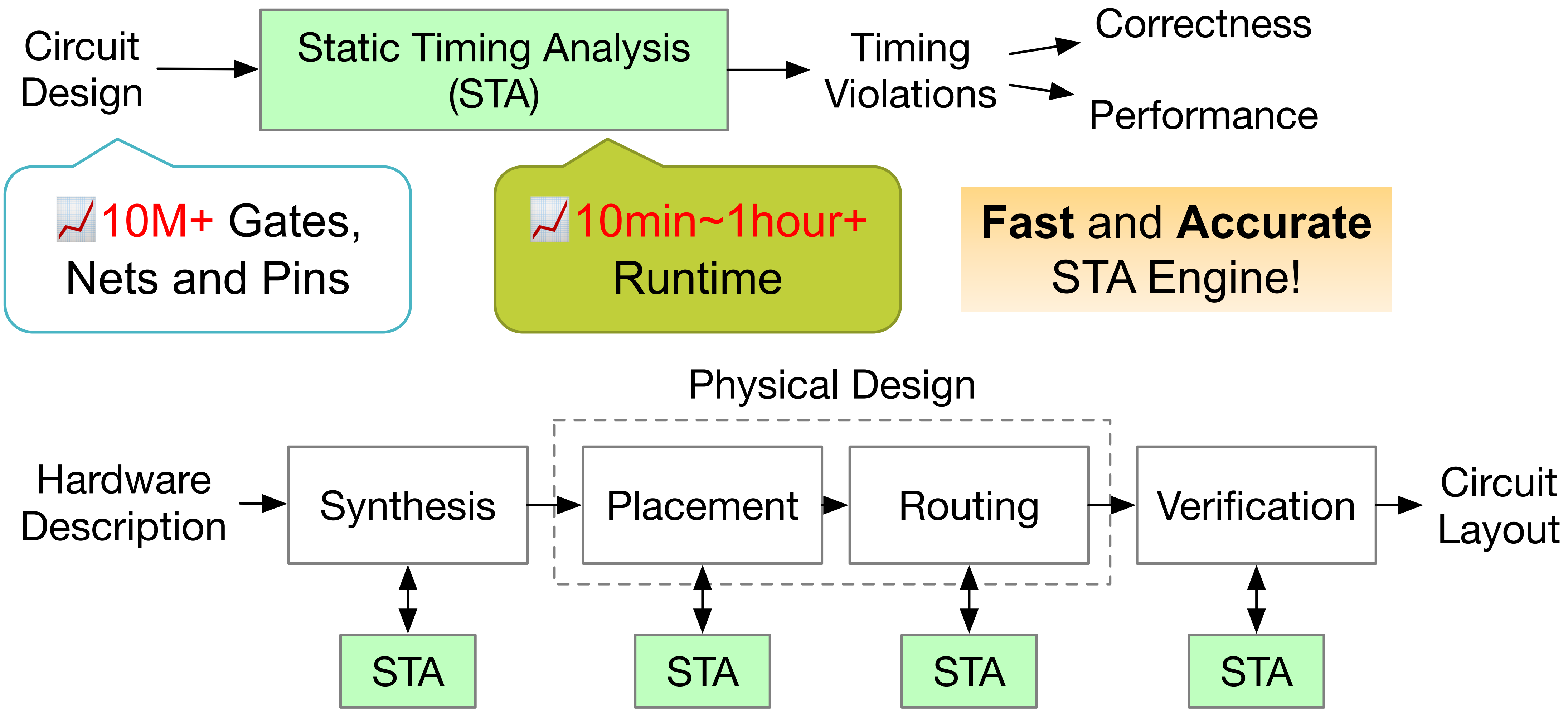
Efficient Critical Paths Search Algorithm using Mergeable Heap

Kexing Zhou*[1], **Zizheng Guo***[1], Tsung-Wei Huang[2], Yibo Lin[1]

