Sprawozdanie z projektu zaliczeniowego
Z Optymalizacji Kombinatorycznej

Przygotowała:
Julia Lemańska,
nr indeksu 155649

Opis algorytmu

1. **Utworzenie grafu za pomocą macierzy sąsiedztwa.** W pętli wypełniana jest macierz zerami. Kolumny i wiersze macierzy są poindeksowane równoważnie do indeksów danych oligonukleotydów. Każdy oligonukleotyd jest porównywany do każdego innego. Sprawdzana jest liczba znaków nakładających się na siebie (zaczynając od wagi 3 do wagi 1) a następnie odpowiednia waga zostaje przypisana do odpowiedniej komórki macierzy.
2. **Rozpoczęcie szukania ścieżki w grafie.** Znany jest pierwszy oligonukleotyd, więc najpierw wyszukiwany jest jego indeks w wektorze spektrum. Jeśli początek ścieżki nie został znaleziony to zwracane jest puste rozwiązanie, ponieważ wystąpił błąd przy zapisywaniu początkowego wierzchołka.
   a. Jednak jeśli indeks zostanie znaleziony, to pierwszy oligonukleotyd zostaje dodany do rozwiązania oraz zostaje zwiększona o 1 liczba użytych wierzchołków.
   b. Odznaczenie flagi *found* jako *false* i wyszukiwanie w wierszu o indeksie wierzchołka poprzedniego (początkowego) wagi o wartości 1. Jeśli zostanie znaleziona, indeks kolumny, która posiada szukaną wartość wagi, odpowiada indeksowi następnego wierzchołka – oligonukleotydu. Zmienna *index*, która przechowuje aktualny indeks wierzchołka, ma teraz przypisany nowy indeks. Zwiększona o 1 zostaje liczba użytych wierzchołków oraz flaga *found* zostaje odznaczona jako *true*.
   c. W innym wypadku, jeśli w wierszu nie będzie wagi 1 i *found* pozostanie *false*, wyszukiwanie zostaje powtórzone z wagą o wartości 2.
3. **Szukanie rozwiązania**. W pętli *while* wyszukiwane jest pełne rozwiązanie problemu i przerwanie pętli nastanie tylko gdy liczba użytych wierzchołków przekroczy p% liczby oligonukleotydów obecnych w spektrum idealnym ($0.75 < p < 1$) oraz gdy długość rozwiązania będzie równa oryginalnej długości DNA. W tej części algorytmu używane są trzy flagi: *found*, *found2, found3*. Pierwsza flaga działa tak samo jak w punkcie 2. Druga flaga używana jest w przypadku, gdy aktualny wierzchołek nie posiada łuku ani o wadze 1, ani o wadze 2. Wtedy wyszukiwany jest łuk o wadze 3. Sam proces wyszukiwania jest identyczny jak w punkcie 2. Natomiast jeśli nie zostanie znaleziony łuk o żadnej z powyżej wymienionych wag, to aktualny wierzchołek jest przesuwany o 1 czyli zmienna *index* zwiększana jest o 1. Ten następny wierzchołek dodawany jest całkowicie do rozwiązania i od niego jest poszukiwane dalsze rozwiązanie. Ma to na celu zapobiec utknięciu algorytmu w ślepej ścieżce i nieskończonej pętli.
4. Po spełnieniu obydwu warunków pętli, następuje jej przerwanie i wtedy zostaje zwrócone gotowe rozwiązanie – ścieżka w grafie.

