Parallel Construction of Succinct Trees

Leo Ferres[a], José Fuentes-Sepúlveda[a]
Meng He[b], Norbert Zeh[b]

June 29, 2015

[a]University of Concepción, Chile
[b]Dalhousie University, Canada

Introduction

Example of trees

- XML document
    - Wikipedia: 249,376,957 nodes.
    - Open Street map: 2,337,888,179 nodes.
- Suffix trees
    - Protein document: 335,360,503 nodes.
    - DNA document: 577,241,087 nodes.

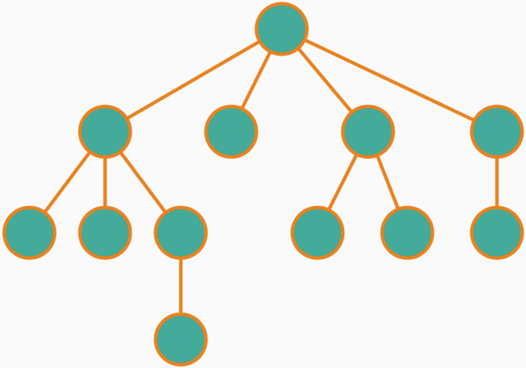
Introduction

- A succinct representation of a tree reduces the space needed to represent it while supporting operations in optimal time *[Jacobson, 1989]*.
- Still, succinct tree representations are costlier to build than tradicional representation, e.g., pointer-based representations.
- Multicore parallelism has been successful in improving construction of other succinct data structures, such as, Wavelet trees *[Fuentes-Sepúlveda et al., 2014]*.
- Our paper's contribution: A theoretical and practical algorithm for succinct tree construction on multicore *SMP* machines.

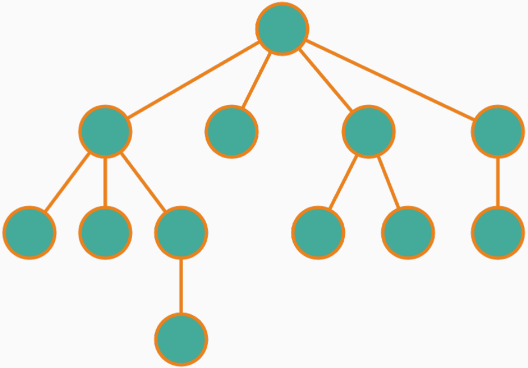
Preliminaries

SUCCINCT DATA STRUCTURES

- A succinct data structure is a space-efficient representation of a data structure which uses $(1 + o(1))lwr$ bits.
- In particular, the information-theoretic lower bound to represent the topology of a tree with n nodes is $2n$ bits.
- A work proposes succinct tree representation that uses $2n + O(n/polylog(n))$ bits *[Navarro and Sadakane, 2014]*.



Succinct data structures



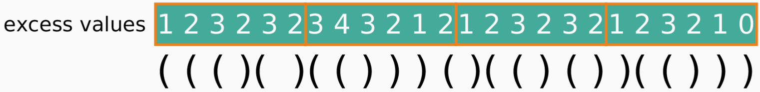
((( ) ( ) (( ))) ( ) (( ) ( )) (( )))

Range min-max tree[†]

((( )( )(( ))) ( )(( ) ( ))(( )))

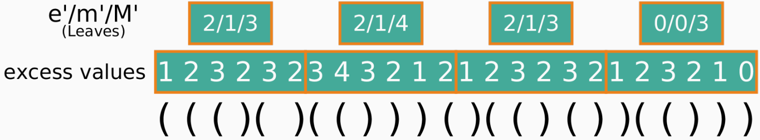
[†]G. Navarro and K. Sadakane. *Fully-functional static and dynamic succinct tree.* ACM Trans. Algorithms. 2014.