[1]CS Department, Peking University; [2]ECE Department, University of Utah

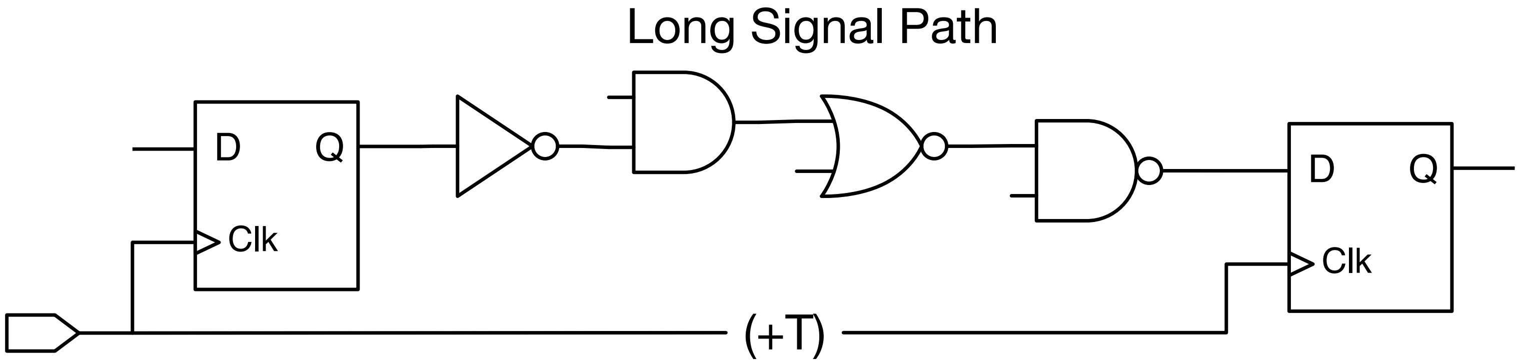
gzz@pku.edu.cn https://guozz.cn

Static Timing Analysis (STA)