Testowanie parametru pokrycia spektrum idealnego

Parametr pokrycia spektrum idealnego jest progiem jaki należy przekroczyć aby znaleźć rozwiązanie. Próg ten jest procentem z liczby oligonukleotydów w spektrum idealnym. Czyli jeśli n=100; k=4; p=0.9 to tyle wierzchołków musi być wykorzystanych w rozwiązaniu: (n-k+1)*p = 87 (zaokrąglenie do liczby całkowitej). Do testowania tego parametru użyte zostaną 32 instancje o danych podanych poniżej:

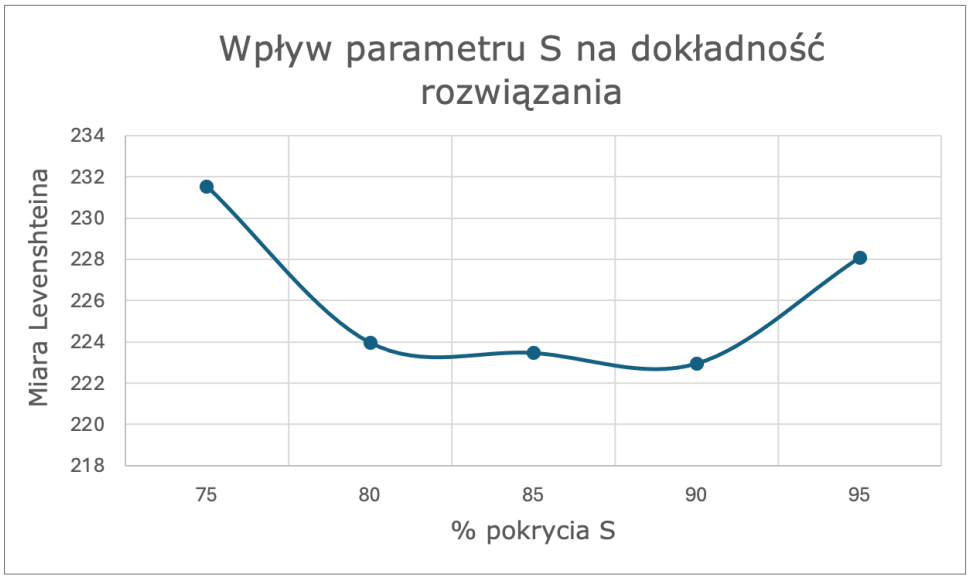
Instancje

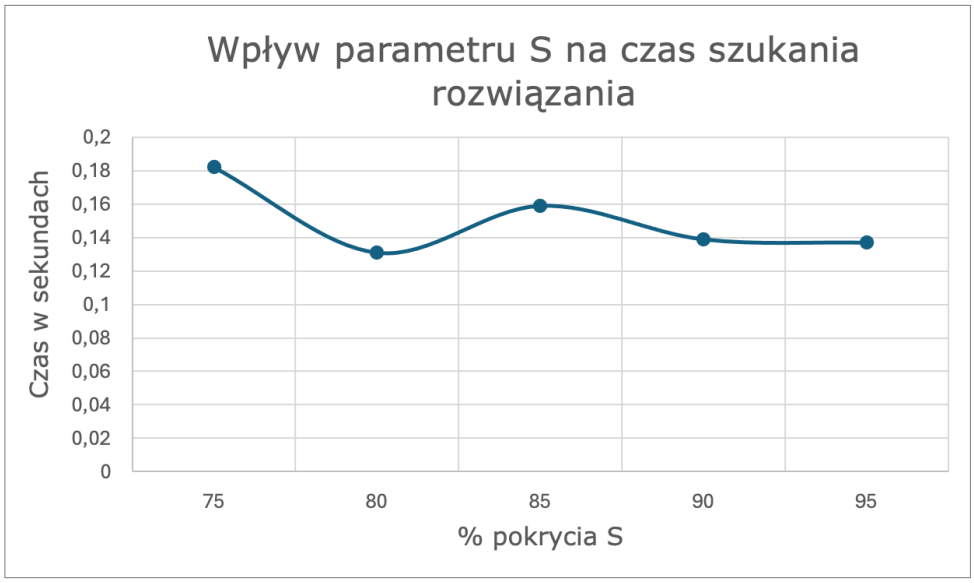
Długość DNA – n	k, liczba błędów negatywnych i pozytywnych
400	1. k=6; negatywne=4; pozytywne=4 2. k=6; negatywne=3; pozytywne=5 3. k=7; negatywne=6; pozytywne=5 4. k=7; negatywne=4; pozytywne=8
450	5. k=6; negatywne=9; pozytywne=9 6. k=6; negatywne=7; pozytywne=5 7. k=7; negatywne=10; pozytywne=11 8. k=7; negatywne=13; pozytywne=11
500	9. k=7; negatywne=0; pozytywne=0 10. k=7; negatywne=7; pozytywne=7 11. k=8; negatywne=2; pozytywne=4 12. k=8; negatywne=5; pozytywne=6
511	13. k=7; negatywne=8; pozytywne=7 14. k=7; negatywne=9; pozytywne=11 15. k=8; negatywne=12; pozytywne=10 16. k=8; negatywne=6; pozytywne=9
575	17. k=7; negatywne=0; pozytywne=0 18. k=8; negatywne=0; pozytywne=6 19. k=8; negatywne=7; pozytywne=0 20. k=8; negatywne=15; pozytywne=17
600	21. k=8; negatywne=9; pozytywne=6 22. k=8; negatywne=6; pozytywne=9 23. k=9; negatywne=5; pozytywne=4 24. k=9; negatywne=6; pozytywne=8
650	25. k=8; negatywne=0; pozytywne=8 26. k=8; negatywne=7; pozytywne=0 27. k=8; negatywne=10; pozytywne=11 28. k=9; negatywne=15; pozytywne=17
690	29. k=8; negatywne=7; pozytywne=9 30. k=8; negatywne=12; pozytywne=13 31. k=9; negatywne=14; pozytywne=18 32. k=9; negatywne=20; pozytywne=19

