

SŁOWNIKOWE STRUKTURY DANYCH I ICH ZASTOSOWANIA

Marcin Piątkowski^{1,2}, Magdalena Mroczek^{1,3}, Marek Nowicki^{1,2}

⁽¹⁾ Interdyscyplinarne Centrum Modelowania

Matematycznego i Komputerowego, Uniwersytet Warszawski

⁽²⁾ Wydział Matematyki i Informatyki,

Uniwersytet Mikołaja Kopernika w Toruniu

⁽³⁾ Department of Biomedicine, University Hospital Basel

and University of Basel, 4031 Basel, Switzerland

martinp@mat.umk.pl, m.mroczek888@gmail.com, faramir@mat.umk.pl

Abstract. Indexing data structures such as tries and suffix trees are fundamental in the analysis of textual data, including biological data, such as DNA. We present a short introduction to indexing data structures, together with possible applications to solving competitive programming tasks. Moreover, practical applications related to the identification of HPV viruses utilised in HPV-KITE software tool are presented, and possible improvements by integrating a trie-based approach to enhance search efficiency are discussed.

1. Wstęp

Wyszukiwanie wzorca (*pattern matching*) jest jednym z najpopularniejszych problemów algorytmicznych. Jego sformułowanie jest na tyle proste, że może pojawić się już na wczesnym etapie nauczania algorytmiki. Z drugiej strony jego praktyczne zastosowanie w takich dziedzinach jak analiza tekstu (w tym również LLM - ang. *Large Language Model* - duży model językowy), rozpoznawanie obrazów czy bioinformatyka są trudne do przecenienia.

Wyróżniamy dwa podstawowe warianty problemu wyszukiwania wzorca:

1. Znany jest poszukiwany wzorec, dane do przeszukania nie są znane.
2. Znane są dane do przeszukiwania, poszukiwany wzorec nie jest znany.

W pierwszym przypadku wykorzystywane są takie algorytmy jak KMP, Boyera-Moore'a, Aho-Corasick, itp. polegające na wstępnym przetworzeniu wzorca w celu ułatwienia wyszukiwania. W drugim przypadku używane są zazwyczaj struktury indeksujące takie

jak drzewa sufiksowe, tablice sufiksowe, drzewa trie, drzewa n-gramów, itp. Wykorzystują one specyficzne cechy kluczy reprezentowanych w postaci łańcuchów znaków, dzięki czemu są dużo wydajniejsze od algorytmów i struktur danych ogólnego przeznaczenia. Oprócz wyszukiwania zadanego klucza pozwalają również znajdować listę wszystkich kluczy o zadanym prefiksie, listę kluczy pasujących do podanego wyrażenia regularnego, i wiele innych. Choć są to zagadnienia nieco bardziej skomplikowane od klasycznego problemu *pattern-matchingu*, mogą być z powodzeniem wykorzystane w ramach przygotowywania uczniów do udziału w Olimpiadzie informatycznej [5] czy Międzynarodowej Olimpiadzie Informatycznej [6].

Omawiane w pracy struktury danych mają również kluczowe zastosowania praktyczne, m.in. w bioinformatyce, gdzie wyróżnia się dwie podstawowe metody identyfikacji sekwencji: dopasowanie (ang. alignment) i metodę wzorca (ang. k-mer). Metoda dopasowania polega na przyrównaniu danej sekwencji do sekwencji referencyjnych z bazy danych. Metoda dopasowania jest bardziej dokładna i pozwala na zidentyfikowanie podobieństw ewolucyjnych. Wadą jest to, że dana sekwencja musi znajdować się w bazie danych, żeby być właściwie dopasowana, co jest mało praktyczne w przypadku szybko mutujących wirusów. Kolejną wadą tej metody jest jej czasochłonność i wymóg dużej mocy obliczeniowej komputera. Metoda wyszukiwania wzorca jest szybsza, tańsza i prostsza, a w dodatku sekwencja nie musi znajdować się w referencyjnej bazie danych. Obie metody stosowane są do wykrywania potencjalnych zakażeń człowieka przez patogeny.

