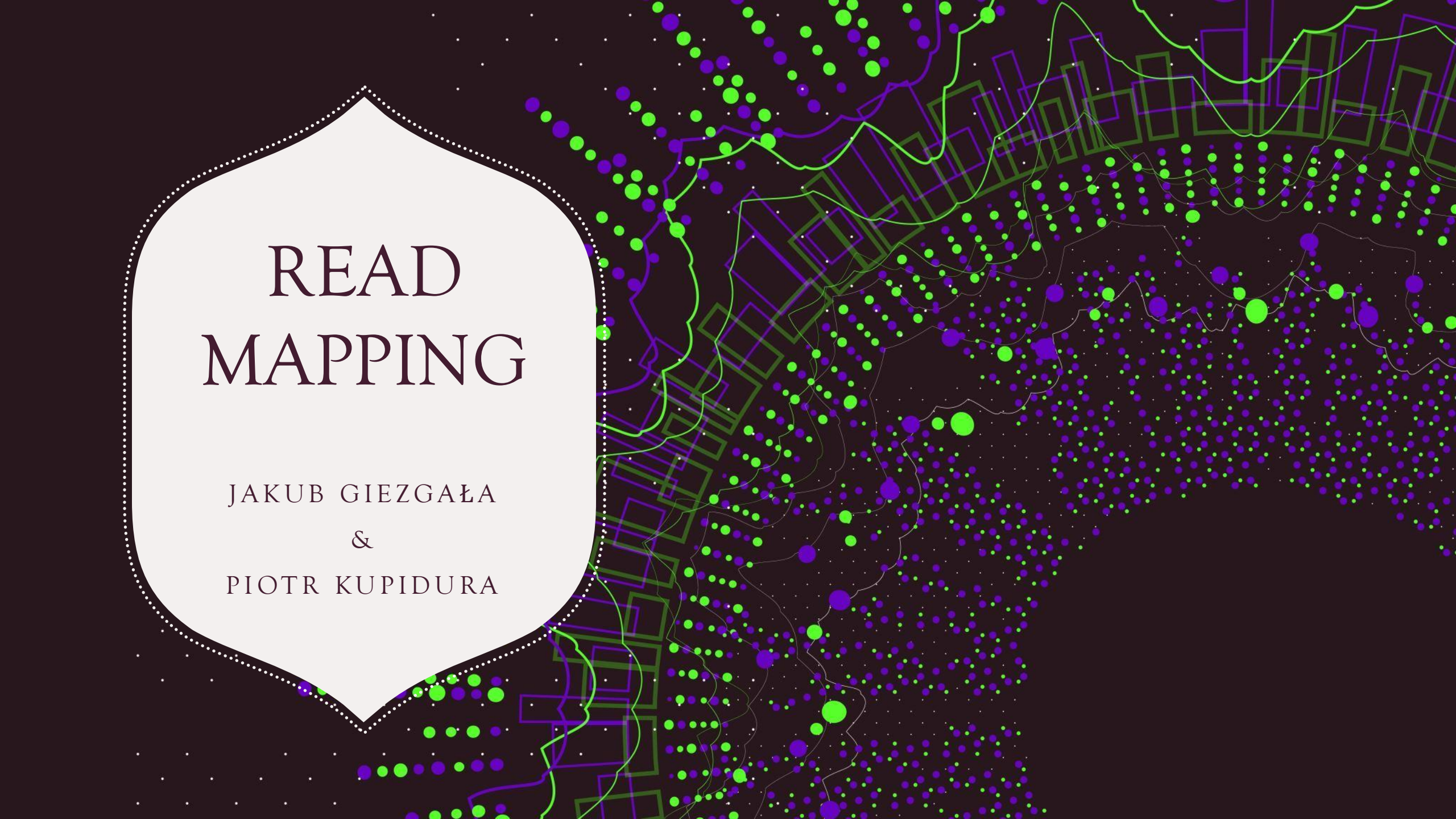
