

# SuccinctX

Efficient RegEx Queries on Compressed Data

Anurag Khandelwal, Rachit Agarwal, Ion Stoica



## Regular Expressions

- **Example:**  $1-\backslash d\backslash d\backslash d-\backslash d\backslash d\backslash d-\backslash d\backslash d\backslash d$  (US Phones)
- **Wide range of applications:** text and document stores, bioinformatics, data mining, etc.

## Existing Techniques

- Full Data Scans (NFA, DFA)
  - Do not scale with data size
  - E.g., MongoDB, Oracle, MySQL, etc
- $m$ -gram Indexes with Partial Data Scans
  - Index token of length  $m$  for multiple or all values of  $m$
  - Avoid scans for indexed tokens.
  - Partial scans for tokens not indexed.
  - Suffer from large memory footprint
  - E.g., Elasticsearch
- Succinct (Search on compressed data)
  - Memory-efficient search for arbitrary length tokens
  - Asymptotic search complexity similar to  $m$ -gram indexes
  - Can be used as a black box

## Black-box RegEx

- Represent RegEx as an RTree
  - Leaves are tokens; Interior nodes are operators
- Search for leaves
- Traverse the RTree bottom-up, combining intermediate results at each node

$(Y_0 | H_0) (H_0^+)$

Operator	Complexity	Optimal?
Union	$O(s)$	Yes
Repeat	$O(s)$	Yes
Wildcard	$O(s \log n)$	Yes
Concat	$O(n + m)$	No

$n, m$ : input sizes (intermediate results)  
 $s$ : output size

Output cardinality for Concat operator can be arbitrarily smaller than cardinality of intermediate results

- Unnecessary operations if the gap is large

Complexity of Union, Repeat, and Wildcard operators depend on output cardinality

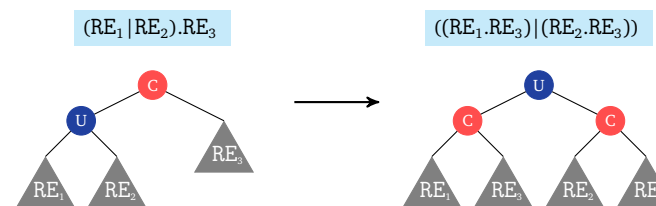
- Output cardinality smaller up the RTree

## Succinct

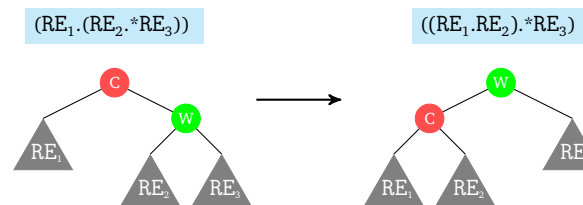
**Main Idea:** Transform the RTree for the RegEx such that:

- the black-box approach can be avoided for the Concat operator altogether
- Union, Repeat and Wildcard operators are pushed up the RTree

### Pull-Up Union Transformation:



### Pull-Up Wildcard Transformation:



### Pull-Out Repeat Transformation:

Replace Repeat operator by Unions of Concat:

$$RE^+ = (RE^1 | RE^2 | RE^3 | \dots | RE^k)$$

$k$ : smallest integer for which  $RE^{k+1}$  has 0 occurrences.

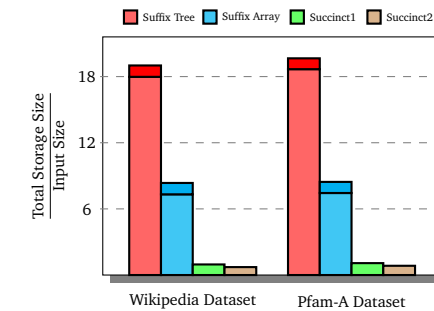
- Use heuristic to upper bound value of  $k$
- $k$  can be large when RegEx contains character classes
- Use partial scans beyond threshold

### Pull-Out Concat Transformation:

- Find Concat nodes whose children are tokens ( $T_1, T_2$ )
- Replace with new token  $T_1 T_2$

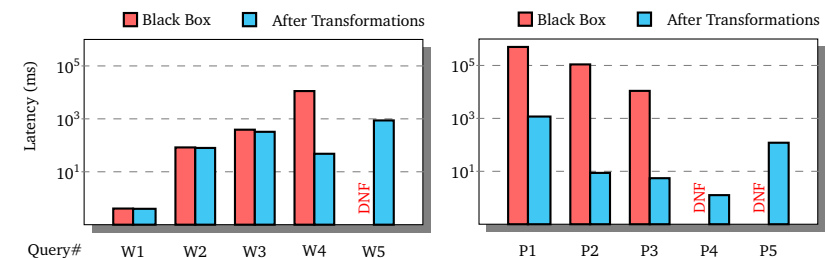
Transformations incorporated within Succinct data structures.

## Results

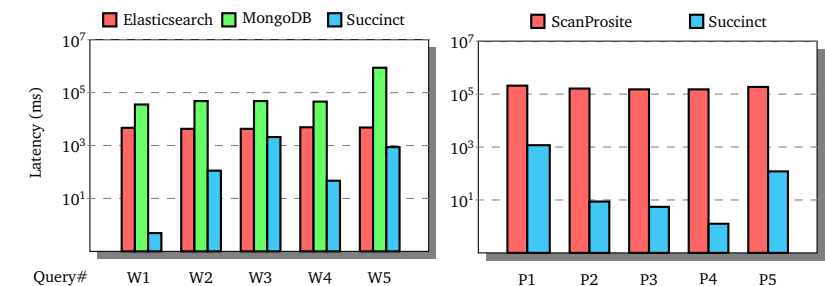


Succinct has 8x smaller storage footprint than uncompressed data structures

	Query ID	Query	Description
Wikipedia	Query#W1	<script>.*</script>	HTML Scripts
	Query#W2	Motorola.*(XPC MPC)([0-9])+([0-9a-z])*	Motorola PowerPC Chip#
	Query#W3	William [A-Z]([a-z])+ Clinton	Bill Clinton's middle name
	Query#W4	1-\d\d\d-\d\d\d-\d\d\d\d	US Phone Numbers
	Query#W5	([a-z0-9 \.])+([a-z0-9])+\.stanford\.edu	Stanford domain URLs.
Pfam-A	Query#P1	[DE]GSW[GE].W[GA][LIVM].[FY].Y[GA]	TERPENE_SYNTHASES
	Query#P2	[AC]GL.FPV	HISTONE_H2A
	Query#P3	GKPKLK.TC	CLUSTERIN_1
	Query#P4	G[MV]ALFCGCGH	MYELIN_PLP_1
	Query#P5	[FYW]P[GS]N[LIVM]R[EQ]L.[NHAT]	SIGMA54_INTERACT_3



Black-box, with and without transformations



Succinct versus {ElasticSearch, MongoDB, ScanProsite}

## Open Source Release

- <https://github.com/amplab/succinct-cpp>
- For questions & feedback, contact us at:

{anuragk, ragarwal, istoica}@berkeley.edu