W pracy przedstawimy wprowadzenie do drzew trie, drzew sufiksowych oraz drzew n-gramów. Następnie zaprezentujemy możliwość ich użycia do rozwiązania wybranych zadań olimpijskich oraz możliwości praktycznego wykorzystania w projektach bioinformatycznych. Z uwagi na ograniczony rozmiar publikacji, przedstawione zagadnienia zostały omówione w sposób skrótowy i ogólny. Szczegółowe analizy oraz pogłębione rozważania wykraczają poza ramy niniejszego opracowania, dlatego zainteresowanych czytelników odsyłamy do [1], [2], [4], [7], [8].

2. Drzewa Trie

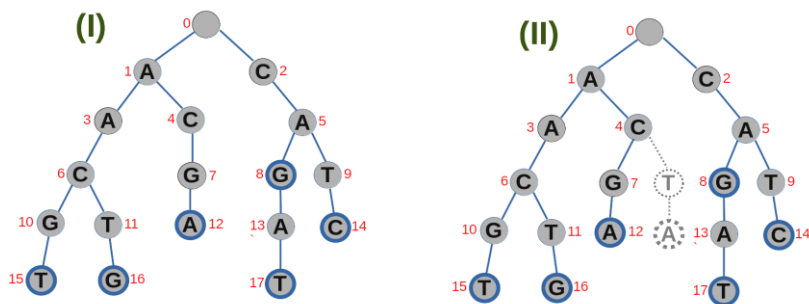
Drzewo trie (od angielskiego *reTRIEval*) jest drzewem wyszukiwań zbudowanym na podstawie kluczy reprezentowanych jako łańcuchy znaków nad ustalonym alfabetem. Każdy węzeł drzewa (poza korzeniem) posiada dokładnie jednego rodzica oraz odnośniki do węzłów potomnych indeksowane znakami alfabetu (lub odnośniki puste w przypadku braku węzłów potomnych). Każdy niepusty odnośnik prowadzi do dokładnie jednego węzła potomnego, zatem węzeł ten może być oznaczony za pomocą znaku odpowiadającego odnośnikowi (patrz Rysunek 1). Klucze w drzewie trie są reprezentowane przez ścieżki prowadzące od korzenia do węzłów zwanych *terminalnymi*. Węzły te mogą zawierać dodatkową informację powiązaną z kluczem.

Podstawowymi operacjami dostępnymi dla drzewa Trie są: wyszukiwanie klucza, dodanie nowego klucza oraz usunięcie istniejącego klucza.

Wyszukiwanie klucza polega na przeglądaniu drzewa rozpoczynając od korzenia i podążając za odnośnikami odpowiadającymi kolejnym znakom klucza. Wyszukiwanie zakończy się sukcesem, jeśli po osiągnięciu ostatniego znaku klucza przeglądanie drzewa zakończy się w węźle terminalnym. W przeciwnym przypadku (przeglądanie zakończyło się w węźle innym niż końcowy lub aktualnie przetwarzany węzeł nie posiada potomka dla kolejnego znaku klucza) wyszukiwanie zakończy się porażką.

Dla przykładu wyszukiwanie klucza CAG polega na odwiedzaniu kolejno wierzchołków o numerach 0, 2, 5 i 8 (zobacz Rysunek 1 (I)). Wierzchołek 8 jest wierzchołkiem terminalnym, zatem wyszukiwanie zakończyło się sukcesem. W przypadku klucza ACG odwiedzamy kolejno wierzchołki 0, 1, 4 i 7. Wierzchołek 7 nie jest wierzchołkiem terminalnym, zatem wyszukiwanie zakończyło się porażką. Podobnie będzie dla klucza ACTG. W tym wypadku odwiedzamy wierzchołki 0, 1 oraz 4. Wierzchołek 4 nie posiada potomka z etykietą T, zatem nie jesteśmy w stanie kontynuować wyszukiwania.

