na podstawie sekwencji RNA, a konkretniej — mRNA. Każdy aminokwas w sekwencji białka kodowany jest przez trzy nukleotydy (Rys. 2). Takie nienakładające się trójki nukleotydów nazywamy kodonami. Jak łatwo zauważyć, istnieje 64 możliwych kodonów, podczas gdy większość białek składa się z 20 rodzajów aminokwasów. Dzięki tym *nadmiarowym* kodonom zmiana pojedynczego nukleotydu nie zawsze skutkuje mutacją w białku. Oprócz kodonów przypisanych do konkretnych aminokwasów istnieją też kodony Stop, które sygnalizują koniec translacji, oraz kodon AUG, pełniący podwójną rolę: kodonu rozpoczynającego transkrypcję oraz kodującego aminokwas metioninę.

c) Dla sekwencji mRNA znajdź sekwencję kodowanego przez nie białka. Sekwencję białka zapisz używając 1- albo 3-literowych identyfikatorów aminokwasów[6].

[6] https://pl.wikipedia.org/wiki/Aminokwasy_bia%C5%82kowe



Rysunek 2: Standardowy kod genetyczny. Źródło: https://commons.wikimedia.org/wiki/File:06_chart_pu3.png (domena publiczna).

Rozwiązanie Zadania 3 powinno zawierać zestaw funkcji rozwiązujących poszczególne problemy: dla zadanej sekwencji wejściowej funkcje powinny zwracać oczekiwane sekwencje wynikowe. Należy zwrócić uwagę na poprawność zarówno danych wejściowych, jak i wyjściowych. Funkcje należy oczywiście opatrzyć komentarzami dokumentującymi. Należy napisać funkcje testujące, które zademonstrują poprawne rozwiązanie problemów. Pseudokod i schemat blokowy *nie* są wymagane, podobnie jak analiza stanów.