Weekplan: Grammar Compression and Random Access

Philip Bille

References and Reading

[1] Random Access to Grammar-Compressed Strings and Trees, P. Bille, G. M. Landau, R. Raman, K. Sadakane, S. Rao Satti, O. Weimann, SICOMP, 2015

[2] Perfect hashing for strings: Formalization and algorithms, M. Farach, S. Muthukrishnan, CPM 2005.

We recommend reading [1] in detail and browsing the section on weighted ancestors in [2].

Exercises

1 Re-Pair Compression

1.1 [w] Run the Re-Pair compression algorithm on the string `abababababcbccccddcabababaaa`

1.2 [w] As described Re-Pair does not produce a straight-line program. Show how to modify the output to get a straight-line program.

2 Re-Pair Speed Let S be a string of length N. Solve the following exercises.

2.1 [w] Give a straightforward algorithm that implements the Re-Pair algorithm in $O(N^2)$ time.

2.2 Show how to implement Re-Pair in $O(N)$ time. *Hint:* dynamically maintain frequencies of occurrences of pairs.

3 Grammar Compression Rate Suppose you are given a string S of size N. What is the best possible grammar compression size?

4 LZ78 and Grammar Compression Let T be a LZ78 trie representing a string S of length N. Show how to convert T into a grammar of size $O(n)$ representing the string S.

5 Grammars and Heavy-Paths Consider the following grammar G.

$$
\begin{aligned}
X_1 &= \mathtt{ab} \\
X_2 &= X_1\mathtt{a} \\
X_3 &= \mathtt{a}X_2 \\
X_4 &= X_2X_6 \\
X_5 &= \mathtt{ca} \\
X_6 &= X_5\mathtt{a} \\
X_7 &= X_3X_4
\end{aligned}
$$

Solve the following exercises.

5.1 [w] Draw the parse tree for G.

5.2 Draw a heavy path decomposition for G. If the children have the same size, pick the left child as heavy.

5.3 Draw the heavy path suffix tree for G.

6 Weighted Ancestor Let T be tree with n nodes. Each edge is assigned a weight and the total *cumulative size* of all weights is at most N. We want a data structure that supports the following operation on T. Given a node v and an integer x define the following operation:

- WA(v, x): return the closest ancestor of v of distance $\geq x$.

Solve the following exercises.

6.1 $[w]$ Give a simple data structure that supports WA queries in $O(n^2)$ space and $O(\log\log N)$ time.

The *level ancestor problem* is to preprocess a tree into a data structure supporting *level ancestor queries*, that is, given a node a node v and an integer k, return the kth ancestor of v. Assume in the following that you have a linear space solution that support queries in constant time.

6.2 Give a data structure that supports WA queries in $O(n)$ space and $O(\log n)$ time.

6.3 Give a data structure that supports WA queries $O(n)$ space and $O(\log\log n + \log\log N)$ time. *Hint:* heavy path decomposition on T.

7 Biased Search Let S be a set of integers $s_1 \leq \cdots \leq s_n$ from a universe $[0, .., N-1]$. For simplicity add $s_0 = 0$ and $s_{n+1} = N-1$ to S. Let $I(x)$ denote the interval successor(x) − predecessor(x). Our goal is to develop a simple comparison-based binary tree data structure that supports predecessor(x) query in time $O\left(\log \frac{N}{I(x)}\right)$. Hence, the query time becomes faster as the interval size $I(x)$ increases.

Consider the intervals $[s_0, s_1], [s_1, s_2], \ldots, [s_n, s_{n+1}]$. The *interval-biased search tree* is a binary tree that stores an interval at each node. The tree is described recursively as follows.

- Let m be such that $(s_{n+1} - s_0)/2 \in [s_m, s_{m+1}]$. The root of the tree stores $[s_m, s_{m+1}]$.
- The left child of the root is the interval-biased search tree storing the intervals $[s_0, s_1], \ldots, [s_{m-1}, s_m]$ and the right child is the interval-biased search tree storing the intervals $[s_{m+1}, s_{m+2}], \ldots, [s_n, s_{n+1}]$.

Solve the following exercises.

7.1 Argue that any interval of length ℓ such that $N/2^{j+1} \leq \ell \leq N/2^j$ must be stored in a node of depth at most j.

7.2 Use the interval-biased search tree to support predecessor(x) queries in time $O\left(\log \frac{N}{I(x)}\right)$.