Łatwo zauważyć, że niezależnie od wyniku wyszukiwania, liczba odwiedzonych węzłów drzewa nie przekracza liczby znaków klucza plus jeden.



Rysunek 1. Struktura drzewa trie dla zbioru kluczy {AACGT, AACTG, ACGA, CAG, CAGAT, CATC} (i) – węzły terminalne oznaczone pogrubioną krawędzią. Dodanie do drzewa nowego klucza ACTA (II).

Operację wstawiania nowego klucza K do drzewa trie możemy podzielić na dwa etapy:

1. Znalezienie najdłuższego prefiksu K już znajdującego się w drzewie. Postępujemy identycznie jak w przypadku wyszukiwania odwiedzając węzły drzewa dla kolejnych znaków K. Jeżeli etap ten zakończy się przetworzeniem wszystkich znaków K (t.j. K jest prefiksem innego, już reprezentowanego w drzewie klucza), ostatni odwiedzony węzeł drzewa zaznaczamy jako końcowy.

2. Dodanie nowych węzłów dla brakujących elementów K. Jeśli etap 1 zakończy się przed osiągnięciem ostatniego znaku K, konieczne jest dodanie nowych węzłów (rozpoczynając od ostatniego odwiedzonego węzła drzewa) odpowiadających kolejnym znakom K i połączonych zależnością rodzic-potomek.

Dla przykładu, dodanie klucza ACTA będzie polegało na odwiedzeniu wierzchołków 0, 1 oraz 4, a następnie dodaniu dwóch nowych wierzchołków dla dwóch ostatnich znaków klucza (zobacz Rysunek 1 (II)). W przypadku dodawania klucza CAT odwiedzone zostaną kolejne wierzchołki 2, 5 oraz 9. Dodatkowo wierzchołek 9 zostanie oznaczony jako terminalny.

Łatwo zauważyć, że we wszystkich przypadkach należy odwiedzić istniejący lub utworzyć nowy węzeł w drzewie dla każdego znaku dodawanego klucza.

Operację usunięcia klucza K z drzewa rozpoczynamy od wyszukania jego wystąpienia. Jeśli odpowiadający mu wierzchołek terminalny nie jest liściem (t.j. ma węzły potomne) wystarczy zaznaczyć, że nie jest już węzłem terminalnym. W przeciwnym przypadku konieczne będzie usunięcie części ścieżki w drzewie od ostatniego węzła posiadającego więcej niż jednego potomka aż do znalezionej węzła terminalnego.

Dla przykładu, usunięcie klucza CAG polega na zaznaczeniu, że wierzchołek 8 nie jest już wierzchołkiem terminalnym. W przypadku usuwania klucza CATC konieczne będzie natomiast usunięcie wierzchołków 9 oraz 14 (zobacz Rysunek 1).

Dodatkowymi operacjami dostępnymi dla drzew trie są między innymi:

1. Zwrócenie zbioru wszystkich reprezentowanych kluczy. W tym celu konieczne jest przejście drzewa dowolną metodą z jednoczesnym zapamiętywaniem fragmentów klucza odpowiadających aktualnie przeglądanej ścieżce.
2. Zwrócenie zbioru wszystkich kluczy o ustalonym prefiksie P. W tym celu należy znaleźć węzeł drzewa odpowiadający wystąpieniu P, a następnie powtórzyć operację (1) dla uzyskanego poddrzewa.
3. Wyszukiwanie z użyciem symboli wieloznacznych. W tym celu, w czasie przeglądania drzewa możliwe jest wybranie dowolnego potomka węzła odpowiadającego symbolowi wieloznacznemu występującemu w kluczu.

Poniżej przedstawiamy podsumowanie wybranych cech drzew trie. Czytelników zainteresowanych szczegółowymi dowodami odsyłamy do literatury przedmiotu.