Range min-max tree[†]



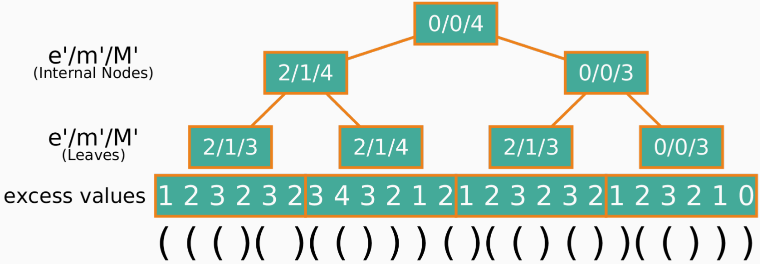
[†] G. Navarro and K. Sadakane. *Fully-functional static and dynamic succinct tree.* ACM Trans. Algorithms. 2014.

Range min-max tree[†]

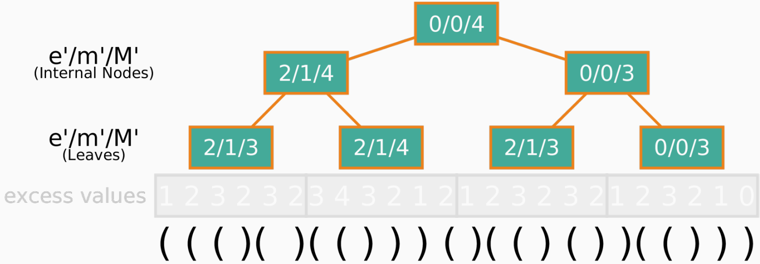


[†] G. Navarro and K. Sadakane. *Fully-functional static and dynamic succinct tree.* ACM Trans. Algorithms. 2014.

Range min-max tree[†]

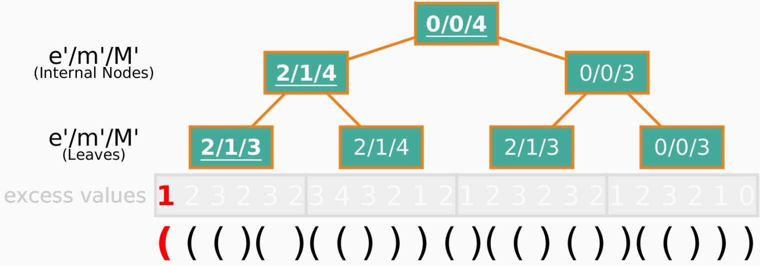


[†] G. Navarro and K. Sadakane. *Fully-functional static and dynamic succinct tree.* ACM Trans. Algorithms. 2014.



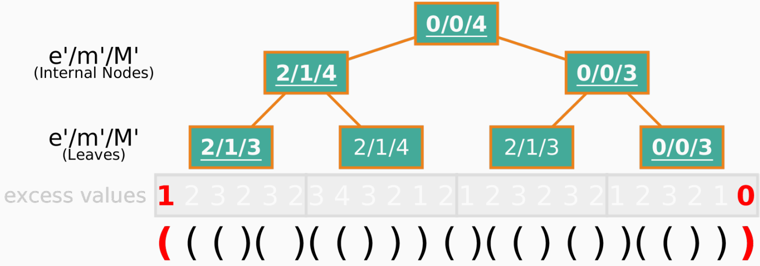
[†] G. Navarro and K. Sadakane. *Fully-functional static and dynamic succinct tree.* ACM Trans. Algorithms. 2014.

Range min-max tree†



†G. Navarro and K. Sadakane. *Fully-functional static and dynamic succinct tree.* ACM Trans. Algorithms. 2014.

Range min-max tree†

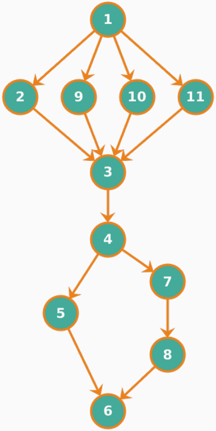


†G. Navarro and K. Sadakane. *Fully-functional static and dynamic succinct tree.* ACM Trans. Algorithms. 2014.

Range min-max tree: Observations

- The range min-max tree reduces a large set of operations on trees to a small set of primitives operations.
- The representation of the range min-max tree consists of four arrays e', m', M' and n'.
- e' does not store excess values of the internal nodes.
- In the range min-max tree siblings can be computed independently.