Wyniki testu parametru

Próg pokrycia S	Średnia miara Levenshteina[1]	Czas w sekundach
75%	231,5355	0,182
80%	223,9500	0,131
85%	223,4500	0,159
90%	222,9312	0,139
95%	228,0750	0,137

[1] Algorytm wzięty z https://github.com/guilhermeagostinelli/levenshtein/blob/master/levenshtein.cpp i jest on używany we wszystkich testach w tym sprawozdaniu.





Wnioski

Po przeprowadzonych testach można zauważyć, że miara Levenshteina nie jest proporcjonalna do stopnia pokrycia S. Najbardziej wydajna okazała się wartość 90% a po niej 85% pokrycia S. Można przypuszczać, że próg 95% jest na tyle wysoki przy algorytmie losowym, który działa dosyć zachłannie, że ta wymagana dokładność bardziej psuje rozwiązanie niż je poprawia. Jeśli instancja ma dużo błędów, to może się okazać, że algorytm, pozostając w pętli, żeby przekroczyć wysoki próg odwiedzonych wierzchołków, może zacząć dodawać dużo błędów pozytywnych albo fragmentów powtarzających się na przykład AAAA, ponieważ wtedy będzie ich używać tak długo, aż nie osiągnie tego progu. Także zbyt niski próg może przepuścić zbyt dużo oligonukleotydów, powodując, że pewna część sekwencji może nawet nie być bliska oryginalnej, bo na przykład, nie został użyty wierzchołek, który ma parę powtórzeń w sekwencji. W związku z tym, z uwagi na prymitywność algorytmu, najbardziej efektywny próg pokrycia liczby spektrum idealnego jest w przedziale 85% - 90%.