1. Struktura drzewa trie nie zależy od kolejności wstawiania i usuwania kluczy. Dla każdego zbioru kluczy istnieje unikatowe drzewo trie.
2. Liczba dostępów do tablicy węzłów przy wyszukiwaniu lub wstawianiu klucza długości n wynosi co najwyżej $n+1$.
3. Średnia liczba węzłów sprawdzanych przy nieudanym wyszukiwaniu w drzewie trie zbudowanym dla N losowych kluczy nad alfabetem rozmiaru R wynosi $O(\log N)$.

3. Drzewa sufiksowe

Drzewo sufiksowe to szczególny rodzaj drzewa trie, w którym kluczami są wszystkie sufiksy ustalonego słowa (łańcucha znaków). Znajduje ono praktyczne zastosowanie między innymi w wyszukiwaniu wzorców, zaawansowanej analizie tekstu, kompresji danych oraz bioinformatyce.

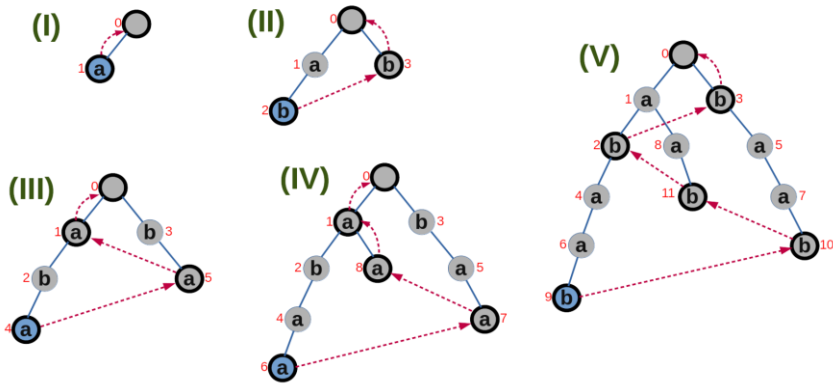
Drzewo sufiksowe można zbudować poprzez wstawianie kolejnych sufiksów danego słowa przy użyciu standardowych metod znanych z drzew trie. Istnieją jednak bardziej efektywne techniki konstrukcji, które wykorzystują szczególne właściwości zbioru sufiksów. Przykładem takiej metody jest algorytm Ukkonena [1].

Algorytm ten realizuje przyrostową (*on-line*) konstrukcję drzewa sufiksowego. Dla łańcucha znaków $w = x_1x_2\dots x_n$ budowane są kolejno drzewa sufiksowe $T_1, T_2, \dots, T_n$ odpowiadające drzewom sufiksowym kolejnych prefiksów łańcucha w , t.j. $x_1, x_1x_2, \dots, x_1x_2\dots x_n$. (patrz Rysunek 2). Dzięki przyrostowej budowie drzewa dla kolejnych znaków tekstu nie jest konieczna znajomość całego tekstu z góry.

Podstawą działania algorytmu Ukkonena jest wykorzystanie następującej zależności: jeśli zbudowane zostało drzewo T_k odpowiadające łańcuchowi znaków $x_1x_2\dots x_k$, drzewo T_{k+1} odpowiadające $x_1x_2\dots x_kx_{k+1}$ może być uzyskane z T_k przez dodanie potomków odpowiadających znakowi x_k wszystkim wierzchołkom terminalnym T_k . Procedurę tworzenia drzewa sufiksowego możemy zatem podzielić na fazy odpowiadające dodawaniu pojedynczego znaku.

W celu ułatwienia procesu rozszerzania drzewa używamy pojęcia tzw. *suffix linku* – wirtualnej krawędzi łączącej wierzchołek terminalny odpowiadający sufiksowi $x_1x_2\dots x_k$ z wierzchołkiem odpowiadającym sufiksowi $x_2\dots x_k$ (czerwone przerywane krawędzie na Rysunku 2). Ponadto wyróżniamy *najgłębszy liść* – wierzchołek terminalny odpowiadający najdłuższemu łańcuchowi reprezentowanemu w drzewie (na Rysunku 2. zaznaczony kolorem niebieskim).