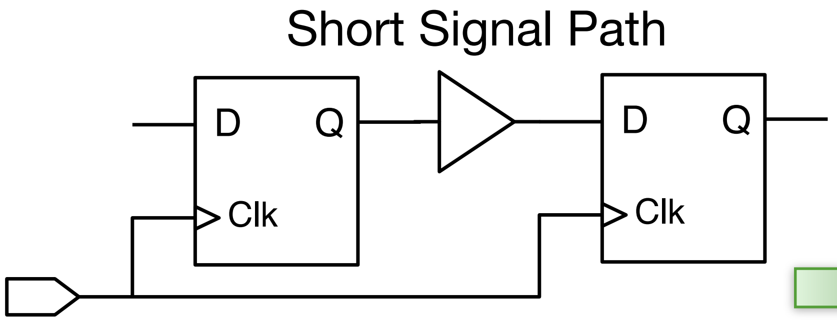
Critical Paths Searching in STA

Setup Violation



Top-*k* **longest** paths

Hold Violation



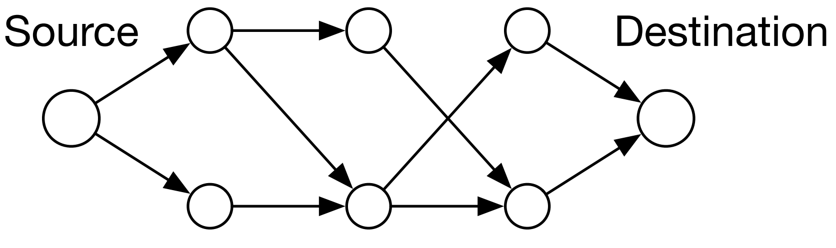
Top-*k* **shortest** paths

Input: Circuit graph, *k*
Output: Top-k shortest paths

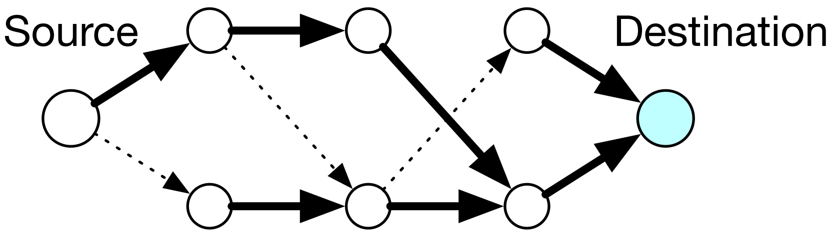
Real case STA: *k* = 10,000 ~ 100,000

The State-of-the-art *K*-Shortest Path Algorithm

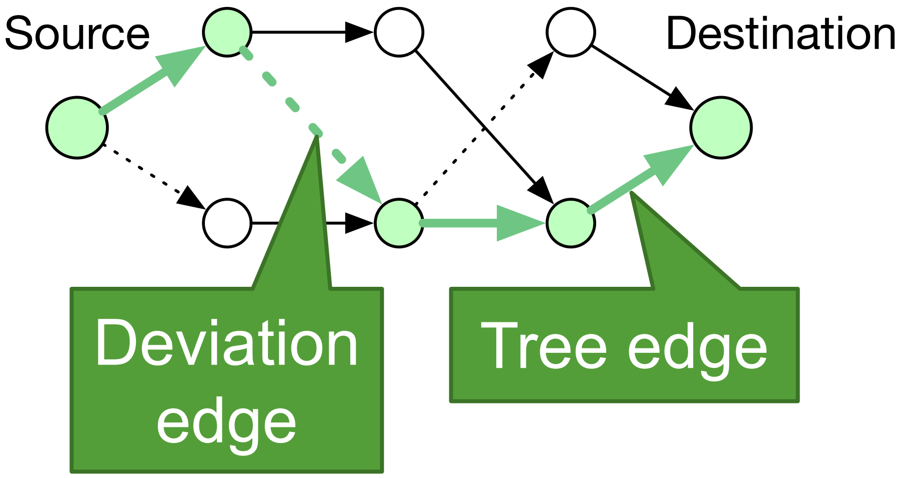
Suffix Tree - Prefix Tree Algorithm [OpenTimer, TCAD'20] [Guo, ICCAD'21]

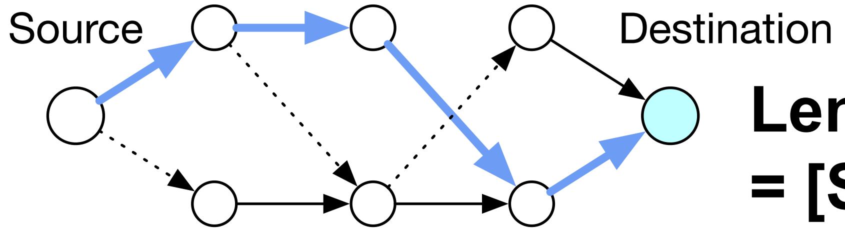


Circuit Graph

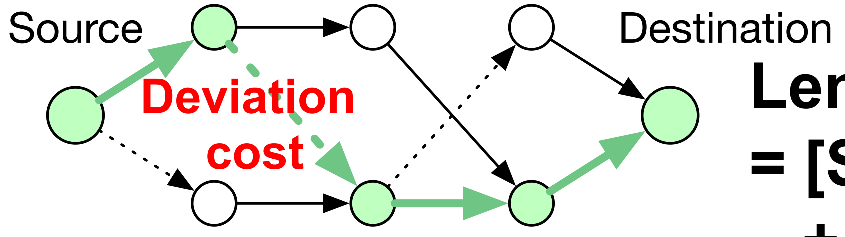


Suffix Tree

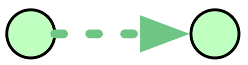




Length = [Shortest path len]

