Joanna Janaszek
122516

Sprawozdanie
Bioinformatyka

1. Opis zadania

Projekt dotyczył implementacji izotermicznego sekwencjonowania przez hybrydyzację z błędami negatywnymi. W zaimplementowanych algorytmach sekwencja DNA zbudowana jest z oligonukleotydów z uwzględnieniem ich liczby powtórzeń, która u mnie wynosi: 0,1,{2,3},{4,5} lub wiele. Danymi wejściowymi są procent błędów, procent powtórzeń, zakres temperatur i sekwencja DNA.

2. Model grafu

Oligonukleotydy są przechowywane w następującej klasie:

```
class Oligonucleotide:
    def __init__(self, i, series, size, min, max, first):
        self.id = i
        self.series = series
        self.size = size
        self.min = min
        self.max = max
        self.max2 = 0
        self.times_used = 0
        self.first = first
        self.complementary = False
```

Powiązanie między oligonukleotydami opisane jest klasą Graph. W zmiennej start znajduje się oligonukleotyd z spectrum a w tablicy edges, łuki do wszystkich oligonukleotydów do których można przejść z tego wierzchołka. Oligonukleotydy te są przechowywane w klasie Edge.

```
class Graph:
    def __init__(self, oli):
        self.start = oli
        self.edges = []
```

Klasa Edge przechowuje oligonukleotyd do którego można przejść z startowego (w polu end), możliwe przesunięcia między nimi i wiadomość czy dany łuk można jeszcze wykorzystać (jest dostępny).

```
class Edge:
    def __init__(self, end=None, pheromone=None):
        self.end = None if end is None else end
        self.move = []
        self.rest = []
        self.available = False
        self.pheromone = 0.1 if pheromone is None else pheromone
```

Generowanie spektrum:

Spektrum generowane jest z podanej na wejściu sekwencji DNA. Dzielona jest ona na oligonukleotydy z uwzględnieniem podanych na wejściu temperatur. Następnie sprawdzane jest czy w wygenerowanych oligonukleotydach liczba powtórzeń zgadza się z procentem powtórzeń podanym na wejściu. Jeśli tak, w kolejnym kroku, z spektrum zostają usunięte losowo oligonukleotydy z uwzględnieniem procentu błędów z danych wejściowych. Kolejno oligonukleotydom zostaje przypisana ich wartość minimalnego i maksymalnego użycia. Następnie zostaje dodany do spektrum dla każdego oligonukleotydu jego odpowiednik komplementarny, jeśli jeszcze nie znajduje się w spektrum. Zostaje mu przypisana taka sama wartość minimalnego i maksymalnego użycia. Natomiast jeśli oligonukleotyd komplementarny znajdował się już w spektrum to ich wartości użycia zostają porównane. W przypadku gdy są one różne, oligonukleotydowi o niższych wartościach zostają przypisane wartości jego odpowiednika komplementarnego.

Generowanie grafu:

Obiekty klasy *Graph* przechowywane są w tablicy *graph*, gdzie w zmiennej *start* znajdują się oligonukleotydy wygenerowane w spektrum. Każdy obiekt klasy *Graph* w tablicy *edges* zawiera łuki do oligonuklotydów z którymi może się połączyć. Na łukach znajduje się informacja o możliwych przesunięciach między nimi.

3. Algorytm dokładny

W poszukiwaniu poprawnych sekwencji, stworzony graf jest rekurencyjnie przechodzony w głąb. Algorytm rozpoczyna się od wierzchołka z pierwszym oligonukleotydem. Z jego sąsiadów wybierany jest po kolei kolejny oligonukleotyd do którego można przejść w grafie. W przypadku gdy wybrany oligonukleotyd zawiera się w oligonukleotydzie z wierzchołka obecnie odwiedzanego, zwiększona zostaje jego liczba użycia oraz jego komplementarnego odpowiednika. W przeciwnym przypadku, jeśli po dodaniu go do sekwencji jej długość nie przekracza długości sekwencji wejściowej, to przechodzimy do wierzchołka z tym oligonukleotydem (ustawiany jest on jako aktualnie odwiedzany) i przechodzimy do następnej iteracji pętli. Do sekwencji dodawany jest odcinek z poprzednio odwiedzonego oligonukleotydu o długości równej przesunięciu między nim a obecnym, która to informacja zapisana jest na łuku ich łączącym. W przypadku gdy długość poprzedniego oligonukleotydu jest równa przesunięciu, na koniec sekwencji zostaje dodane ‘X’. Oznacza to, że potencjalnie może tu brakować fragmentu DNA. Po wykorzystaniu oligonukleotyda zostaje zwiększona jego ilość użycia, jak i jego komplementarnego odpowiednika. Następnie sprawdzane jest czy suma długości obecnej sekwencji z długością oligonukleotydu w wierzchołku obecnie odwiedzanym nie jest większa od długości sekwencji wejściowej. Jeśli jest mniejsza to sprawdzane jest czy wygenerowana do tej pory sekwencja może być poprawna i można przechodzić do następnego wierzchołka, czy należy się wrócić. Podczas powrotu do wcześniejszego wierzchołka z sekwencji zostają usunięte dodane ciągi oraz wartość

