Prof. Dr. Knut Reinert
Thimo Wellner
Sascha Meiers

Institut für Informatik
AG Algorithmische Bioinformatik

Algorithmen und Datenstrukturen in der Bioinformatik
Weihnachtsaufgabe WS 13

Abgabe Montag, 20.01., 15:00 Uhr per SVN

In dieser Programmieraufgabe implementieren Sie den Quasar-Algorithmus zum semi-globalen Alignment kurzer Sequenzen gegen eine Referenz-Sequenz (*Read Mapping*). Die Aufgabe wird mit **doppelter Punktzahl** bewertet und das Programm mit der schnellsten Laufzeit[1] (und dennoch akzeptabler Genauigkeit bei den Ergebnissen) wird mit einem kleinen Preis belohnt :)

Quasar Setzen Sie sich mit dem Skript[2] und den Slides aus der P-VL auseinander. Hier eine Übersicht der benötigten Schritte:

a) Einlesen von Genom- und Readdatei (strikt als DNA interpretieren; vorgegeben)

b) Aufbauen des Q-Gram Index mit $q = 10$

- Q-Gramme werden als 4-näre Zahlen interpretiert mit $A = 0, C = 1, G = 2, T = 3$
- Eine Tabelle `sa` speichert die Textpositionen sortiert nach den ersten q Buchstaben, die 2. Tabelle `hashTable` speichert für jedes mögliche q-gram die Position des ersten Vorkommens in `sa`.
- Es empfiehlt sich die Tabellen in 3 Schritten aufzubauen: (1) Zählen der Q-gramme, (2) Bilden der kummulativen Summe der Zähler und (3) Eintragen der Textpositionen in `sa` mithilfe von `hashTable`.

c) Genom in Buckets aufteilen mit $w = Readlnge$ und $b = 2w$.

d) Für jeden Read: Alle überlappenden Q-gramme (des Reads) in den Buckets (des Genoms) anzählen und Buckets über dem Threshold speichern.

e) Für jeden Read: Semi-globales Alignment gegen jene Buckets über dem Threshold.

- Semi-globales Alignment bestraft keine End-gaps im Genom, wohl aber im Read. Kein lokales Alignment.
- Distanzmaß ist die Editdistanz

[1] gemessen wird auf einem 8-Kerne Uni-Rechner
[2] `https://www.mi.fu-berlin.de/wiki/pub/ABI/AlignmentHeuristicsWS12/quasar.pdf`

- Berechnen Sie nur den Score (kein Traceback) und geben sie die Endposition des Alignments zurück.

f) Ausgeben von Read, Text**end**position und Fehleranzahl für gefundene Alignments wie in den `.result`-Dateien gezeigt. Ausgabe in Datei schreiben.

Aufruf Wie in der vorbereiteten C++ Datei vorgegeben. Der 3. Parameter ist die max. Fehlerzahl k, der letzte Paramemter die Ausgabedatei, in die Sie Ihre Ergebnisse schreiben sollen.

```
./aufgabe5 random10M.bin random10M_reads50_10k.bin 3 myresults.txt
```

Bereitgestellte Dateien

- Genomdatenbanken (komprimiert): `chr22`, `random10M` und `random100M`. Enthalten genau eine Zeile, die die gesamte Sequenz darstellt.
- Readfiles (komprimiert) mit den Namen: *Genom*`_reads`*Länge*`_`*Anzahl*`.bin.gz`. Enthalten je eine Sequenz pro Zeile, welche aus dem zugehörigen Genom simuliert wurden. Die Reads der Länge 50 enthalten maximal 3 Fehler, Reads der Länge 100 maximal 8.
- Mapping-Ergebnisse (komprimiert): *Genom*`_reads`*Länge*`_`*Anzahl*`.result.gz`. Entsprechen genau den Readfiles, nur dass sie zusätzlich Position und Fehleranzahl aus der Simulation speichern. Ihre Ausgabedateien sollen dieses Format haben.
- Vergleichsskript `a5_compare.py`. Skript zum Vergleichen Ihrer Ergebnisse mit den korrekten Ergebnissen. Listet alle Reads die Ihr Programm gar nicht, an der falschen Stelle (mehr als 20 Positionen daneben) oder mit zu vielen Fehlern aligniert hat. Aufruf durch

  ```
  python a5_compare.py myresults.txt random10k_reads50_10k.result
  ```
- Code-Snippet zum Einlesen von Genom und Reads: `aufgabe5_basis.cpp`

Praktikumshinweise Beachten Sie die Hinweise unter `https://www.mi.fu-berlin.de/w/ABI/AlDaBiWS13Praktikum`.