BWA

Burrows-Wheeler Aligner

Marcel Bargull

Fakultät für Informatik, Technische Universität Dortmund

7. Mai 2014

Inhalt

- BWA-backtrack
- 2 BWT / SA-Intervall / FM-Index
- 3 BWA Algorithmus
- 4 BWA Implementierung



BWA-backtrack

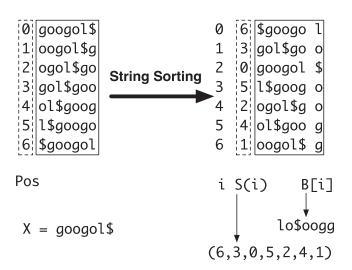
- BWT / FM-Index auf Referenzgenom
- lacksquare Ausgelegt auf ≤ 100 bp Illumina Reads
- Unterstützt Mismatches und Gaps
- Backward Search
- Bounded Traversal
 - Untere Schranke f
 ür Abweichungen im Readprefix
- Nachträgliches Paired-end Mapping mit Smith-Waterman

BWA-backtrack

- Toleriert bis zu k Abweichungen in Reads
 - < 4% der Reads mehr als k Unterschiede bei 2% Basenfehlerrate (gleichverteilt)

Optional: Abweichung in Saat (= erste n Basen) beschränken
 Bsp.: 70 bp Reads, 32 bp Saat, max. 2 Abweichungen







SA-Intervall / FM-Index

- SA-Intervall: $\left[\underline{R}(W), \overline{R}(W)\right]$
- $\underline{R}(W) = \min \left\{ k : W \text{ is the prefix of } X_{\mathcal{S}(k)} \right\}$
- $lackbox{$\overline{R}$}(W) = \max \left\{ k : W ext{ is the prefix of } X_{S(k)}
 ight\}$
- C(a): Anzahl Symbole in X, die kleiner als a sind
- O(a, i): Anzahl Vorkommen von a in B[0, i]
- $\underline{R}(aW) = C(a) + O(a, \underline{R}(W) 1) + 1$
- $\overline{R}(aW) = C(a) + O(a, \overline{R}(W))$



BWA Algorithmus

- InexactSearch
 - Eingaben:
 - W: Read
 - z: Maximale Editierdistanz
 - Ausgabe:
 - *I*: Menge von SA-Intervallen
- CalculateD
 - Eingaben:
 - W: Read
 - Ausgabe:
 - D: Untere Schranken für Abweichungen der Präfixe W[0, i] von Referenz



BWA Algorithmus

```
precalculate
   BWT B for reference string X
  Array C(\cdot) and O(\cdot, \cdot) from B
   BWT B' for the reverse reference
  Array O'(\cdot,\cdot) from B'
procedure CALCULATED(W)
  k \leftarrow 1
  I \leftarrow |X| - 1
  z \leftarrow 0
  for i = 0 to |W| - 1 do
     k \leftarrow C(W[i]) + O'(W[i], k-1) + 1
     k \leftarrow C(W[i]) + O'(W[i], I)
     if k > l then
        k \leftarrow 1
        I \leftarrow |X| - 1
        z \leftarrow z + 1
      D(i) \leftarrow z
```

```
procedure INEXACTSEARCH(W, z)
  CALCULATED(W)
  return INEXRECUR(W, |W|-1, z, 1, |X|-1)
procedure CalculateD(W)
  z \leftarrow 0
  i \leftarrow 0
  for i = 0 to |W| - 1 do
     if W[i,j] not a substring of X then
       z \leftarrow z + 1
       i \leftarrow i + 1
     D(i) \leftarrow z
```

BWA Algorithmus

```
procedure INEXRECUR(W, i, z, k, l)
  if z < D(i) then
     return Ø
  if i < 0 then
     return \{[k, l]\}
  I \leftarrow \emptyset
  I \leftarrow I \cup \text{INEXRECUR}(W, i-1, z-1, k, I)
                                                               // Insertion
  for each b \in \{A, C, G, T\} do
     k \leftarrow C(b) + O(b, k-1) + 1
     I \leftarrow C(b) + O(b, I)
     if k \le l then
        I \leftarrow I \cup INEXRECUR(W, i, z - 1, k, I)
                                                                   Deletion
        if b = W[i] then
           I \leftarrow I \cup \text{INEXRECUR}(W, i-1, z, k, I)
        else
           I \leftarrow I \cup \text{INEXRECUR}(W, i-1, z-1, k, I)
  return /
```

BWA-Implementierung

- BFS mit heapähnlicher Datenstruktur statt DFS
 - Priorisiert nach Alignment Score der partiellen Zuordnungen
- Unterschiedliche Bewertung für
 - Mismatch
 - Gap Open
 - Gap Extension
- Reverses Komplement wird zeitgleich abgearbeitet

BWA-Implementierung

- $O(\cdot, k)$ nur für jede 128te (k-te) Stelle gespeichert
- Stellen dazwischen werden "one the fly" berechnet
- S(k) wird für jede 32. Stelle gespeichert
- Gesamtspeicherbedarf bei 3 Gb Genom:
 - 2,3 GB für Single End
 - aca. 3 GB bei Paired End