Czas algorytmu dla każdej wartości badanego parametru jest do siebie zbliżony, szczególnie w przedziale 80% - 95%. Najdłuższy czas, widoczny przy progu 75%, może wynikać z pierwszeństwa testowania tej wartości, gdy komputer pierwszy raz zaczynał liczyć wysokie instancje, szczególnie te powyżej 500 nukleotydów. Jednakże, algorytm

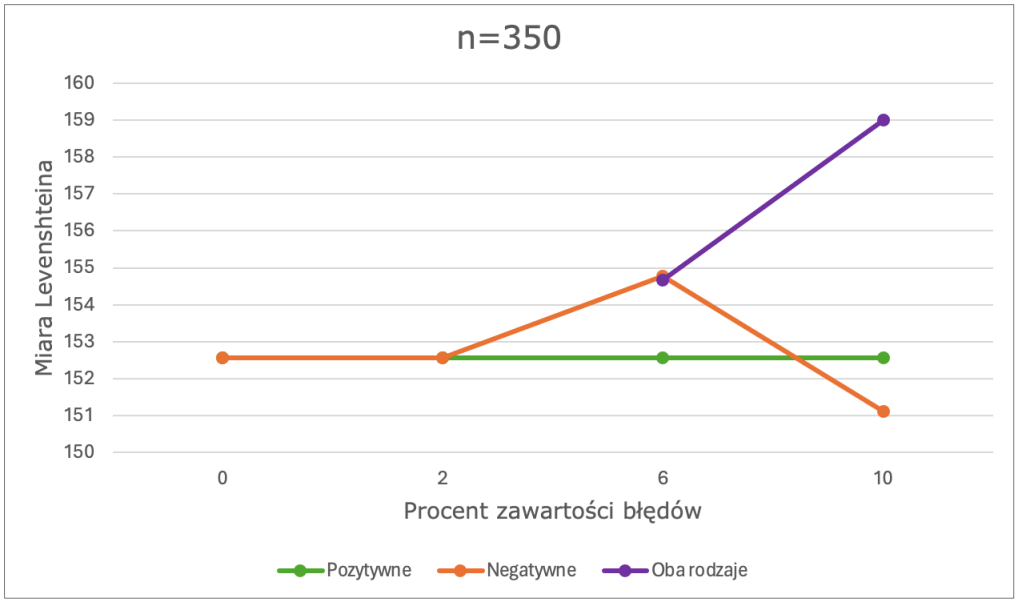
losowy jest na tyle szybki (i niedokładny), że pomiary czasów są zbliżone i nie odgrywają większej roli w wynikach.

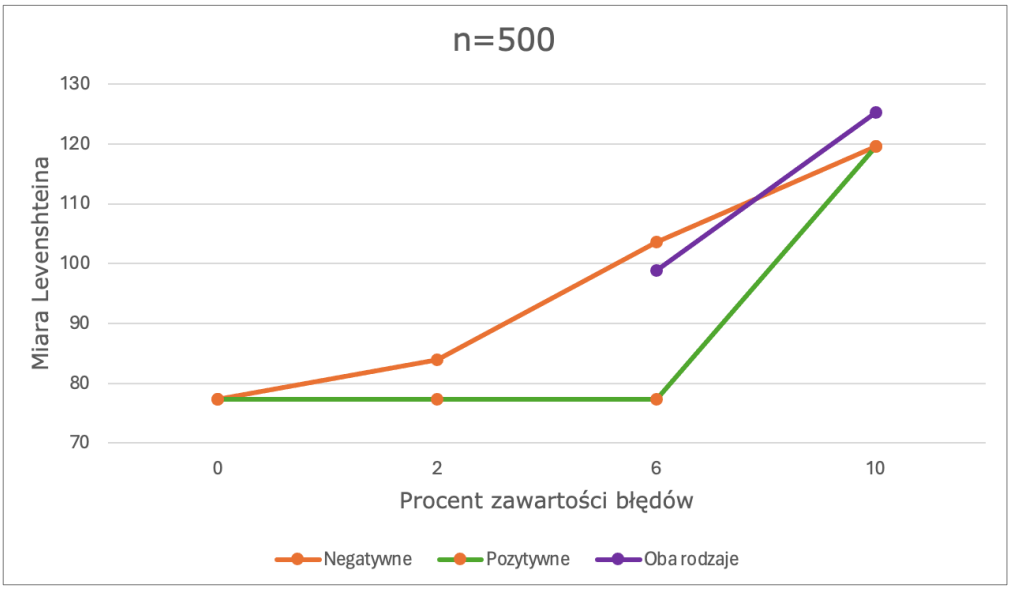
Testowanie parametrów instancji problemu

