# Algorytmy kombinatoryczne w bioinformatyce

wykład 1: wprowadzenie

prof. dr hab. inż. **Marta Kasprzak**
**Instytut Informatyki, Politechnika Poznańska**

## Treść wykładów

- Modelowanie problemów biologicznych za pomocą znanych problemów kombinatorycznych
- Grafy jako struktury danych powszechnie używane do kodowania instancji problemów bioinformatycznych
- Obecność błędów eksperymentalnych w instancji a złożoność obliczeniowa problemu dla wybranych przykładów
- Przykłady zastosowania znanych algorytmów kombinatorycznych do rozwiązania problemów biologicznych
- Przykłady nowych algorytmów dedykowanych problemom biologicznym, opartych na strukturach grafowych
- Operacje na sekwencjach znaków i algorytmy dla tych struktur

## Cel przedmiotu

- Przedstawienie wybranych klasycznych problemów i algorytmów kombinatorycznych stosowanych w bioinformatyce, w zagadnieniach związanych z poznawaniem i analizą pierwszorzędowej struktury DNA
- Wykształcenie umiejętności postrzegania problemu biologicznego w kategoriach algorytmicznych
- Ze względu na charakter przedmiotu poruszane problemy i algorytmy reprezentują zazwyczaj wczesne podejścia do rozwiązania danych problemów biologicznych. Problemy bywały wtedy często upraszczane, np. przez ograniczenie modelu błędów eksperymentalnych w danych. Podejścia nowsze, dla szerszego spektrum problemów i w bardziej realistycznym ujęciu danych zostaną przedstawione w ramach innych przedmiotów



## Zaliczenie przedmiotu

- Ocena końcowa z wykładów zostanie wystawiona na podstawie pisemnego sprawdzianu wiedzy
- Na każdy z pierwszych siedmiu wykładów przypadnie jedno pytanie (zadanie) i odpowiedź na nie zostanie oceniona w skali 0–2 pkt.
- Na ocenę wpływa nie tylko poprawność, ale i kompletność odpowiedzi. Ocenę pozytywną za całość można uzyskać po przekroczeniu połowy maksymalnej sumarycznej liczby punktów
- Obecność na wykładzie potwierdzona wpisem na liście obecności może podnieść ocenę za powiązane z nim pytanie
- Ocena z zajęć laboratoryjnych wystawiona zostanie na odrębnych zasadach

## Na styku biologii i informatyki

- Bioinformatyka jest dziedziną wiedzy obejmującą stosowanie modeli matematycznych oraz metod i narzędzi informatycznych do rozwiązania problemów biologicznych, głównie wywodzących się z biologii molekularnej
- Bioinformatyka a biologia obliczeniowa
- Olbrzymie zasoby danych pochodzących z badań nad m.in. kwasami nukleinowymi wymusiły zastosowanie do ich analizy komputerów, a wraz z tym poszukiwanie efektywnych metod obliczeniowych
- Wiele z istniejących algorytmów rozwiązujących problemy kombinatoryczne mogło zostać zastosowanych do rozwiązania odpowiednio zamodelowanych problemów biologicznych. Inne problemy wymagały stworzenia nowych algorytmów w oparciu o złożoność obliczeniową danego problemu



## Problemy kombinatoryczne

- *Kombinatoryka* jest obszarem matematyki dyskretnej obejmującym operacje na zbiorach elementów i ich atrybutach
- Z uwagi na dyskretną naturę informacji zakodowanej w kwasach nukleinowych, kombinatoryka idealnie nadaje się do modelowania powiązanych problemów biologicznych
- *Problem kombinatoryczny* wyrażony jest w postaci opisu instancji problemu i rozwiązania, którego w danej instancji poszukujemy
- Przykładowe problemy kombinatoryczne: problem plecakowy, problem komiwojażera, kolorowanie mapy, szeregowanie zadań w procesie produkcyjnym

