Chris Bielow

Institut für Informatik
Bioinformatics Solution Center

# Algorithmen und Datenstrukturen in der Bioinformatik
# Dritte Programmieraufgabe

**Abgabe Donnerstag, 10.12.,23:59 Uhr via GIT**

## P2 (10 Punkte)

Implementieren Sie den **Needleman-Wunsch-Algorithmus** zur Berechnung eines globalen Sequenz-Alignments. Nutzen Sie dazu das vorgebene Interface der Klasse *Alignment.hpp* und implementieren Sie es in einer *Alignment.cpp* (Gross/Kleinschreibung beachten!). Dieses mal müssen sie Alignment.hpp verändern und dementsprechend auch ins GIT einchecken. (beachten Sie die Hinweise in *Alignment.hpp*; modifizieren sie NICHT das public-Interface!). Implementieren Sie weiterhin eine *aufgabe3_ main.cpp*, welche ihre Alignment-Klasse benutzt.

Checken sie *Alignment.hpp, Alignment.cpp und aufgabe3_ main.cpp* unter *./aufgabe3/* ins GIT ein.

**Aufruf:** `./aufgabe3_main seqV seqH [match=3 mismatch=-1 gap=-2]`

Die beiden Sequenzen (vertikal und horizontal) werden als Kommandozeilenargumente übergeben. Zusätzlich können als weitere Argumente die Scores für *matches*, *mismatches* und *gaps* in dieser Reihenfolge angegeben werden (alle Scores sind additiv). Sind die Scores nicht angegeben, sollen $(3, -1, -2)$ als Standardwerte genommen werden. Es müssen also entweder 2 oder 5 Parameter angegeben werden. Bei anderen Eingaben soll das Programm mit *exitcode 1* und einer Fehlermeldung beendet werden.

Die Ausgabe soll in den ersten 3 Zeilen das Alignment der beiden Sequenzen enthalten (`seqV` über `seqH`), in einer 4. Zeile soll der Score angegeben werden. Achten Sie dabei auf das genaue Ausgabenformat (keine zusätzlichen Informationen):

```
./aufgabe3_main IMISSMISSISSIPPI MYMISSISAHIPPIE 3 -4 -6
IMISSMISSIS-SIPPI-
 |   ||||||  ||||
-M--YMISSISAHIPPIE
score:-5
```

# Zusatzaufgabe (4 Punkte)

Implementieren Sie lokales Sequenzalignment (Smith-Waterman) in *compute(..., true)*. Um ihr Programm besser testen zu können, erlauben sie ein 7. optionales Argument für ihr Hauptprogramm. Wenn dieses angegeben wird (egal mit welchem Wert), dann soll intern SW benutzt werden.

### Praktikumshinweise

- Sie brauchen nur eines der globalen Alignments mit optimalem Score auszugeben.
- Gibt es beim Traceback mehrere Pfade mit gleichem (optimalen) Score, beachten Sie bitte folgende Reihenfolge in der sie sich für einen Pfad entscheiden: Mismatch(↖), Gap in `seqH` (↑) und dann erst Gap in `seqV` (←) (siehe obiges Beispiel).
- Benutzen Sie keine(!) globalen Variablen für die Scoringmatrizen, Sequenzen etc, sondern Membervariablen der Alignment-Klasse.
- Compilieren Sie ihren Code mit
  *g++ -std=c++17 -g -Wall -pedantic -D_GLIBCXX_DEBUG -fsanitize=address -O2 ...* um evtl Laufzeitfehler zu finden. Der automatisierte Test zur Bestimmung ihrer Punkte wird das auch tun!
- Ein lokales Alignment ist normalerweise kürzer als ein globales Alignment:

  ```
  ./aufgabe3_main IMISSMISSISSIPPI MYMISSISAHIPPIE 3 -4 -1 letsDoLocal
  MISSIS--SIPPI
  ||||||   ||||
  MISSISAH-IPPI
  score:27
  ```

- Ein makefile liegt bei; der Aufruf

  ```
  make aufgabe3_test
  ```

  oder

  ```
  make aufgabe3_main
  ```

  erzeugt die entsprechende Executable aus den Sourcen ohne viel Tipparbeit. Schauen Sie ins Makefile und lesen Sie ein Make Tutorial wenn Sie wissen wollen was genau passiert. Compilieren sie ihr Programm mithilfe des Makefiles auf dem Poolrechner um Compile-Errors (und damit 0 Punkte) auszuschliessen