Wpływ błędów negatywnych oraz pozytywnych

Do zbadania wpływu błędów zostały użyte dwa rozmiary instancji: n=500 i k=9 oraz n=350 i k=7. Dwa rozmiary mają pomóc w określeniu ogólnym wpływu błędów, niezależnie od wielkości instancji. Próg pokrycia = 85% S.

	Miara Levenshteina	
Błędy	**n=500 i k=9**	**n=350 i k=7**
Negatywne = 0% Pozytywne = 0%	77,33	152,56
Negatywne = 2%	83,89	152,56
Negatywne = 6%	103,56	154,78
Negatywne = 10%	119,56	151,11
Pozytywne = 2%	77,33	152,56
Pozytywne = 6%	77,33	152,56
Pozytywne = 10%	119,56	152,56
Negatywne = 3% Pozytywne = 3%	98,89	154,67
Negatywne = 5% Pozytywne = 5%	125,22	159





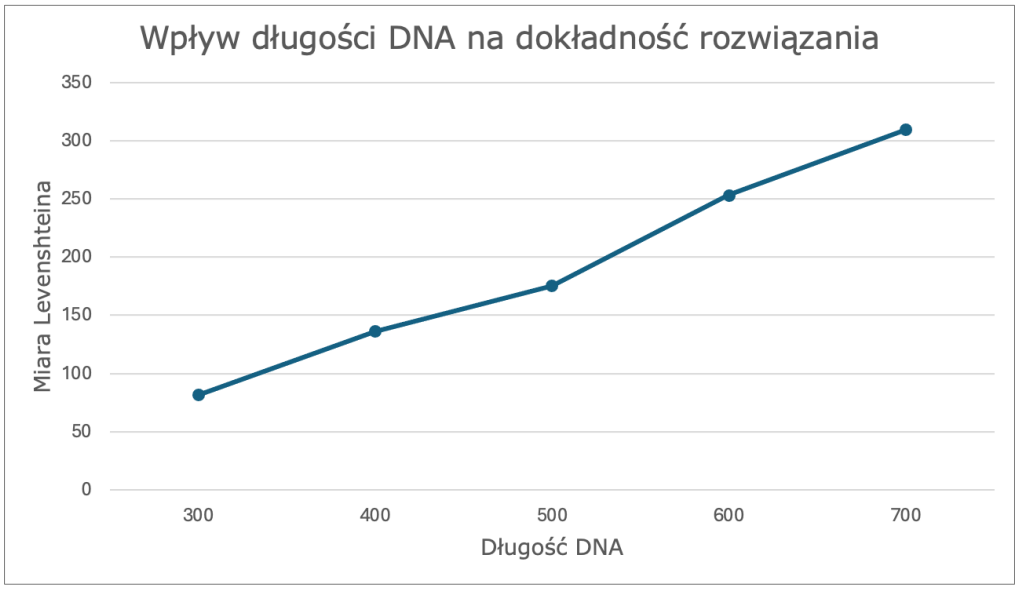
Wnioski

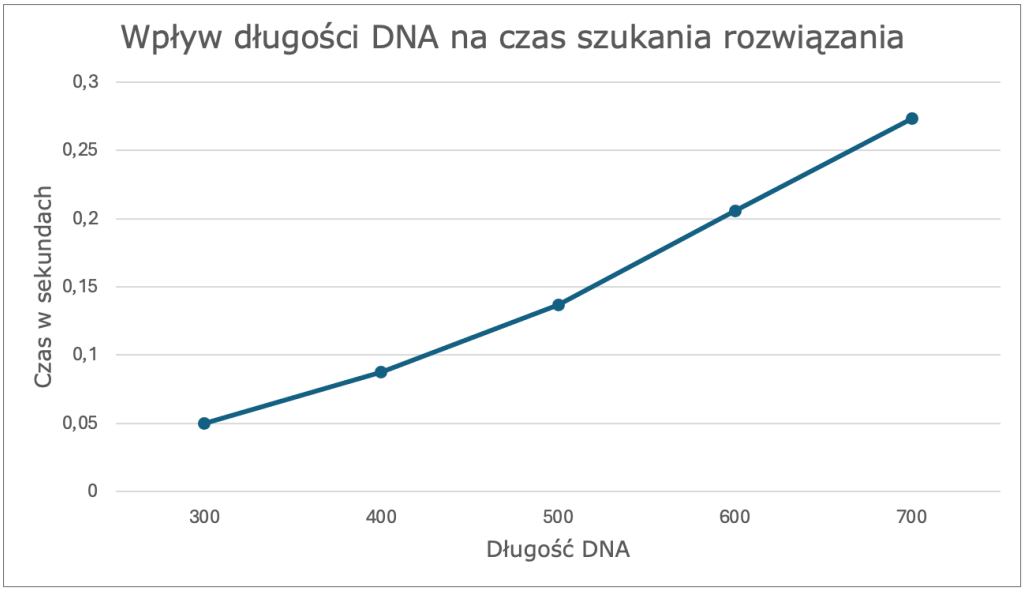
Wyniki testów okazały się dość zaskakujące – rozwiązania problemu przy dłuższych sekwencjach przy n=500 są bardziej dokładnie niż przy n=350. Może to być spowodowane obciążeniem próby badawczej. Niektóre sekwencje wygenerowane przy n=350 mogą mieć więcej powtórzeń, co powoduje, że niezależnie od błędów, trudno jest odtworzyć sekwencję. Przy n=500 zależność jest bardziej zbliżona do liniowej, co więcej, błędy negatywne mają większy wpływ na niedokładność rozwiązania niż pozytywne. Ta właściwość jest dość jasna, przy wiedzy, że dla jakiejkolwiek ilości błędów, spektrum jest