Advanced Algorithms in Bioinformatics (P4)

Sequence and Structure Analysis

Freie Universität Berlin, Institut für Informatik
Prof. Dr. Knut Reinert, Sandro Andreotti
Sommersemester 2009

6. Exercise sheet, 30. May 2009
Discussion: 8. June 2009

Exercise 1.

Efficient searching with suffix arrays

In the lecture we discussed two strategies how to reduce the number of redundant character comparisons during a binary search. One uses the mlr values, while the other one makes use of lcp values. The mlr trick in practice already brings the running time to $O(m + \log n)$.

a Find a pair of pattern and text where the mlr trick still needs time $O(m \log n)$.

b For the same text and pattern perform the binary search using the lcp values. To compute the lcp values construct the binary search tree and use the height array (you can either compute this array with the Kasai algorithm or simply by counting).

c Prove that using the lcp method the search algorithm does at most $O(m + \log n)$ character comparisons.

Exercise 2.

Pairwise Segment Match Refinement

Draw the projection maps α and β fot the following aligment:

```
ABCD---HI
AB--EFGHI
```

Exercise 3.

Minimal Resolved Match Refinement

Prove the following Lemma:

Lemma 1. *There exists a unique resolved refinement $\bar{\mathcal{S}}$ of $\mathcal{S}$ of minimal cardinality.*

Proof: Sketch: Consider two different resolved refinements $\mathcal{S}_1$ and $\mathcal{S}_2$ of $\mathcal{S}$,both of minimal cardinality. Divide proof into two cases. 1) ($supp_A(\mathcal{S}_1) \neq supp_A(\mathcal{S}_2)$ 2) $supp_A(\mathcal{S}_1) = supp_A(\mathcal{S}_2)$, $supp_B(\mathcal{S}_1) = supp_B(\mathcal{S}_2)$

Exercise 4.

Running Time

From the lecture we know that in order to insert an A-position h into V_A, we first have to determine which segment matches contain it. Using a range tree this can be achieved in time $O(\log^2 n+k)$. (see Lemma 4.)

From Wikipedia, the free encyclopedia:
In computer science, a range tree is an ordered tree data structure to hold a list of points. It allows all points within a given range to be efficiently retrieved, and is typically used in two or higher dimensions. It is similar to a kd-tree except with faster query times of O(log2 n + k) but worse storage of O(n log n), with n being the number of points in the tree, and k being the number of points retrieved for a given query.

To answer the question it is not necessary to completely understand the query process in a range tree. For our problem we would use a 2-d range tree for each sequence. Each segment match then defines a point in a two dimensional coordinate system. What do the dimensions define? What would be the query to find all segment matches containing the A-position h. Draw the coordinate system and show where all these segment matches lie in the 2-dimensional space.

Reading through the slides *http://www.cs.umd.edu/class/fall2001/cmsc420/rt.ps.gz* will make it much easier to understand the range tree.