Simple and efficient fully-functional succinct trees

Reelika Tõnisson
Institute of Computer Science
University of Tartu

Abstract

The fully-functional succinct tree representation of Navarro and Sadakane (2014) \cite{2} supports many operations in constant time using $2n + o(n)$ bits. Since the full idea is hard to implement, only a simplified version with $O(lg n)$ operation time has been implemented and shown to be practical and competitive. Cordova and Navarro \cite{1} describe a new variant of the original idea that has worst-case time $O(lg lg n)$ for the operations and is much simpler to implement. An implementation based on this version is experimentally shown to be superior to existing implementations.

1 INTRODUCTION

Combinatorial arguments show that it is possible to represent any ordinal tree with $n$ nodes using less than $2n$ bits of space. The number of such trees is $\frac{1}{n} \binom{2n-2}{n-1}$, which is the $(n-1)$th Catalan number. A simple way to encode any ordinal tree in $2n$ bits is the so-called balanced parentheses (BP) representation. However, efficiently navigating the tree using that representation is challenging.

Navarro and Sadakane \cite{2} introduced a new representation that was based on BP. It was said to be fully-functional because it supported all of the operations in constant time and using a single set of structures. This enabled the development of an efficient implementation.

Cordova and Navarro \cite{1} introduce an alternative construction that builds on binary rmM-trees (range min-max trees) and is simple to implement. It reaches $O(lg lg n)$ time, and requires $2n + O(n/lg n)$ bits of space. They describe a new implementation building on these ideas, and experimentally show that it becomes the new state-of-the-art implementation of fully-functional succinct trees.

2 BASIC CONCEPTE

2.1 BP representation

As said in the Introduction, an ordinal tree with $n$ nodes is represented with $2n$ parentheses by opening a parenthesis when we arrive at a node and closing it when we leave the node.

![Figure 1: An ordinal tree on the left (preorder numbers) and its BP representation on the right.](image)

The resulting sequence is balanced. An example is shown on Figure 1.

2.2 Range min-max trees

Cordova and Navarro \cite{1} describe the simple version of the structure used by Navarro and Sadakane \cite{2}. They choose a block size $b$. Then, the range min-max tree of $B[1, 2n]$ is a complete binary tree where the $k$th leaf covers $B[(k-1)b+1, kb]$. Each rmM-tree node $v$ stores the following fields:

- $v.e$ - total excess of the area covered by $v$,
- $v.m$ - minimum excess in this area,
- $v.M$ - maximum excess in the area,
- $v.n$ - number of times the minimum excess occurs in the area.

RmM-tree can be stored without pointers, like a heap. See Figure 2.
Figure 2: The rmM-tree of Figure 1. The numbers below are excess(i) (number of opening minus closing parentheses).

3 AN $O(\lg \lg n)$ TIME SOLUTION

The main idea for this solution is to cut a bitvector $B[1, 2n]$ into $n' = 2n/\beta$ buckets of $\beta = \Theta(\lg^3 n)$ bits. Cordova and Navarro [1] maintain one rmM-tree for each bucket. The block size of the rmM-trees is set to $b = \lg n \lg \lg n$. This maintains the extra space of each rmM-tree within $O((\beta/b) \lg \beta)$ bits, which adds up to $O((n/b) \lg \beta) = O(n/\lg n)$ bits. Operation times also stay $O(\lg \lg n)$. Therefore, the operations that are solved within a bucket take $O(\lg \lg n)$ time.

4 EXPERIMENTAL RESULTS

Cordova and Navarro [1] used a fixed bucket size of $\beta = 2^{15}$ parentheses (i.e., 4 KB). To measure the performance of the implementation they used two public datasets: wiki with 498,753,916 and prot with 670,721,008 parentheses.

The authors of [1] replicate the benchmark methodology used by Arroyuelo et al. [3]. They fix a probability $p \in [0, 1]$ and generate a sample dataset of nodes by performing a depth-first traversal of the tree in which they descend to a random child and with probability $p$, they also descend to other child. All the experiments were ran on an Intel(R) Core(TM) i5 running at 2.7 GHz with 8 GB of RAM running Mac OS X 10.10.5.

Figure 3 shows the result for close operation. The times reported are in microseconds. The space is reported in bits per node (bpn). The new-prefix refers to the implementation of the new structure, while sdsd- refers to the SDSL implementation.

Figure 3: Space-time tradeoffs for the new implementation and the SDSL baseline, for operation close (left), when $p = 0.0$.

The implementation from Cordova and Navarro for operation "close" is considerably faster than SDSL. For $p = 0.0$, they are up to 4 times faster when using the least space.

5 CONCLUSIONS

Cordova and Navarro [1] have described a solution for representing ordinal trees of $n$ nodes within $2n + O(n/\lg n)$ bits of space. The solution solves a large number of queries in time $O(\lg \lg n)$. They presented a practical implementation of the solution and showed that it achieves better space–time tradeoffs than current state-of-the-art implementations. This shows that the new design has practical value. The implementation is publicly available at http://www.dcc.uchile.cl/gnavarro/software.

6 ACKNOWLEDGMENT

The author of this report has received the IT Academy Specialization Stipend for the Autumn semester of academic year 2017/18.

References