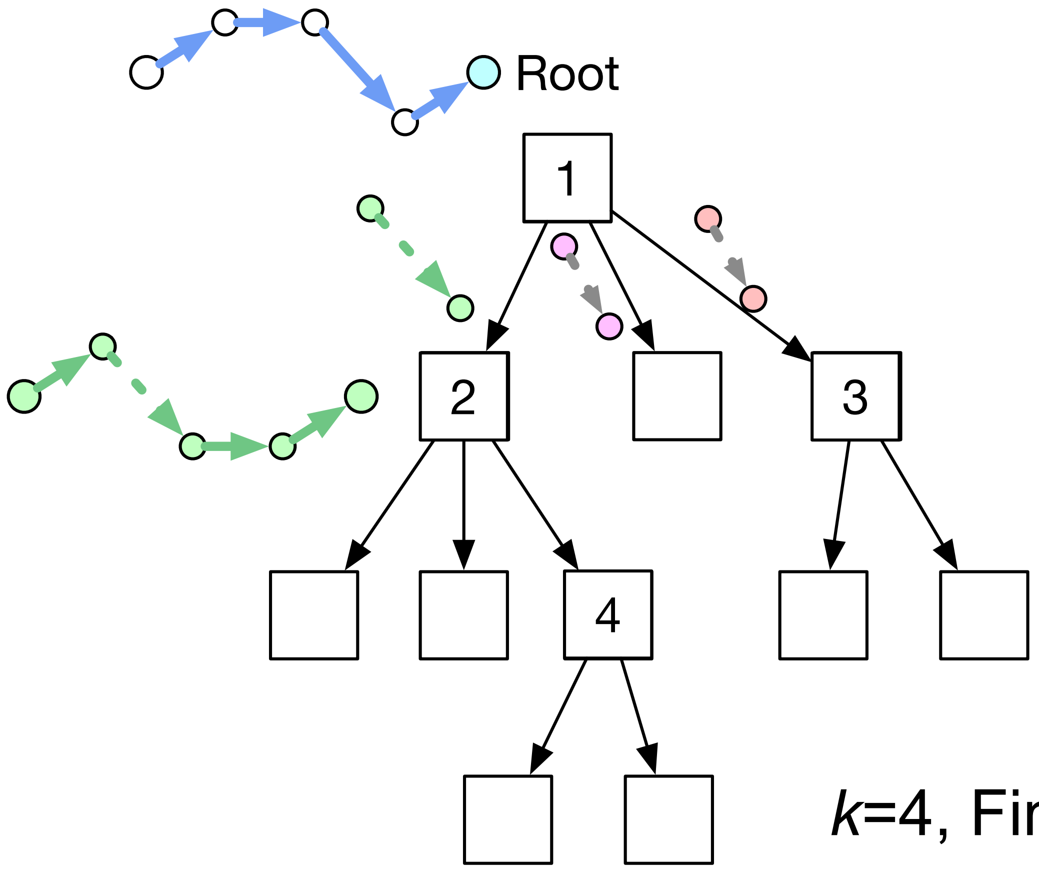


Length = [Shortest path len] + [Deviation cost]

Path ⬄ The set of deviation edges on the path {  , … }

The State-of-the-art *K*-Shortest Path Algorithm

Suffix Tree - Prefix Tree Algorithm [OpenTimer, TCAD'20] [Guo, ICCAD'21]