Parallel model: Dynamic multithreading model



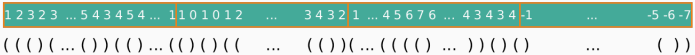
- Work (T_1)
- T_p
- Span (T_∞)
- Speedup (T_1/T_p)
- Parallelism (T_1/T_∞)

Parallel succinct tree algorithm

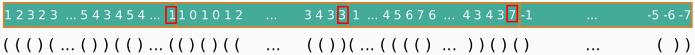
Parallel succinct tree algorithm

((( ) (... ( )) (( ) ... (( ) ( ) ((... (( ))(... (((( ) ...)) ( ) ( ) ... ( ))

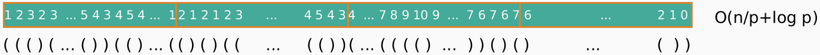
Parallel succinct tree algorithm



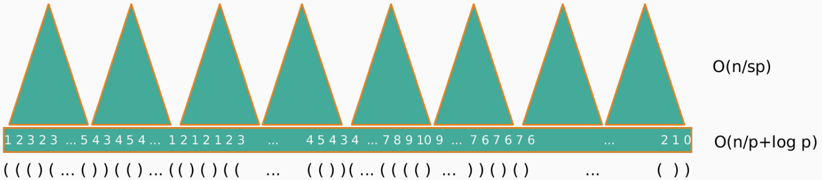
Parallel succinct tree algorithm



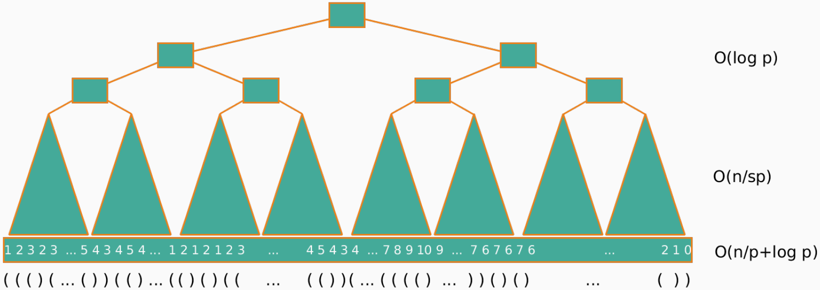
Parallel succinct tree algorithm



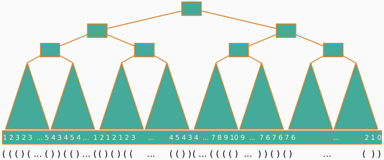
Parallel succinct tree algorithm



Parallel succinct tree algorithm



- $T_1 = O(n + \sqrt{2^w} poly(w))$
- $T_p = O(n/p + \lg p + \sqrt{2^w} poly(w)/p)$
- $T_\infty = O(\lg n)$
- $Speedup = O(\frac{p(n+\sqrt{2^w} poly(w))}{n+p \lg p+\sqrt{2^w} poly(w)})$
- $Parallelism = O(\frac{n+\sqrt{2^w} poly(w)}{\lg n})$



Parallel succinct tree algorithm: Theorem

A $(2n + o(n))$-bit representation of an ordinal tree on n nodes and its balanced parenthesis sequence can be computed in $O(n/p + \lg p)$ time using $O(n \lg n)$ bits of working space, where p is the number of cores. This representation can support the operations in $O(\lg n)$ time.

Experiments

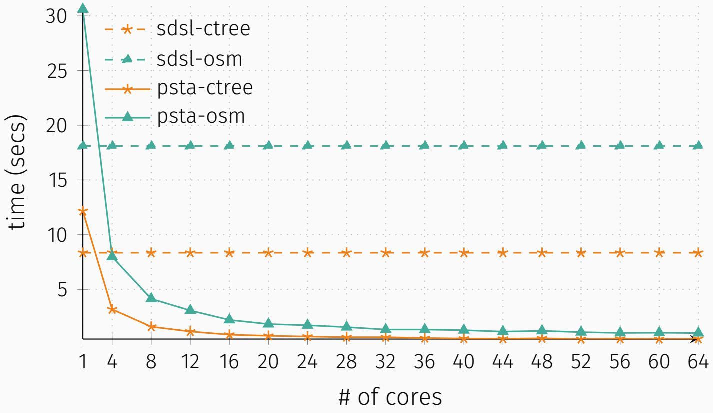
Compiler	GCC 4.9 (Cilk branch)
Baseline	SDSL and LibCDS
Machine	Four 16-core AMD Opteron™ 6278 processors, clocked at 2.4GHz. L1 of 64KB per core, L2 of 2MB shared by 2 cores, L3 of 6MB shared by 8 cores and 189GB of DDR3 RAM, clocked at 1333MHz

