CS Final Year Project
fyp20013

Detailed Project Plan
for
Advanced Indexing Techniques and Their Applications in Bioinformatics

supervised by Dr. H.F. Ting
Student Name: Lai Ka Yee (3035473244)
Background

Bioinformatics is a discipline that combines biology, computer science and some other fields together to study biological data. Usually, the data involved are very large. It takes enormous amount of time and effort to handle those data. For instance, the first human genome project, which aims to construct the whole DNA sequence of a human genome, spans over a decade of time after initiated in 1990 [1]. With help of computational tools, we hope to make the retrieval and processing of biological data more efficient. So that bioinformaticians can consume less time to process the data, such as aligning genetic sequence segments and visualising the structure of protein, and hence, pay more attention to analyse the findings.

In bioinformatics, sometimes it involves some pattern matching problems, like protein sequencing and DNA sequencing. The sequencing problems compare a segment of biological data obtained with the reference data to find the matches and mismatches. The mismatch parts are called the variants. Variants are responsible for the individuality of each creature. By studying the variants, bioinformaticians may find out the property of the variant sequence, which makes the pattern matching interesting to them.

But as mentioned before, the data are massive, a full human genome itself is three billion characters long. It would be too naive if we simply compare the pattern with every sub-sequence of the reference data. Fortunately, there are some indexing methods that bioinformaticians can use to boosts the matching.

In this project, I will discuss those indexing techniques that can be utilised in pattern matching problem in bioinformatics. At first, I will talk about the suffix array. If time allows, other indexing techniques such as Burrows-Wheeler transform will also be covered.

Introduction to Suffix Array

Suffix array was first proposed by Manber and Myers in 1990 [2]. It serves as an alternative to implement suffix tree in a more space-efficient way. Both suffix array and suffix tree are a kind of data structure, they need to be constructed before searching any pattern through a text string.

Suffix Array is an array storing the indices of all suffixes of a string. For every string, we may append a ‘$’ at the tail to denote the termination of the string. And we define ‘$’ to be lexicographically smaller than any other letters in the string. Then, we list out all the suffixes of the string, each of them is labelled an index number to denote the starting position of suffix in the string, see Figure 1. Figure 1 uses a segment of a genome sequence “GCTTACTGTA$” as the string S.

<table>
<thead>
<tr>
<th>S[i]</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>T</th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>T</th>
<th>A</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>
Next, we will sort the suffixes in ascending lexicographic order, and hence obtain the result in Figure 2.

Extract the index in the first column in Figure 2, we will get the suffix array of “GCTTACTGTA$”, which is \{10, 9, 4, 5, 1, 0, 7, 8, 3, 6, 2\}.

Constructing the suffix array of a string input can be computational hard as the input string is usually long, and hence, sorting all the suffixes can be a challenging task. After suffix array is first introduced, there are people working on the construction algorithms of suffix array. They aim to achieve linear asymptotic time complexity, fast practical running time and small space usage of the construction algorithm, where the three criteria are proposed by Puglisi et al. [3]. The SA-IS algorithm invented by Nong et al. in 2009 has fulfilled these requirements [4]. This algorithm was later enhanced by Felip et al. with some additional information and the extended algorithm was named gSAIS [5]. Before all that, another well-known algorithm called DC3 by by Kärkkäinen and Sanders can also run in linear time complexity [6].

After the construction, Manber and Myers (1990) [2] mentioned that we can use the suffix array for searching a pattern using binary search. Since the elements in the suffix array are all sorted, we can compare the pattern with the middle element of the suffix array, if they are equal, we have found the result. If the pattern is lexicographically smaller than the substring obtained from the middle element, we restrict the ending position to be the middle. Else, restrict the starting position to be the middle position. Then go on with the binary search. The time complexity of this pattern searching using binary search is \(O(p \log n)\), where \(p\) is the length of the pattern, \(n\) is the length of the string. Manber and Myers [2] then provided another \(O(p + \log n)\) time for the searching using Longest Common Prefix.
Project Content and Objective

This project will give an overview of what we currently have for solving the indexing problem and see what they can do to help the pattern matching problem in genome sequencing or protein sequencing.

I will review and compare different suffix array construction algorithms to see which ones are the good candidates to be applied to bioinformatics to solve the sequencing problem. Through studying the indexing algorithms, I hope to practise the achievements in computer science to bioinformatics. We may be able to improve the pattern searching in bioinformatics.

Other than bioinformatics, the indexing techniques can also be applied into other domains. For instance, we may encounter a pattern matching problem when having a search query in a text document. And the suffix array algorithms that I am going to look in this project or other indexing methods can aid in these problems.

Methodology

In this project, I will study various suffix array construction algorithms and compare the performance of those algorithms. I will also test the program of the algorithm by building the suffix array of some text samples. Through testing them, I would like to know how their performances differ when applied to a real-life problem. They will be evaluated by their practical running time, the memory used and the size of the workspace, which is the additional memory used except the input string and the output suffix array. The testing criteria is based on the construction algorithm goals mentioned by Puglisi et al [3] as mentioned above. The time complexity will not be evaluated as most of the algorithms studied in this project run in linear time. The text samples that I will use includes the reference genome retrieved from GenBank [7], the protein sequence and some articles.
Schedule (tentative)

<table>
<thead>
<tr>
<th>Dates</th>
<th>Deliverables / Tasks</th>
</tr>
</thead>
</table>
| 4 Oct, 2020        | • detail project plan  
|                    | • project website                                                                |
| Oct - Dec, 2020    | • study of suffix array construction algorithms (including DC3, SAIS, gSAIS, gSACA) 
|                    | • testing of the suffix array construction algorithm                              |
| 11-15 Jan, 2021    | • First presentation                                                              |
| 24 Jan, 2021       | • Preliminary Implementation  
|                    | • Detailed interim report                                                        |
| Jan – Apr, 2021    | • study of suffix array construction algorithms                                   |
|                    | • study of other indexing algorithms                                              |
| 18 Apr, 2021       | • Finalised implementation  
|                    | • Final report                                                                    |
| 19-23 Apr, 2021    | • Final presentation                                                              |
| 4 May, 2021        | • Project exhibition                                                             |

If the progress is ideal, I may get on with other indexing techniques, like Burrows-Wheeler transform.

References / Bibliography