zbiorem, nie multizbiorem i nie zawiera powtórzeń oligonukletydów. W związku z tym, jeśli nawet niewielka ilość błędów negatywnych zostanie wprowadzona, jest prawdopodobieństwo usunięcia istotnie powtarzającego się oligonukleotydu, które widocznie wpłynie na dokładność rozwiązania. W obu przypadkach testów, (nie)dokładność błędów negatywnych i pozytywnych naraz jest proporcjonalna do % zawartości błędów.

Wpływ długości parametru *n*

W badaniu wpływu długości n, użyte zostały instancje o stałych parametrach: k=8; błędy negatywne=0; błędy pozytywne=0. Zmieniany zostaje tylko parametr n. Każda instancja testowana jest 20 razy i poniższe wartości miary Levenshteina i czasu są uśrednione z 20 testów. Próg pokrycia = 85% S.

n	300	400	500	600	700
Miara Levenshteina	81,68	136,16	175	253,16	309,74
Czas	0,0500	0,0873	0,1366	0,2058	0,2732





Wnioski

Zarówno czas jak i dokładność rosną wprost proporcjonalnie do wielkości instancji. To pokazuje, że algorytm działa także wprost proporcjonalnie do złożoności instancji w zakresie wielkości. Jest to osiągalne m.i. dzięki prostej budowie algorytmu i struktury grafu. Mimo wysokiej losowości i prymitywności algorytmu, program działa szybko i umiarkowanie dokładnie i może być przydatny gdy potrzebne jest szybkie oszacowanie możliwej ścieżki.