Sommersemester 2003
Dipl.-Inf. A. Thum, Dr. A. Gerisch

6. Übung „Algorithmen der Bioinformatik I“

Abgabe am 12. Juni 2003 in der Übung

Institut für Informatik
Martin-Luther-Universität Halle-Wittenberg

Aufgabe 1. In der Vorlesung wurden die Begriffe *maximales Paar* (i, j, k) und *maximaler Repeat (wiederholter Teilstring)* eingeführt und Eigenschaften dazu bewiesen. Außerdem wurde ein Algorithmus angegeben, der alle maximalen Repeats ermittelt.

1. Formulieren Sie eine Hypothese über die maximale Anzahl von maximalen Repeats in einem String der Länge n und beweisen Sie sie.

2. Beschreiben Sie eine Methode, mit der alle maximalen Paare eines Strings S der Länge n gefunden werden können und diskutieren Sie ihre Korrektheit. Finden Sie nun auch einen Algorithmus, der in Zeit $O(n + Anzahl\ der\ Paare)$ läuft! Welchen Speicherplatzbedarf hat Ihr Algorithmus (abgesehen vom Bedarf für den üblichen Suffixbaum)? Modifizieren Sie den Algorithmus so, dass der Speicherplatzbedarf $\mathcal{O}(n)$ nicht übersteigt.

   Hinweis: Für jeden inneren Knoten v des Suffixbaums von S sollen maximal soviele Listen erzeugt werden, wie es Buchstaben im Alphabet gibt. Die Liste im Knoten v, die zum Buchstaben x gehört, soll alle Anfangspositionen von Teilstrings des Strings S enthalten, die mit der Pfadbeschriftung von v im Suffixbaum übereinstimmen und als linkes Zeichen ein x haben. Aus diesen Listen können nun die maximalen Paare konstruiert werden.

3. Was müsste an Ihrem Algorithmus zum Finden aller maximalen Paare geändert werden, wenn nur maximale Paare einer minimalen Länge m gesucht sind? Welchen Einfluss hat das auf die Laufzeitabschätzung?

Aufgabe 2. In der Vorlesung wurde zwei Möglichkeiten zur Konstruktion eines generalisierten Suffixbaums für eine Menge von Strings $S_1, S_2, ..., S_m$ diskutiert. Im ersten Fall werden alle Strings, getrennt durch unterschiedliche Trennzeichen, aneinandergehängt und für diesen langen String wird der Suffixbaum aufgebaut, der dann noch leicht nachbearbeitet werden muss. Alternativ kann man die Strings auch sequentiell mit dem Ukkonen-Algorithmus in einen Suffixbaum integrieren. Dazu wird zuerst für den String S_1, versehen mit einem Endezeichen, der Suffixbaum konstruiert. Zur Integration von S_2, versehen mit dem gleichen Endezeichen, wird nun wiederum der Ukkonen-Algorithmus benutzt. Der Algorithmus wird aber sofort mit Phase $i+1$ gestartet, wobei i die Länge des längsten Präfixes von S_2 ist, der, von der Wurzel des Suffixbaums ausgehend, im Suffixbaum von S_1 enthalten ist. Für die weiteren Strings $S_3, ..., S_m$ erfolgt die Integration analog. Beweisen Sie, dass dieses Vorgehen korrekt ist.

Aufgabe 3. Entwickeln und diskutieren Sie eine Methode zur Entfernung eines Strings aus einem generalisierten Suffixbaum, die in $\mathcal{O}(|String|)$ läuft und formulieren Sie diese in Pseudocode.