## Problemy kombinatoryczne a teoria grafów

- Niektóre problemy kombinatoryczne mogą zostać w naturalny sposób zamodelowane za pomocą teorii grafów. Modele wywodzące się z teorii grafów są szczególnie popularne ze względu na czytelność opisywanych w ten sposób problemów oraz istnienie efektywnych algorytmów
- Najczęściej zbiór elementów staje się zbiorem wierzchołków w grafie, relacje pomiędzy elementami tworzą zbiór krawędzi/łuków, atrybuty elementów i ich relacji stają się wagami związanymi z wierzchołkami bądź krawędziami
- W zależności od problemu rozwiązaniem w takim grafie może być np. ścieżka obejmująca wszystkie wierzchołki lub ich część, nieuporządkowany podzbiór wierzchołków lub krawędzi, sumaryczna waga dla wybranego podzbioru



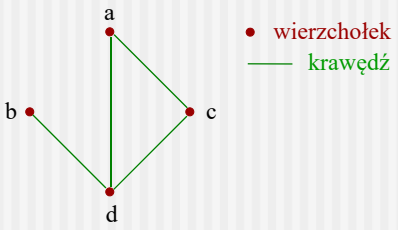
## Przypomnienie podstaw z teorii grafów

- Graf $G$ jest zbiorem wierzchołków $V$ z wyróżnionym podzbiorem par wierzchołków
- Graf *nieskierowany* – gdy pary wierzchołków są nieuporządkowane (są dwuelementowymi podzbiorami). Podzbiór par wierzchołków $E$ składa się z elementów $\{u,v\}$, $u,v \in V$, nazywanych *krawędziami*

Przykład grafu nieskierowanego:

$V = \{a, b, c, d\}$

$E = \{\{a,c\}, \{a,d\}, \{b,d\}, \{c,d\}\}$



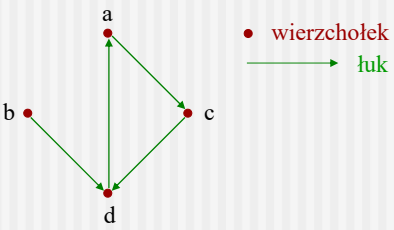
# Przypomnienie podstaw z teorii grafów

- Graf *skierowany* – gdy pary wierzchołków są uporządkowane (związane relacją). Podzbiór par wierzchołków $A$ składa się z elementów $(u,v)$, $u,v \in V$, nazywanych *łukami*

Przykład grafu skierowanego:

$V = \{a, b, c, d\}$

$A = \{(a,c), (b,d), (c,d), (d,a)\}$

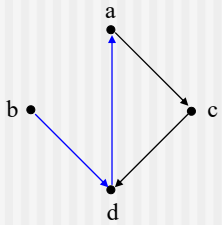





# Przypomnienie podstaw z teorii grafów

- *Ścieżką* nazywamy sekwencję wierzchołków $(v_1, v_2, ..., v_k)$ taką, że $\{v_i, v_{i+1}\} \in E$ (w grafie nieskierowanym) lub $(v_i, v_{i+1}) \in A$ (w grafie skierowanym), $i=1, ..., k-1$
- *Cykl* to ścieżka zamknięta — spełniająca warunek $\{v_k, v_1\} \in E$ lub $(v_k, v_1) \in A$
- *Ścieżka Hamiltona* to ścieżka zawierająca każdy wierzchołek grafu dokładnie raz
- *Ścieżka Eulera* to ścieżka przechodząca przez każdą krawędź (łuk) grafu dokładnie raz

