Exposé zur Diplomarbeit

Entwurf und Implementierung eines generischen Substring-Index

David Weese

12. September 2005

Betreuer: Prof. Dr. Ulf Leser
Institut für Informatik, Humboldt-Universität zu Berlin
Prof. Dr. Knut Reinert
Institut für Informatik, Freie Universität Berlin

Motivation

Mit der Sequenzierung kompletter Genome in den letzten Jahren wurden erhebliche Datenmengen veröffentlicht, die nun weiter analysiert werden müssen. Viele der dafür benötigten Algorithmen, suchen sehr oft nach Vorkommen bestimmter Teilsequenzen innerhalb dieser Genome, so dass es Datenstrukturen und Algorithmen bedarf, die allgemein das Auffinden von Mustern in großen Datensätzen effizient realisieren. Ein Substring-Index ist eine Datenstruktur zu einem gegebenen String s, die Informationen zur lexikographischen Ordnung der Teilstrings von s enthält und Methoden zur Verfügung stellt, effizient nach Vorkommen von Strings zu suchen.
Ein generischer Substring-Index ist weitestgehend typ- und problemunabhängig und nicht nur auf Anwendungen in der Bioinformatik festgelegt. Ebenso ist der Einsatz in Suchmaschinen, o.ä. denkbar.
SEQAN ist eine Software-Bibliothek, die in der Arbeitsgruppe Algorithmische Bioinformatik der Freien Universität Berlin entwickelt wird, Algorithmen und Datenstrukturen zur Analyse von Genom- oder Proteinsequenzen bereit stellt und die Entwicklung von eigenen Algorithmen erleichtern soll.

Zielsetzung

Ziel dieser Arbeit ist die Entwicklung, Implementierung und Analyse eines generischen Substring-Index für große Strings. Dieser soll in die Software-Bibliothek SEQAN integriert werden.

Vorgehen

Inhaltlich besteht die Arbeit aus den drei Teilen: Entwicklung, Implementierung und Analyse.
Zunächst soll ein abstrakter Datentyp 'Substring-Index' konzipiert werden, der einen schnellen Zugriff auf alle Substrings einer Menge von Strings erlaubt. In dieser Arbeit wird dabei der Schwerpunkt vor allen Dingen auf der exakten Suche von Substrings im Substring-Index liegen. Anschließend soll dieser abstrakte Datentyp implementiert und in die Softwarebibliothek SEQAN integriert werden. Dem Anwender der SEQAN-Bibliothek werden u.a. Methoden zur Verfügung gestellt zum Aufbau des Index zu einem oder mehreren Strings und zum Suchen nach Teilstrings.
Für exaktes Substring-Matching innerhalb sehr großer Datenmengen haben sich Suffix-Arrays und Enhanced Suffix-Arrays als geeignet erwiesen. Mit ihnen lassen sich Vorkommen von Substrings der Länge m innerhalb eines festen Strings s der Länge n in $O(m \cdot \log n)$ bzw. $O(m)$ finden [1]. Ihre Struktur ist einfacher und kompakter als die der Suffix-Bäume und viele auf Suffix-Bäumen basierende Algorithmen lassen sich auf Suffix-Arrays oder Enhanced Suffix-Arrays übertragen [2].
Der Skew-Algorithmus [3], [4] ermöglicht erstmals den Aufbau von Suffix-Arrays in linearer Zeit. Das Enhanced Suffix-Array ist eine Erweiterung des Suffix-Arrays um eine LCP-Tabelle, die ebenfalls in linearer Zeit konstruiert werden kann [5].
Da der Algorithmus mindestens $4n$ Byte zusätzlichen Hauptspeicher benötigt, um das Array aufzubauen, soll ausgehend von der Studienarbeit *Implementierung des Skew-Algorithmus* nun der Algorithmus für die Verwendung von externem Speicher modifiziert werden.
Das Konzept des Pipelinings [6] eignet sich hervoragend für externen Speicher und die Minimierung von Random Accesses. Es werden einfache und generische Pipeline-Module entwickelt und implementiert, wie Filter, Funktoren, Sortierer, Permutierer, etc., die es dem Entwickler ermöglichen sollen, eigene Algorithmen modular zusammenzusetzen. Der Skew-Algorithmus wird sequentialisiert und kann nun mit Hilfe dieser Module auf einfache Art und Weise implementiert werden. Ebenso soll der Skew-Algorithmus mit der Erweiterung um Difference Covers [4] in Pipeline-Form gebracht werden und dem Anwender zur Verfügung gestellt werden.
Dem Benutzer der SEQAN-Bibliothek soll überlassen werden, ob aus dem Suffix-Array ein Enhanced Suffix-Array konstruiert werden soll, das die Substring-Suche weiter beschleunigt. Ausgehend davon soll der Substring-Index spätere Erweiterungen ermöglichen, bspw. zur Bestimmung von maximalen oder supermaximalen Repeats [2].
In der Analyse sollen die verschiedenen Implementierungen an biologischen Daten getestet und mit anderen bekannten Suchalgorithmen verglichen werden. Genauer untersucht werden Ausführungszeiten, Speicherbedarf und

I/O-Zugriffe, die für den Aufbau des Index und die Suche von Substrings benötigt werden. Weiter sollen theoretische average-case und worst-case Abschätzungen für die Zeit-, Speicher- und I/O-Komplexitäten der Implementierungen gefunden werden und mit den empirischen Ergebnissen verglichen werden. Den theoretischen Abschätzungen sollen verschieden I/O-Modelle bspw. von Vitter und Shriver [7] zu Grunde liegen.

Literatur

[1] M. I. Abouelhoda, E. Ohlebusch, S. Kurtz, *Optimal exact string matching based on suffix arrays*, Proc. 9th Symposium on String Processing and Information Retrieval, Springer, 2002.

[2] M. I. Abouelhoda, S. Kurtz, E. Ohlebusch, *The enhanced suffix array and its applications to genome analysis*, Proc. 2nd Workshop on Algorithms in Bioinformatics, Springer, 2002.

[3] J. Kärkkäinen, P. Sanders, *Simple Linear Work Suffix Array Construction*, Proc. 30th International Colloquium on Automata, Languages and Programming (ICALP '03), 2003.

[4] J. Kärkkäinen, P. Sanders, S. Burkhardt, *Linear Work Suffix Array Construction*, Journal of the ACM, To appear.

[5] T. Kasai, G. Lee, H. Arimura, S. Arikawa, and K. Park, *Linear-Time Longest-Common-Preffix Computation in Suffix Arrays and its Applications*, Proc. 12th Annual Symposium on Combinatorial Pattern Matching, Springer, 2001.

[6] R. Dementiev, J. Mehnert, J. Kärkkäinen, *Better external memory suffix array construction*, Workshop on Algorithm Engineering & Experiments, Vancouver, 2005.

[7] J. S. Vitter, E. A. M. Shriver, *Algorithms for parallel memory I: Two level memories*, Algorithmica, 1994.