użycia tych oligonukleotydów zostaje zmniejszona. Jeżeli długość jest większa to do sekwencji dodawana jest końcówka z oligonukleotydu w poprzednim wierzchołku. Następnie sprawdzane jest czy długość sekwencji z dodaniem oligonukleotyda w obecnym wierzchołku jest równa długości sekwencji wejściowej. Jeśli tak użycie tego oligonukleotyda jak i jego odpowiednika komplementarnego zostaje zwiększona i utworzona sekwencja zostaje poddana sprawdzeniu czy jest poprawna. Podczas sprawdzania poprawności najpierw zostaje ustalone czy wszystkie oligonukleotydy zostały użyte odpowiednią ilość razy, pamiętając, że maksymalna liczba użycia dopuszcza błąd o jeden przedział w górę, czyli np. dla oligonukleotydu z powtórzeniami z przedziału {2,3} dopuszcza się liczbę powtórzeń w przedziale {4,5}. Następnie generowany jest odpowiednik komplementarny dla utworzonej sekwencji, i sprawdzane jest czy wszystkie oligonukleotydy z spektrum się w nim zawierają z uwzględnieniem ich liczby powtórzeń. Po sprawdzeniu poprawności i wypisaniu wyniku na konsolę następuje cofnięcie się rekurencji i graf jest dalej przeszukiwany by znaleźć wszystkie możliwe poprawne połączenia oligonukleotydów.