Przykład ścieżki: $(b, d, a)$




Przykład cyklu: $(d, a, c)$

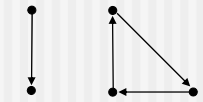
Ścieżka Hamiltona: $(b, d, a, c)$

Ścieżka Eulera: $(b, d, a, c, d)$

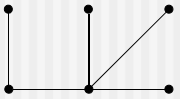
# Przypomnienie podstaw z teorii grafów

- Graf jest *spójny*, jeżeli nie można zbioru wierzchołków podzielić na dwa podzbiory $V_1$ i $V_2$ takie, że dla wszystkich $v_i \in V_1$, $v_j \in V_2$ spełniony jest warunek $\{v_i, v_j\} \notin E$ (w grafie nieskierowanym) lub $(v_i, v_j) \notin A \wedge (v_j, v_i) \notin A$ (w grafie skierowanym)
- *Drzewo nieskierowane* to graf spójny nieskierowany bez cykli
- *Drzewo skierowane* rozchodzące się to graf skierowany, którego nieskierowany odpowiednik jest drzewem i w którym od jednego z wierzchołków do każdego z pozostałych można poprowadzić ścieżkę

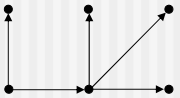
Graf niespójny:



Drzewo nieskierowane:



Drzewo skierowane rozchodzące się:

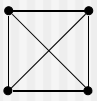




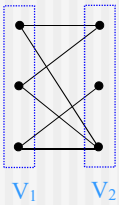
# Przypomnienie podstaw z teorii grafów

- Graf jest *pełny*, jeżeli dla wszystkich par $v_i, v_j \in V$ spełniony jest warunek $\{v_i, v_j\} \in E$ (w grafie nieskierowanym) lub $(v_i, v_j) \in A \wedge (v_j, v_i) \in A$ (w grafie skierowanym)
- Graf jest *dwudzielny*, jeśli zbiór wierzchołków można podzielić na dwa podzbiory $V_1$ i $V_2$ takie, że wszystkie krawędzie (łuki) grafu mają jeden wierzchołek w $V_1$ i drugi w $V_2$
- *Drzewem rozpinającym* grafu jest jego podgraf o tym samym zbiorze wierzchołków będący drzewem
- *Klika* to podgraf będący grafem pełnym

Graf pełny nieskierowany:



Graf dwudzielny nieskierowany:



## Problemy kombinatoryczne a teoria grafów

- Przykładowe problemy z życia wzięte i ich reprezentacja grafowa

| *Problem* | *Reprezentacja w grafie* | *Rozwiązanie w grafie* |
|---|---|---|
| szukanie powiązań w serwisach społecznościowych | wierzchołki — osoby<br>krawędzie — znajomość | maksymalna klika / kliki |
| rozpoznawanie wzorców w obrazie | wierzchołki — narożniki<br>krawędzie — kontury | izomorficzne podgrafy |
| planowanie elektryfikacji wsi | wierzchołki — wsie<br>krawędzie — odległości | minimalne drzewo rozpinające |

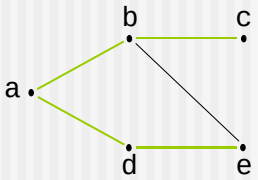


## Problemy kombinatoryczne

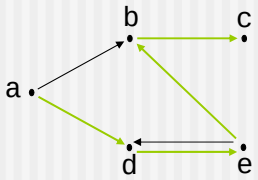
- Problemy kombinatoryczne dzielone są na dwie klasy w odniesieniu do postaci poszukiwanego rozwiązania: klasa problemów *decyzyjnych* i klasa problemów *przeszukiwania*
- W *problemach decyzyjnych* poszukiwanym rozwiązaniem jest odpowiedź „tak” lub „nie” na odpowiednio sformułowane pytanie dotyczące instancji problemu
- W *problemach przeszukiwania* poszukuje się konkretnego rozwiązania – pewnego obiektu czy wartości – spełniającego sformułowane warunki, lub odpowiedzi „nie”, jeśli takie rozwiązanie nie istnieje dla danej instancji
- Większość problemów kombinatorycznych może zostać sformułowana zarówno w postaci decyzyjnej, jak i przeszukiwania

## Problemy kombinatoryczne – przykład

- Ścieżka Hamiltona w grafie jest zdefiniowana jako ścieżka odwiedzająca każdy wierzchołek grafu dokładnie raz.
  Jest to innymi słowy taka permutacja wszystkich wierzchołków, dla której istnieje krawędź lub odpowiednio skierowany łuk w grafie pomiędzy wszystkimi sąsiednimi parami wierzchołków z permutacji



graf nieskierowany



graf skierowany



## Problemy kombinatoryczne – przykład

- Sformułowanie problemu poszukiwania ścieżki Hamiltona w grafie skierowanym – wersja decyzyjna
  Instancja: graf skierowany $G=(V, A)$.
  Pytanie: czy istnieje ścieżka Hamiltona w grafie $G$?