Pojedyncza faza rozszerzenia drzewa o znak x_{k+1} polega na odwiedzeniu kolejnych wierzchołków terminalnych, rozpoczynając od najgłębszego liścia oraz wykorzystując *suffix linki*. Każdemu z nich dodajemy potomka odpowiadającego x_{k+1} . Nowo dodane wierzchołki stają się wierzchołkami terminalnymi (jednocześnie ich przodkowie przestają być wierzchołkami terminalnymi).



Rysunek 2. Kolejne etapy budowy drzewa sufikсового dla słowa *abaab*. Wierzchołki terminalne zostały zaznaczone pogrubionym konturem. W celu podniesienia czytelności *suffix linki* zostały zaznaczone wyłącznie dla wierzchołków terminalnych.

W celu ilustracji powyższej procedury omówimy rozszerzenie drzewa przedstawionego na Rysunku 2 (IV) o symbol *b*. Rozpoczynamy od wierzchołka 6, do którego dodajemy nowy wierzchołek 9 jako potomka i oznaczamy go jako terminalny. Korzystając z *suffix linku* odwiedzamy wierzchołek 7, dodajemy mu jako potomka wierzchołek 10, zaznaczamy go jako terminalny i uzupełniamy *suffix link* między wierzchołkami 9 oraz 10. Ponownie korzystając z *suffix linku* odwiedzamy wierzchołek 8, dodajemy mu jako potomka wierzchołek 11, zaznaczamy go jako terminalny i uzupełniamy *suffix link* między wierzchołkami 10 oraz 11. Następnie, korzystając z *suffix linku* odwiedzamy wierzchołek 1. Ponieważ posiada on już potomka odpowiadającego *b* (wierzchołek 2), wystarczy zaznaczyć go jako wierzchołek terminalny oraz uzupełnić *suffix link* między wierzchołkami 11 oraz 2¹. Ostatnim odwiedzanym wierzchołkiem jest wierzchołek 0. W tym wypadku również wystarczy zaznaczyć wierzchołek 3 jako terminalny. Nie ma potrzeby tworzenia *suffix linku* między wierzchołkami 2 oraz 3, ponieważ został on już stworzony w fazie (II). Ostateczna postać drzewa została przedstawiona na Rysunku 2 (V).

4. Drzewa n-gramów

Drzewo n-gramów jest modyfikacją drzewa sufikсового, w którym zamiast reprezentować wszystkie sufiksy rozważanego łańcucha znaków, ograniczamy się

¹ Można zauważyć, że w dalszej części aktualnej fazy algorytmu nie będzie potrzeby tworzenia nowych wierzchołków ani uzupełniania *suffix linków*, ponieważ zostały one utworzone we wcześniejszych fazach.

wyłącznie do podciągów o ustalonej długości n . Takie podejście pozwala na zwięzłe modelowanie i efektywne przetwarzanie danych sekwencyjnych znacznych rozmiarów.

Ze względu na możliwość szybkiego wyszukiwania i analizy wzorców drzewa n -gramów wykorzystywane są między innymi w modelowaniu języka naturalnego (NLP, ang. *Natural Language Processing*), systemach autouzupełniania, systemach rozpoznawania mowy, kompresji danych czy klasyfikacji tekstów. Zliczanie i analiza wspólnych n -gramów pozwalają wykorzystać drzewa n -gramów również w porównywaniu dokumentów i wykrywaniu plagiatów.

W bioinformatyce efektywne reprezentacje zbiorów n -gramów (zwanymi również *k-mers*) znajdują zastosowanie do analizy sekwencji DNA, RNA i białek – umożliwiają identyfikację powtarzalnych motywów, wykrywanie mutacji, porównywanie genomów, a także tworzenie genetycznych „podpisów” organizmów. Ponadto, pełnią one istotną rolę m.in. w diagnostyce, metagenomice i badaniach filogenetycznych.