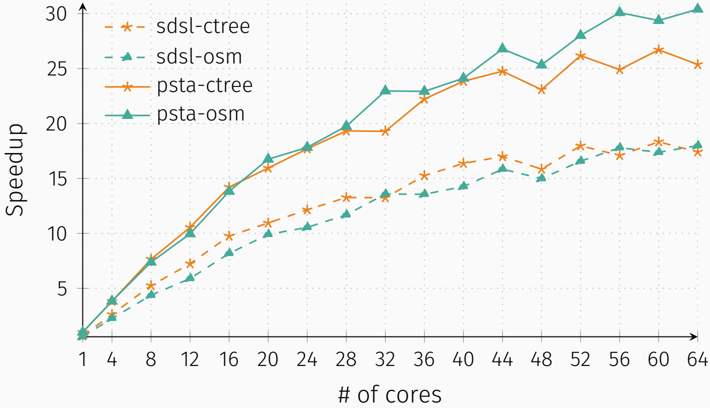
Evaluation

Datasets

Dataset	Number of parentheses
Wikipedia	498,753,914
Protein	670,721,006
DNA	1,154,482,174
Complete tree	2,147,483,644
Open Street map	4,675,776,358

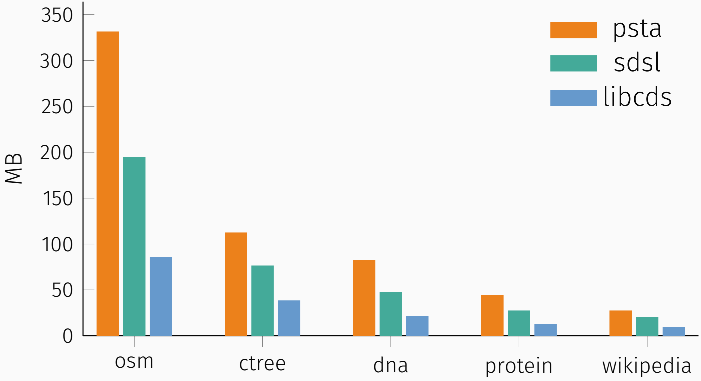
Results: time





30
25
20
15
10
5
Speedup
sdsl-ctree
sdsl-osm
psta-ctree
psta-osm
1
4
8
12
16
20
24
28
32
36
40
44
48
52
56
60
64
# of cores

Results: Memory consumption



Conclusions

Conclusions

- We introduce a $O(n/p + \lg p)$-time practical algorithm to construct a succinct representation of a tree with n nodes and p threads.
- The representation supports operations in $O(\lg n)$ time. The next step will be construct in parallel a representation that supports operations in $O(1)$ time.
- To use less memory in construction time we can reduce the number of bits per each elements in the arrays e', m', M' and n'.
- Our approach can be extended to the parallel construction of
  - Dynamic succinct trees
  - Succinct representation of labelled trees
  - Other succinct data structures that use succinct trees as building blocks, as for example, succinct representation of planar graphs.

Finally ...

Visit `http://www.inf.udec.cl/~josefuentes/sea2015` for datasets, code and more details.