- Sformułowanie problemu poszukiwania ścieżki Hamiltona w grafie skierowanym – wersja przeszukiwania
  Instancja: graf skierowany $G=(V, A)$.
  Rozwiązanie: ścieżka Hamiltona w grafie $G$.

- Wersja decyzyjna problemu jest nie trudniejsza niż jego wersja przeszukiwania

## Problemy optymalizacyjne

- *Problem optymalizacyjny* jest szczególną odmianą problemu przeszukiwania. W problemie takim rozwiązaniem jest pewien obiekt o optymalnej wartości funkcji celu (funkcji kryterialnej) zdefiniowanej w problemie
- Rozwiązanie problemu optymalizacyjnego jest *dopuszczalne*, jeśli spełnia wszystkie warunki jak w sformułowaniu problemu z pominięciem warunku na optymalną wartość funkcji celu. Rozwiązanie o najlepszej wartości funkcji celu spośród dopuszczalnych jest *optymalne*
- Funkcję celu można *maksymalizować* (gdy poszukiwane jest rozwiązanie o najwyższej wartości funkcji) lub *minimalizować* (gdy wartość funkcji ma być jak najniższa)



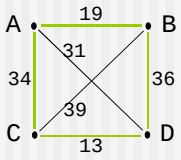
## Problemy optymalizacyjne – przykład

- W *problemie komiwojażera*, dla podanego zbioru miast i odległości między wszystkimi parami miast, poszukiwana jest trasa zamknięta o minimalnej sumarycznej długości odwiedzająca wszystkie miasta, każde dokładnie raz
- Funkcją celu jest tu suma odległości pomiędzy parami sąsiednich miast na trasie, jest ona minimalizowana
- Gdy macierz odległości pomiędzy miastami jest kompletna, rozwiązanie dopuszczalne w tym problemie jest łatwo osiągalne i jest nim dowolna permutacja miast
- Znalezienie rozwiązania optymalnego dla problemu komiwojażera jest obliczeniowo trudne

## Problemy optymalizacyjne – przykład

- W teorii grafów można problem komiwojażera przedstawić jako poszukiwanie cyklu Hamiltona o minimalnym koszcie w grafie pełnym nieskierowanym (gdy macierz odległości jest kompletna i symetryczna). Koszty to odległości pomiędzy miastami i są przypisane krawędziom grafu. Wartości kosztów krawędzi są w pewien sposób ograniczone (muszą być nieujemne i spełniać tzw. nierówność trójkąta)

| | A | B | C | D |
|---|---|---|---|---|
| A | 0 | 19 | 34 | 31 |
| B | 19 | 0 | 39 | 36 |
| C | 34 | 39 | 0 | 13 |
| D | 31 | 36 | 13 | 0 |



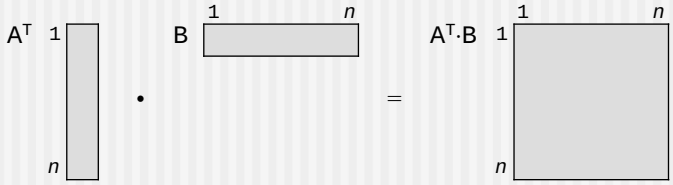



## Złożoność obliczeniowa algorytmów

- Czas obliczeń algorytmu rozwiązującego problem kombinatoryczny – w ogólności – rośnie wraz ze wzrostem rozmiaru instancji problemu
- Czas obliczeń dla większej instancji może okazać się krótszy niż dla mniejszej, lecz „trudniejszej”
- Funkcja czasu algorytmu utożsamiana jest z liczbą jego elementarnych kroków. Zmienną tej funkcji jest rozmiar instancji problemu $n$. Jeśli liczba kroków może zostać opisana (ograniczona od góry) funkcją wielomianową zmiennej $n$, mówimy, że algorytm jest *wielomianowy* (w sensie czasu). W przeciwnym przypadku przyjęło się nazywać algorytm *wykładniczym*