W celu zbudowania drzewa n -gramów dla ustalonego łańcucha znaków możemy wykorzystać modyfikację algorytmu Ukkonena. Ponieważ docelowo drzewo będzie zawierało wyłącznie ciągi długości n , wszystkie liście będą znajdować się na tej samej głębokości. Dodatkowo, wierzchołkami terminalnymi będą wyłącznie liście. Dzięki temu rozbudowa ścieżek w drzewie może być przerywana po osiągnięciu ustalonej głębokości.

Podobnie jak w oryginalnej wersji algorytmu, w celu przyspieszenia aktualizacji drzewa wykorzystywane są *suffix linki*, zaś punktem startowym każdej fazy rozszerzania (odpowiednikiem najgłębszego liścia) jest wierzchołek terminalny odpowiadający ostatniemu dodanemu do drzewa n -gramowi.

W niektórych zastosowaniach wystarczająca jest informacja o obecności n -gramu w zbiorze, podczas gdy inne wymagają także znajomości częstości jego występowania. W tym celu do liści drzewa można dołączyć liczniki, które będą aktualizowane w trakcie przetwarzania danych.

5. Zastosowania w edukacji

Opisane powyżej struktury danych mogą być z powodzeniem wykorzystane jako części rozwiązań zadań pojawiających się na Olimpiadzie Informatycznej [5] oraz Międzynarodowej Olimpiadzie Informatycznej [3]. Poniżej zamieszczamy listę kilku zadań dostępnych w popularnym serwisie Online Judge [6] wraz ze szkicami idei rozwiązań opartych o drzewa trie oraz drzewa sufiksowe.

12506 - Shortest Names

Dane wejściowe sformatowane są w postaci listy słów składających się z małych liter alfabetu łacińskiego. Dla każdego słowa na liście należy znaleźć długość najkrótszego prefiksu, który odróżni je jednoznacznie od pozostałych. Rozwiązaniem zadania jest suma długości znalezionych prefiksów.

Aby rozwiązać to zadanie należy umieścić wczytane słowa w drzewie trie. Następnie dla każdej ścieżki prowadzącej od korzenia do wierzchołka terminalnego należy znaleźć pierwszy v wierzchołek za ostatnim rozgałęzieniem (od wierzchołka v do wierzchołka terminalnego powinna prowadzić pojedyncza ścieżka bez rozgałęzień). Jako rozwiązanie należy zwrócić sumę długości wszystkich znalezionych ścieżek.

1223 – Editor

Dane wejściowe sformatowane są w postaci listy słów składających się z małych liter alfabetu łacińskiego. Dla każdego wczytanego słowa w należy znaleźć długość najdłuższego pod słowa posiadającego co najmniej dwa wystąpienia wewnątrz w .

Aby rozwiązać to zadanie należy zbudować drzewo sufiksowe dla wczytanego słowa w , a następnie znaleźć maksymalną długość ścieżki od korzenia do wierzchołka posiadającego co najmniej dwóch potomków.

11512 – GATTACA

Dane wejściowe sformatowane są w postaci listy słów nad alfabetem $\{A, C, G, T\}$. Dla każdego wczytanego słowa w należy znaleźć najdłuższe pod słowo posiadające co najmniej dwa wystąpienia wewnątrz w oraz wyznaczyć liczbę jego wystąpień. Jeśli w zawiera więcej pod słów o tej własności (tej samej maksymalnej długości) należy wydrukować najmniejsze w porządku leksykograficznym.

Aby rozwiązać to zadanie należy zbudować drzewo sufiksowe dla wczytanego słowa w , a następnie znaleźć maksymalną ścieżkę od korzenia do wierzchołka x posiadającego co najmniej dwóch potomków. Liczbę wystąpień znalezionej pod słowa określamy jako liczbę wierzchołków terminalnych znajdujących się w podrzewie rozpoczynającym się w x . Struktura drzewa sufiksowego dodatkowo ułatwia znalezienie najmniejszego leksykograficznie pod słowa o szukanej własności.