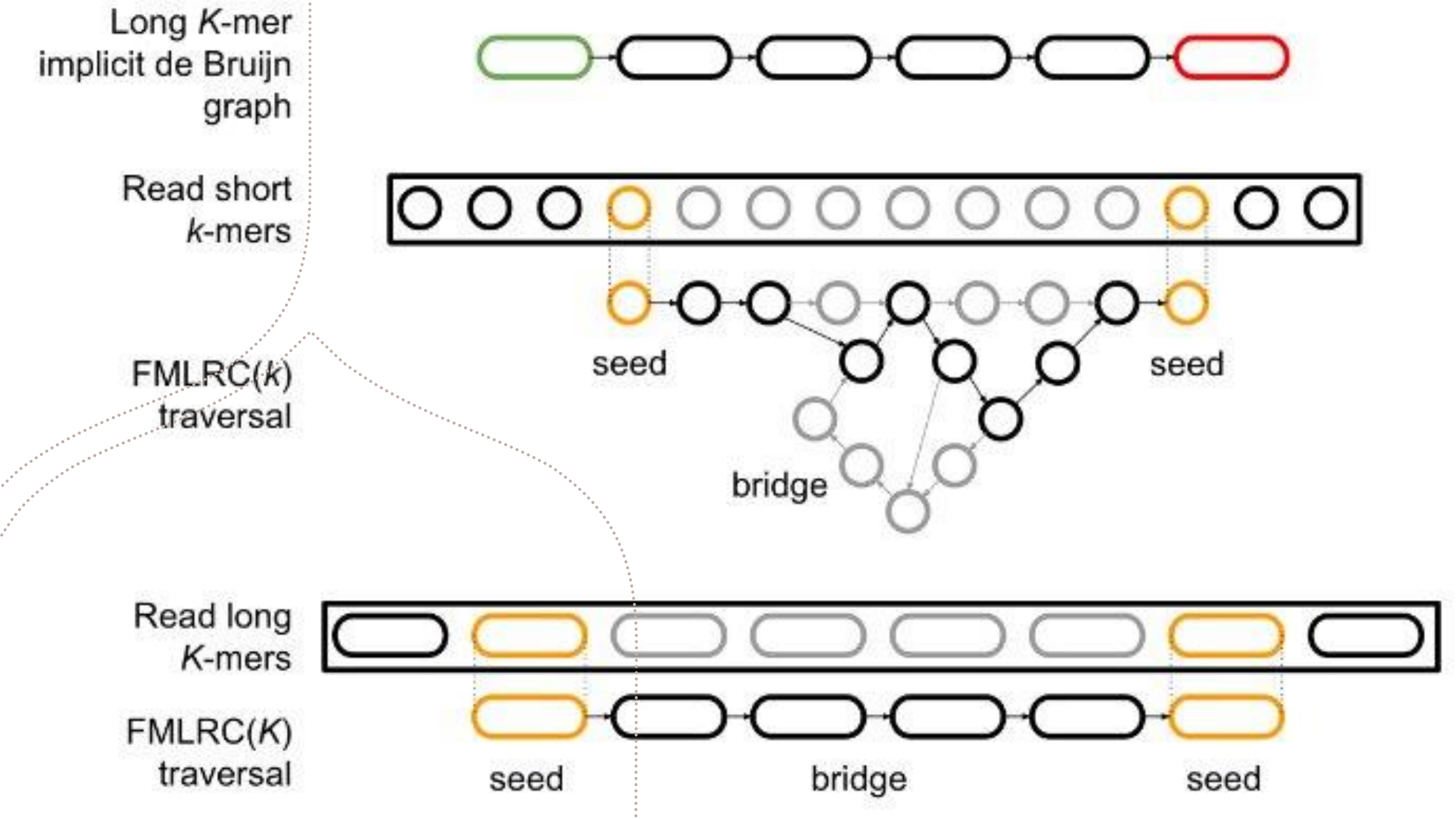