# Złożoność obliczeniowa algorytmów

- Przykładowo, algorytm mnożenia dwóch $n$-elementowych wektorów (jednowymiarowych macierzy) $A^T$ i B realizowany jest w $n^2$ krokach. Jest to algorytm wielomianowy o złożoności czasowej $O(n^2)$

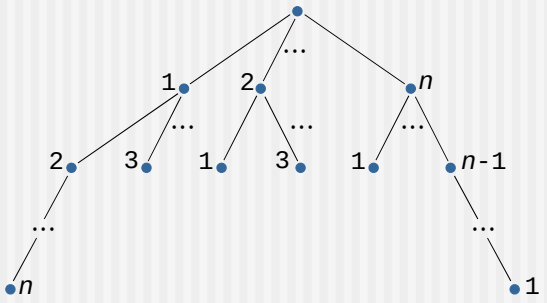



- W zapisie funkcji czasu pomija się mniejsze składniki wielomianu oraz stałe: $2n^3+n^2+3n \rightarrow O(n^3)$ , $\frac{1}{2}n^2 \rightarrow O(n^2)$



# Złożoność obliczeniowa algorytmów

- Przykładem algorytmu wykładniczego jest algorytm generujący wszystkie permutacje w zbiorze $n$-elementowym o złożoności czasowej $O(n!)$



## Złożoność obliczeniowa problemów

- Problemy kombinatoryczne dzielimy pod względem ich złożoności obliczeniowej na – w uproszczeniu – problemy łatwe i trudne obliczeniowo
- Problem jest *łatwy obliczeniowo*, jeśli udowodniono, że znalezienie jego dokładnego rozwiązania (dla dowolnej instancji problemu) możliwe jest w czasie ograniczonym od góry wielomianem zależnym od rozmiaru instancji problemu
- Problem jest *trudny obliczeniowo*, jeśli – w uproszczeniu – przeprowadzono dowód jego przynależności do klasy problemów NP-zupełnych (problemy decyzyjne) lub NP-trudnych (problemy przeszukiwania). Dla takich problemów nie są znane dokładne algorytmy wielomianowe

*Zakłada się opanowanie zagadnień (w sensie wiedzy i umiejętności) zrealizowanych w ramach przedmiotu „Algorytmy i struktury danych”*



## Złożoność obliczeniowa problemów

- Przykładem problemu trudnego obliczeniowo jest problem poszukiwania ścieżki Hamiltona w grafie. Pozornie podobny do niego problem poszukiwania ścieżki Eulera jest już obliczeniowo łatwy
- Wśród problemów łatwych obliczeniowo można wymienić: poszukiwanie najkrótszej ścieżki w grafie, problem przydziału, szeregowanie zadań podzielnych (pewne warianty)
- Problemami trudnymi obliczeniowo są między innymi: poszukiwanie najdłuższej ścieżki w grafie, problem komiwojażera, szeregowanie zadań niepodzielnych (pewne warianty)

## Złożoność obliczeniowa problemów

- W biologii obliczeniowej większość problemów jest trudna obliczeniowo. Zdarza się, że najprostsze teoretyczne modele odzwierciedlające podstawowy wariant problemu biologicznego są rozwiązywalne w czasie wielomianowym. Natomiast uwzględnienie naturalnej złożoności problemu, a zwłaszcza uwzględnienie obecności błędów eksperymentalnych w instancji, skutkuje zazwyczaj trudnym obliczeniowo problemem kombinatorycznym
- Przykładowo, klasyczny problem sekwencjonowania przez hybrydyzację (mając zbiór oligonukleotydów o stałej długości, odtwórz badaną sekwencję) jest prosty obliczeniowo jedynie w przypadku braku jakichkolwiek błędów czy braków w zbiorze. Jest rozwiązywany przez transformację do problemu poszukiwania ścieżki Eulera w grafie skierowanym