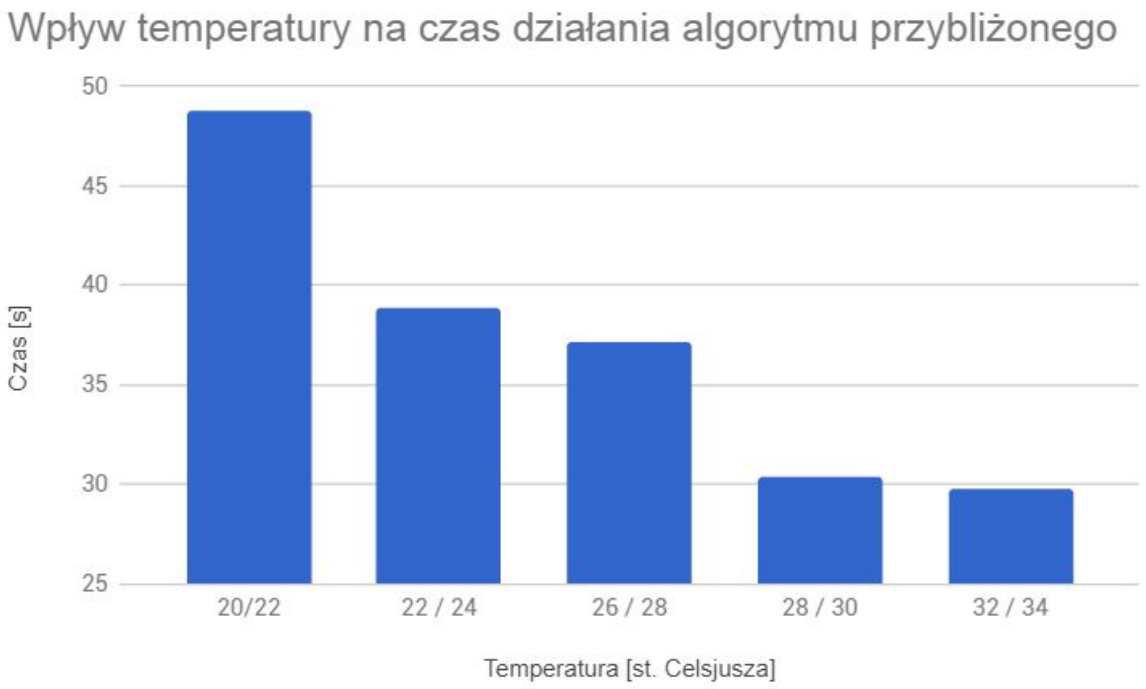
4. Algorytm przybliżony

Algorytm przybliżony jest wzorowany na algorytmie mrówkowym. Przebieg algorytmu rozpoczyna się w wierzchołku z pierwszym oligonukleotydem. Mrówki przechodzą przez graf tak długo aż nie zostanie spełniony warunek minimalnego użycia każdego z oligonukleotydów. Za pierwszym razem wybór ścieżki przez mrówki jest dokonywany losowo. Przechodząc do kolejnego wierzchołka mrówki zostawiają na łukach feromony w ilości zależnej od przesunięcia między dwoma oligonukleotydami w tych wierzchołkach (im jest ono mniejsze tym liczba feromonów jest większa). Zanikanie feromonów następuje w momencie gdy wszystkie mrówki wygenerują swoje ścieżki. Kolejne razy podczas wyboru ścieżek przez mrówki wzrasta prawdopodobieństwo wyboru łuku który nie został użyty odpowiednią ilość razy oraz łuku z większą liczbą pozostawionych na nim feromonów. Po wybraniu ścieżki zostaje zwiększona ilość użycia oligonukleotydu w tym wierzchołku, komplementarnego do niego oraz wszystkich oligonukleotydów (i ich komplementarnych odpowiedników) które się w nim zawierają. Do sekwencji dodawany jest odcinek z odwiedzanego oligonukleotydu o długości równej przesunięciu między nim a kolejnym wybranym, która to informacja zapisana jest na łuku ich łączącym. Liczba feromonów, którą mrówka pozostawi na łuku, zostaje zapamiętana. Następnie sprawdzane jest czy wszystkie oligonukleotydy zostały użyte odpowiednią ilość razy i czy można przejść do sprawdzania poprawności wygenerowanej sekwencji i dodania zapamiętanej liczby feromonów do odpowiednich łuków. W przypadku gdy nie wszystkie oligonukleotydy zostały wykorzystane następuje dalsze wybieranie kolejnych wierzchołków grafu.

5. Przeprowadzone pomiary


Wpływ temperatury na czas działania algorytmu dokładnego
150
100
50
0
Czas [s]
20 / 22
22 / 24
26 / 28
28 / 30
32 / 34
Temperatura [st. Celsjusza]


Wpływ długości sekwencji na czas działania algorytmu przybliżonego
50
40
30
20
10
0
Czas [s]
18
19
20
21
23
24
Długość sekwencji



6. Wnioski

Zaimplementowany algorytm dokładny ma złożoność czasową wykładniczą, co można zauważyć z wykresu przedstawiającego wpływ długości sekwencji na czas działania algorytmu. Wynika to z rozszerzania się grafu, który należy przeszukać, wraz ze wzrostem długości sekwencji.
Kolejny wykres przedstawiający wpływ ilości błędów na czas działania algorytmu pokazuje że im więcej błędów tym algorytm działa szybciej. Jest to spowodowane zmniejszoną liczbą wierzchołków w grafie i powtórzeń ze względu na dużą liczbę błędów.
Z wykresu przedstawiającego wpływ temperatury na działanie algorytmu widać, że wyższa temperatura znacznie zmniejsza czas działania algorytmu. Wynika to z faktu że oligonukleotydy są dłuższe, zatem większą powierzchnią nakładają się na kolejne oligonukleotydy, co przyspiesza proces ich dopasowania.
Dla algorytmu przybliżonego można zauważyć te same tendencje czasowe, jednak nie są one już wykładnicze a wielomianowe.
Dokładnego wpływu temperatury i ilości błędów na czas znalezienia rozwiązania nie można dokładnie zaobserwować, ze względu na niską wydajność zaimplementowanych algorytmów. Pomiary nie były możliwe dla dużej długości sekwencji DNA wejściowej.