760 – DNA Sequencing

Dane wejściowe sformatowane są w postaci listy par słów nad alfabetem $\{a, c, g, t\}$. Dla każdej pary wczytanych słów w_1 oraz w_2 należy znaleźć najdłuższe pod słowo występujące zarówno w w_1 jak i w_2 .

Aby rozwiązać to zadanie należy zbudować drzewo sufiksowe zawierające jednocześnie wszystkie sufiksy w_1 oraz w_2 . W tym celu wystarczy wykorzystać algorytm Ukkonena kolejno dla w_1 i w_2 tworząc jedno wspólne drzewo sufiksowe. W celu rozróżnienia wierzchołków terminalnych odpowiadających w_1 oraz w_2 konieczne będzie dodanie flagi pozwalającej poprawnie zidentyfikować słowo źródłowe. Dla tak skonstruowanego drzewa trzeba znaleźć wierzchołek x położony najdalej od korzenia o tej własności, że podrzewo rozpoczynające się w x zawiera wierzchołki terminalne zarówno dla w_1 jak i w_2 .

11107 – Life forms

Dane wejściowe sformatowane są w postaci kilku list słów nad alfabetem łacińskim. Dla każdej listy słów należy znaleźć najdłuższe pod słowo występujące w co najmniej połowie słów znajdujących się na liście.

Aby rozwiązać to zadanie uogólnimy rozwiązanie zadania *DNA Sequencing* budując drzewo sufiksowe dla listy wczytanych słów (konieczny będzie większy zakres flag dla wierzchołków terminalnych aby prawidłowo określić słowo źródłowe). Dla tak skonstruowanego drzewa trzeba znaleźć wierzchołek x położony najdalej od korzenia o tej własności, że podrzewo rozpoczynające się w x zawiera wierzchołki terminalne odpowiadające co najmniej połowie słów z rozważanej listy.

6. Zastosowania w praktyce

Jednym z możliwych praktycznych zastosowań opisanych powyżej struktur danych jest wyszukiwanie podobieństwa między sekwencjami. Stosowane jest to przede wszystkim w celu wykrycia zakażenia wirusami, których sekwencje RNA/DNA przenikają do sekwencji ludzkich przy pobieraniu wymazu, m.in. wirusów z grupy HPV (wirusy brodawczaka ludzkiego).

HPV stanowi przykład jednej z najczęstszych infekcji wirusowych. Szacuje się, że około 8 na 10 osób zakażonych jest wirusem HPV. Istnieje ponad 200 różnych typów HPV. O ile niektóre typy HPV powodują powstanie jedynie łagodnych zmian przypominających brodawki, inne mogą prowadzić do powstania złośliwych nowotworów, zwłaszcza w szyjce macicy. Wirusy HPV nie wnikają do krwiobiegu, a pozostają w komórkach nabłonka. Większość zakażeń w początkowej fazie przebiega bezobjawowo. W celu wykrycia wirusa HPV u kobiet najczęściej przeprowadza się wymazy z szyjki macicy, izoluje DNA, a następnie amplifikuje się sekwencje wirusa HPV, żeby sprawdzić czy znajduje się on w próbce. Tymczasem nowe metody sekwencjonowania wielkoskalowego mogą jednocześnie analizować całe sekwencje znajdujące się w próbce: nie tylko wybrane sekwencje wirusowe, ale sekwencje ludzkie, sekwencje wszystkich wirusów i bakterii. Jednak analiza tego typu danych jest czasochłonna i wymaga dużej mocy komputerowej. W związku z tym niezbędny jest rozwój nowych algorytmów, które pozwolą w sposób szybki i dokładny identyfikować sekwencje HPV i innych wirusów w bardzo dużych zbiorach sekwencji DNA. Do tego celu dobrze nadają się metody wyszukiwania wzorca, ze względu na ich szybkość i efektywność.