## Algorytmy dokładne

- Algorytmy dokładne służą rozwiązywaniu problemów w sposób dokładny (czyli *nieheurystyczny*). W przypadku problemów optymalizacyjnych oznacza to gwarancję wygenerowania rozwiązania optymalnego
- Jak dotąd nie udowodniono, że problem trudny obliczeniowo można (lub nie) rozwiązać w sposób dokładny algorytmem wielomianowym. W praktyce takie problemy, a także problemy otwarte, rozwiązuje się algorytmami wykładniczymi (choć tylko stosunkowo małe instancje) lub w sposób przybliżony (niedokładny) wielomianowymi *heurystykami*
- Algorytmy wykładnicze często są konstruowane wg schematu *branch-and-bound*, czasem *branch-and-cut*, a w najprostszej wersji jako *przeszukiwanie z nawrotami*

## Heurystyki

- W przypadku problemów trudnych obliczeniowo czas oczekiwania na dokładne rozwiązanie dla większych instancji problemu staje się zbyt duży, nawet dla najlepszych algorytmów wykładniczych
- Użytkownik często w takiej sytuacji jest skłonny zrezygnować z rozwiązania dokładnego na rzecz rozwiązania przybliżonego, za to uzyskanego w akceptowalnym czasie
- *Heurystyką* nazywamy algorytm (metodę) zwracający rozwiązanie przybliżone. Zazwyczaj oczekuje się, że algorytm taki ma złożoność wielomianową i działa szybko w praktyce
- Zaawansowanie heurystyki koreluje z jej złożonością czasową i jakością zwracanych rozwiązań. Przykłady heurystyk: losowa, zachłanna, lokalnego przeszukiwania, metaheurystyka



## Heurystyki

- W przypadku problemów optymalizacyjnych metoda heurystyczna ma na celu zwrócenie rozwiązania dopuszczalnego o wartości funkcji celu jak najbliższej optymalnej
- W problemie komiwojażera z kompletną macierzą odległości wygenerowanie rozwiązania dopuszczalnego jest proste, co nie jest regułą dla dowolnego problemu optymalizacyjnego. Metoda heurystyczna może nie zwrócić żadnego rozwiązania, jeśli nie znajdzie dopuszczalnego
- Jakość heurystyki można ocenić m.in. na podstawie różnicy w wartości funkcji celu osiągniętej przez nią i optymalnej. Wartość optymalną można dla niektórych problemów/instancji lepiej lub gorzej oszacować, ewentualnie dla mniejszych instancji można ją uzyskać istniejącym algorytmem dokładnym

## Literatura


- Michael S. Waterman, *Introduction to Computational Biology. Maps, Sequences, and Genomes*, Chapman & Hall, London, 1995
- João Setubal, João Meidanis, *Introduction to Computational Molecular Biology*, PWS Publishing Company, Boston, MA, 1997
- Pavel A. Pevzner, *Computational Molecular Biology: an Algorithmic Approach*, MIT Press, Cambridge, MA, 2000
- Dan Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press, Cambridge, MA, 1997
- Marta Kasprzak, *Wybrane algorytmy i modele grafowe w bioinformatyce*, Wydawnictwo Politechniki Poznańskiej, Poznań, 2013




## Literatura – cd.


- Claude Berge, *Graphs and Hypergraphs*, North-Holland Publishing Company, London, 1973
- Eugene L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976
- Christos H. Papadimitriou, Kenneth Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, Englewood Cliffs, 1982
- Michael R. Garey, David S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W.H. Freeman & Co., San Francisco, 1979
- Jacek Błażewicz, *Złożoność obliczeniowa problemów kombinatorycznych*, WNT, Warszawa, 1988