READ MAPPING

JAKUB GIEZGAŁA
&
PIOTR KUPIDURA

Opis Rozwiązania

Rozwiązanie opiera się na wykorzystaniu indeksu FM (Ferragina-Manzini), bazującego na Transformacie Burrowsa-Wheelera (BWT), co pozwala na efektywne przeszukiwanie tekstów w skompresowanej postaci. W celu przezwyciężenia wysokiego poziomu szumu w danych wejściowych, zastosowano strategię *seed-and-extend* (zaszczep i rozszerz), wykorzystującą Dokładne Dopasowania Maksymalne (MEMs – Maximal Exact Matches) jako kotwice. Implementacja wykorzystuje biblioteki NumPy w celu optymalizacji operacji numerycznych.


Long K-mer implicit de Bruijn graph
Read short k-mers
FMLRC(k) traversal
seed
seed
bridge
Read long K-mers
FMLRC(K) traversal
seed
bridge
seed

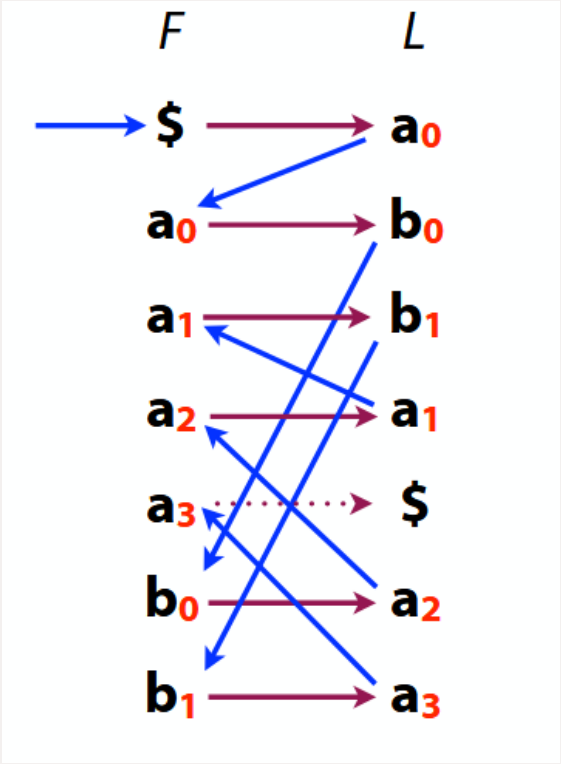
Konstrukcja Tablicy Sufiksowej

- **Algorytm:** Karkkainen-Sanders (DC3).
- **Działanie:**
  - Rekurencyjne sortowanie sufiksów na pozycjach (i % 3 != 0);
  - Wykorzystanie wyników do posortowania reszty (i % 3 = 0);
  - Scalenie wyników.
- **Optymalizacja w kodzie:** Użycie modułu `array` (`array('i',...)`) zamiast list Pythona, pozwala na redukcje pamięci (4 bajty vs. 28 bajtów na int).

FM-Index i Struktury Danych

- Transformata Burrowsa-Wheelera (BWT) tworzona jest z tablicy suffiksowej, a macierz zliczeń zaimplementowana jest w NumPy, co pozwala na redukcje zużycia pamięci RAM o 50%

```
wt = text[sa - 1]
mask = (bwt == c_idx).astype(np.int32)
occ[c_idx, 1:] = np.cumsum(mask)
```



Seeding & Chaining

- Generowanie **seed: P**obieramy co 29-ty k-mer długości 15

  Ilość nasion dla 1 kb: (1000−15) // 29 + 1 = 34 (1000 - 15) // 29 + 1 = 34

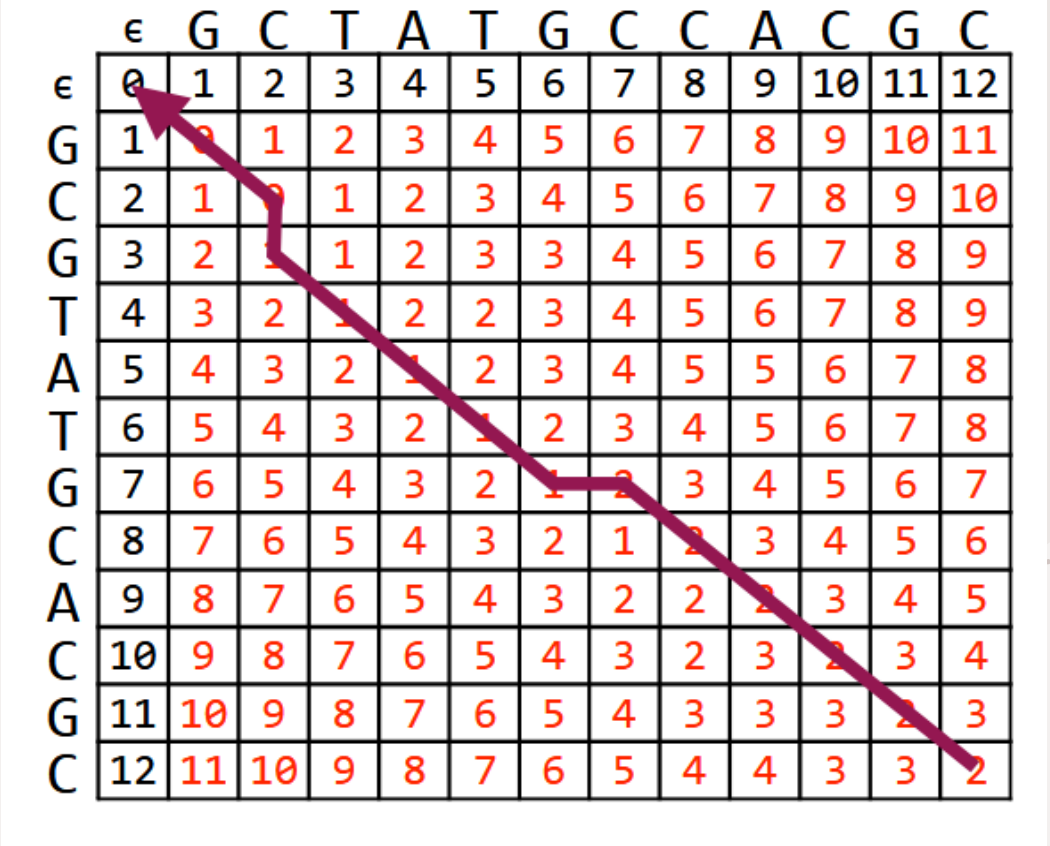
  Jeden k-mer jest bez błędów z prawdopodobieństwem:
  p = (1 − 0.12)^15 = 0.88^15 ≈ 0.147