W ramach projektu HEAP (Human Exposome Assessment Platform), którego zadaniem jest badanie ekspozycji, czyli narażenia na czynniki środowiskowe, które wpływają na zdrowie w ciągu całego życia, stworzyliśmy rozwiązanie do wykrywania zakażenia pacjenta przez wirusy typu HPV. W tym celu stworzyliśmy narzędzie HPV-KITE [4], którego zadaniem jest wysokowydajne przetwarzanie sekwencji ludzkich.

W tym celu sekwencje wirusowe z bazy referencyjnej zawierającej wirusy HPV zostały podzielone kroczącym oknem na n-gramy ustalonej długości (domyślnie: 31-znakowe) i zapamiętane w zbiorze. Następnie sekwencja pacjenta, badana w celu znalezienia zakażenia, również została podzielona kroczącym oknem na n-gramy tej samej długości. Oprogramowanie następnie bada pokrycie poszczególnych wirusów HPV przez n-gramy ludzkie. Jeśli pokrycie było wyższe niż wcześniej ustalony próg to mówimy o potencjalnym zakażeniu. W HPV-KITE zastosowaliśmy zbiory (*set*) przechowujące ciągi znaków (*string*) do przechowywania danych z referencyjnej bazy i wykorzystując funkcję skrótu (*hash*) do porównywania zbiorów wygenerowanych z sekwencji ludzkich. Możliwe jest jednak zastosowanie omawianych w pracy struktur drzewiastych, co może pozwalać na użycie mniejszej ilości pamięci do przechowywania danych i szybsze znajdowanie podobieństw. HPV-KITE pozwala na przesiewowe badanie szerszej populacji i wyłapanie jednostek, które należy obserwować, aby wykryć chorobę na jak najwcześniejszym stadium. HPV-KITE jest znacznie szybszy i równie czuły i swoisty co metody oparte na przyrównaniu i nie wymaga dużej mocy obliczeniowej. Dzięki temu metodę tę można stosować również w krajach rozwijających się, gdzie jest limitowana ilość mocy obliczeniowej komputerów. Z biologicznego punktu widzenia, metodą tą można wykrywać nie tylko wirus HPV, ale sekwencje wielu wirusów i bakterii pochodzące z jednej próbki.

7. Podsumowanie

W pracy przedstawione zostało krótkie wprowadzenie do słownikowych struktur danych takich jak drzewa trie, drzewa sufiksowe czy drzewa n-gramów wraz z omówieniem ich wykorzystania do rozwiązywania zadań na poziomie Międzynarodowej Olimpiady Informatycznej. Nakreślona też została problematyka analizy danych biologicznych (m.in. DNA), gdzie dane biologiczne pochodzące z próbki i bazy referencyjnej są dzielone na sekwencje o określonej długości (n-gramy), a następnie badany jest stopień pokrycia pomiędzy tymi sekwencjami. Dodatkowo, przedstawiona została idea optymalizacji problemu przeszukiwania tak przygotowanych danych przez użycie drzew n-gramów.

Finansowanie

Niniejsza praca została częściowo wykonana w ramach projektu Human Exposome Assessment Platform (HEAP; Project No. 874662) finansowanego przez program ramowy Horyzont 2020 Unii Europejskiej.

Literatura

1. Crochemore M., Rytter W., *Jewels of Stringology*, World Scientific, 2003.

2. Halim S., Halim F., Effendy S., *Competitive Programming 4*, Morrisville, N.C, 2022
3. International Olympiad in Informatics, <https://ioinformatics.org/>.
4. Nowicki M, Mroczek M, Mukhedkar D, Bala P, Nikolai Pimenoff V, Arroyo Mühr LS. HPV-KITE: sequence analysis software for rapid HPV genotype detection. *Brief Bioinform.* 2025;26(2):bbaf155. doi:10.1093/bib/bbaf155.
5. Olimpiada Informatyczna, <https://oi.edu.pl/>.
6. Online Judge, <https://https://onlinejudge.org>.
7. Sedgwick R., Wayne K., *Algorytmy*, Helion, 2012.
8. Skiena S., Revilla M., *Wyzwania programistyczne*, Wydawnictwa Szkolne i Pedagogiczne, 2004.