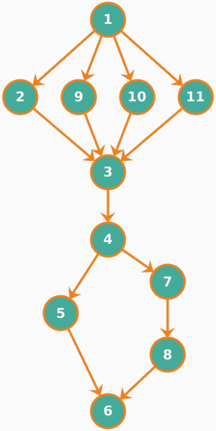
Parallel Construction of Succinct Trees

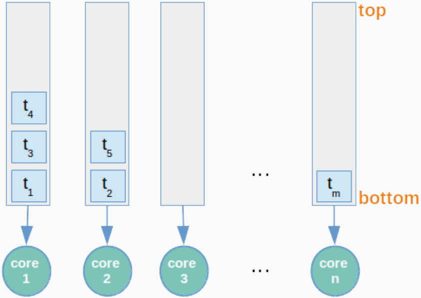
Leo Ferres[a], José Fuentes-Sepúlveda[a]
Meng He[b], Norbert Zeh[b]

June 29, 2015

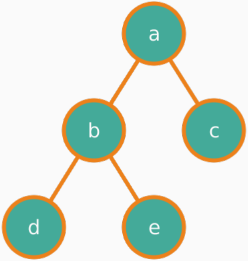
[a]University of Concepción, Chile
[b]Dalhousie University, Canada

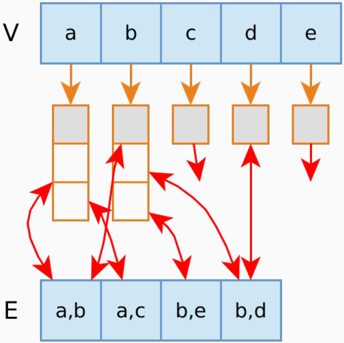
Parallel model: Dynamic multithreading model



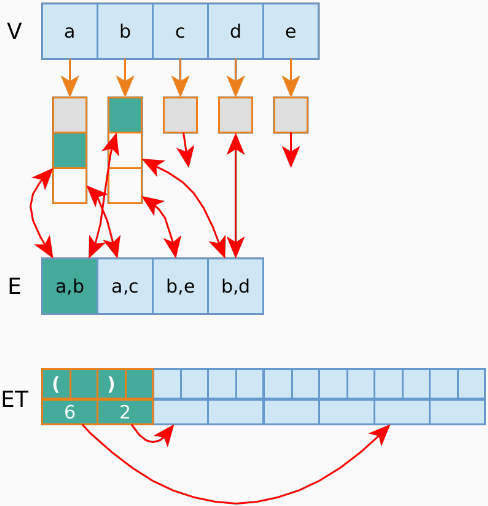


Parallel folklore encoding algorithm

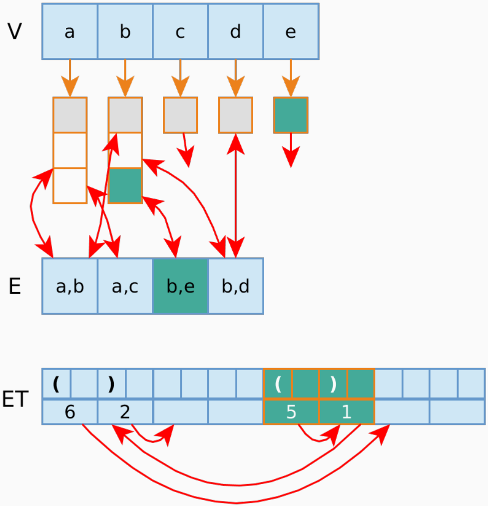




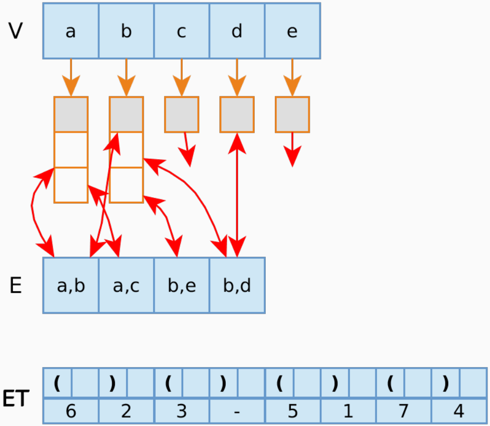
Parallel folklore encoding algorithm



Parallel folklore encoding algorithm



Parallel folklore encoding algorithm



Parallel folklore encoding algorithm