Arbeitsgruppe für Genominformatik **Universität Hamburg**
Stefan Kurtz **Zentrum für Bioinformatik**

Genominformatik
Sommersemester 2017
Übungen zur Vorlesung: Ausgabe am 11.04.2017

Punkteverteilung: Aufgabe 1.1: 4 Punkte, Aufgabe 1.2: 6 Punkte

Abgabe bis zum 20.04.2017, 23:59 Uhr.

Aufgabe 1.1 Zeigen Sie, dass für die Einheitskostenfunktion δ die folgende Rekurrenz gilt:

$$E_\delta(i,j) = \begin{cases} E_\delta(i-1,j-1) & \text{if } u[i] = v[j] \\ 1 + E_\delta(i-1,j) & \text{else if } E_\delta(i-1,j) < E_\delta(i-1,j-1) \\ 1 + \min\{E_\delta(i,j-1), E_\delta(i-1,j-1)\} & \text{otherwise} \end{cases}$$

Tipp: Führen Sie in einer Fallunterscheidung jeden der drei Fälle auf die bekannte Rekurrenz für $E_\delta(i,j)$ zurück und beweisen Sie formal, dass die entsprechende Beziehung gilt. Die Ungleichungen in den Beobachtungen am Anfang des Kapitels über die schnelle Berechnung der Edit-Distanz sind dabei hilfreich.

Aufgabe 1.2 Implementieren Sie den in der Vorlesung vorgestellten output-sensitiven Algorithmus zur Berechnung der Edit-Distanz von zwei Sequenzen u und v für die Einheitskostenfunktion. Dieser Algorithmus benutzt eine Matrix von *front*-Werten $front(h,d)$ für $d \geq 0$ und $-d \leq h \leq d$. Gehen Sie bei Ihrer Implementierung wie folgt vor:

1. Überlegen Sie sich, wie man die $front$-Werte der d-ten Generation, d.h. die Werte $front(h,d)$ für $-d \leq h \leq d$ speichern kann. Falls Sie die $front$-Werte in einem Array speichern wollen, sollte beücksichtigt werden, dass h negativ sein kann, aber in einem Array keine nicht-negativen Indizes möglich sind. Daher muss man die Addressierung so gestalten, dass der Wert für $h = -d$ an Index 0 gespeichert wird.
2. Implementieren Sie eine Funktion $lcplen$, die für ein Paar von Substrings von u und v die Länge des längsten gemeinsamen Präfix der Substrings berechnet. Dabei sollen die übergebenen Substrings niemals Kopien der Substrings von u und v sein, denn das würde zu einem quadratischen Algorithmus führen. Verwenden Sie daher als Argumente der $lcplen$-Funktion Verweise auf die entsprechenden Suffixe und ihre Länge.
3. Implementieren Sie eine Funktion `outsense_next_front`, die aus der $(d-1)$-ten Generation die d-te Generation von $front$-Werten berechnet. Hier ist der Kopf der Funktion in C-Syntax:

```
void outsense_next_front(unsigned long *destfront,
                         const unsigned long *sourcefront,
                         unsigned long d,
                         const unsigned char *useq,
                         unsigned long ulen,
                         const unsigned char *vseq,
                         unsigned long vlen)
```

`useq` und `vseq` sind Zeiger auf die beiden Sequenzen u und v der Längen `ulen` und `vlen`. `sourcefront` ist ein Zeiger auf die $(d-1)$-te Generation von $front$-Werten und `destfront` ist ein Zeiger auf die d-te Generation von $front$-Werten. Es gilt dabei `d>0`. Beachten Sie, dass die Arrays von *front*-Werten niemals kopiert werden.

4. Implementieren Sie eine Funktion `outsense_edist`, die den Speicherplatz für die jeweiligen Generationen von *front*-Werten bereitstellt, die $front(0,0)$ berechnet, und die durch Aufrufe von `outsense_next_front` nacheinander die Generationen von *front*-Werten bestimmt. Sobald das Abbruchkriterium $-d \leq vlen - ulen \leq d$ und $front(vlen - ulen, d) = ulen$ gilt, ist der aktuelle d-Wert die Edit-Distanz, die von der Funktion als `return`-Wert geliefert wird.

   Hier ist der Kopf der Funktion in C-Syntax:

   ```
   unsigned long outsense_edist(const unsigned char *useq,
                                unsigned long ulen,
                                const unsigned char *vseq,
                                unsigned long vlen)
   ```

5. Falls Sie Ihr Programm in C implementiert haben, finden Sie im Material zur Übung (siehe STiNE) einen Testrahmen, der es erlaubt, Ihr Programm einfach zu testen. Sie müssen lediglich in einer Datei `outsenseedist.c` die Funktion `outsense_edist` implementieren, `make` aufrufen und dann `outsenseedist.x` q aufrufen, wobei q eine positive ganze Zahl ≤ 9 ist. Es werden dann alle Paare von Sequenzen der Längen $\leq q$ über dem Alphabet $\{a, b\}$ generiert, und für jedes Paar wird getestet, ob die von `outsense_edist` berechnete Edit-Distanz korrekt ist.

6. Falls Sie Ihr Programm nicht in C implementiert haben, finden Sie in der Datei `testcases.tsv` 1 900 Testfälle, die Sie verwenden sollen. Jede Zeile enthält Tabulator-separiert zwei kurze Sequenzen und die entsprechende Einheitskosten-Edit-Distanz. Lesen Sie in Ihrem Programm zeilenweise die 1 900 Sequenzpaare, werten dann ihre Distanz aus, verifizieren, dass diese identisch ist mit der vorgegebenen Edit-Distanz.

Die Lösungen zu diesen Aufgaben werden am 25.04.2017 besprochen.