  1 − (1 − p)^ 34 ≈ 0.995

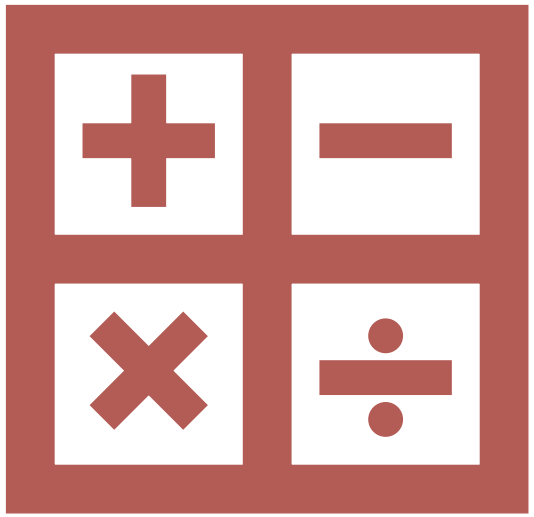
- **Wyszukiwanie seedów w FM-index** (O(k=15)), gdzie wynikiem jest przedział w SA
- **Agregacja kandydatów oraz filtrowanie:** Dla każdego seed → tworzymy histogram offsetów, wybierając 5 najlepszych.
- **Lokalne wyrównanie**: Dla każdego kandydata wykonujemy **banded_DP** w oknie `slack = 100` (tolerancja indeli).

Weryfikacja - Banded Alignment

- Implementacja algorytmu programowania dynamicznego (Smith-Waterman).
- Obliczanie macierzy tylko w wąskim pasmie wzdłuż przekątnej:
- `band = max(10, max_errors * 2 + 5)`
- Złożoność: O(L * band)
- **Heurystyka:** Jeśli minimalny koszt w bieżącym wierszu przekracza limit błędów, obliczenia są przerywane.



	ϵ	G	C	T	A	T	G	C	C	A	C	G	C
ϵ	0	1	2	3	4	5	6	7	8	9	10	11	12
G	1	0	1	2	3	4	5	6	7	8	9	10	11
C	2	1	0	1	2	3	4	5	6	7	8	9	10
G	3	2	1	1	2	3	3	4	5	6	7	8	9
T	4	3	2	1	2	2	3	4	5	6	7	8	9
A	5	4	3	2	1	2	3	4	5	5	6	7	8
T	6	5	4	3	2	1	2	3	4	5	6	7	8
G	7	6	5	4	3	2	1	2	3	4	5	6	7
C	8	7	6	5	4	3	2	1	2	3	4	5	6
A	9	8	7	6	5	4	3	2	2	2	3	4	5
C	10	9	8	7	6	5	4	3	2	3	2	3	4
G	11	10	9	8	7	6	5	4	3	3	3	2	3
C	12	11	10	9	8	7	6	5	4	4	3	3	2



Bibliografia

- **Kärkkäinen, J., & Sanders, P. (2003).** *Simple linear work suffix array construction.* (Implementacja DC3).
- **Ferragina, P., & Manzini, G. (2000).** *Opportunistic Data Structures with Applications.* (Podstawy FM-Index).
- **Li, H. (2013).** *Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM.* (*Spiritus Movens* strategii Seed-and-Extend).

DZIĘKUJEMY ZA UWAGĘ!