⬅️ Prefix Tree

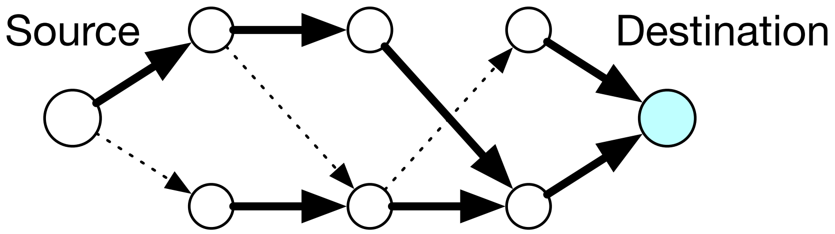
Node ⇔ *k*-Shortest Paths

Edge ⇔ Deviation Edges

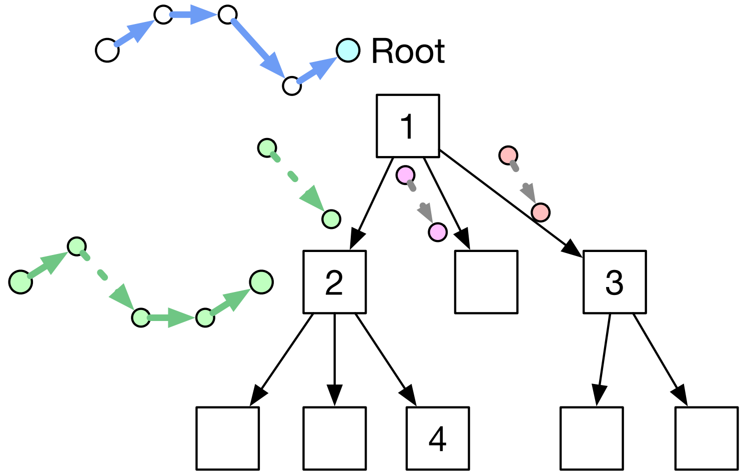
k=4, Finished

The Burden for Suffix-Prefix Algorithm

- n = #Pins + #Edges, k = #Paths
- Step 1: Build the suffix tree



- Step 2: Search for top-k paths

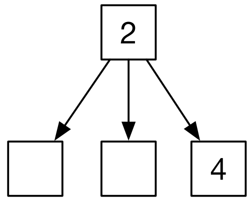


Time Complexity

$O(n)$
on acyclic graph

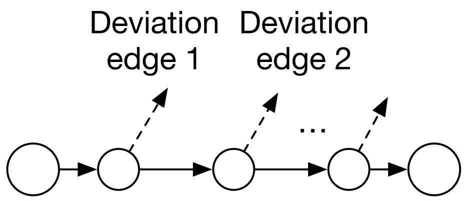
Expected:
$\sim O(k+n)$
This work

Total:
$O(kn \log n)$
Large?
Large?

k * 

$= O(kn \log n)$

Each exploration:
at most $O(n)$ deviation edges

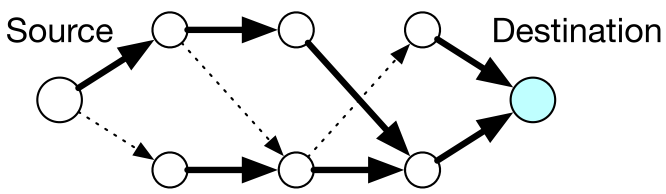


Our Contributions

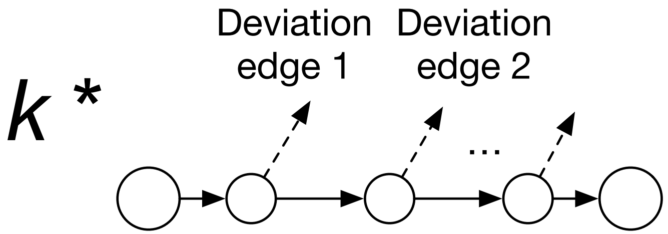
- A novel *k*-shortest path searching algorithm that runs in **$O(n\log n + k\log k)$**, asymptotically lower than baselines $O(kn\log n)$.
- Incorporating **persistent mergeable heaps** to store all path deviations for fast merging and duplicating.
- Introducing a novel **deviation preprocessing** step to precompute path deviations and speed up path searching.

Motivation: Pre-computing for Future Use

Step 1: Suffix tree



Step 2: Prefix tree

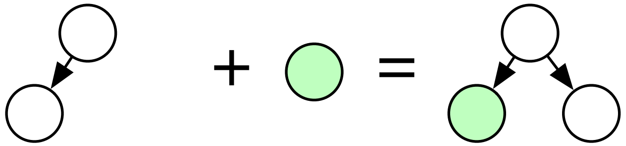


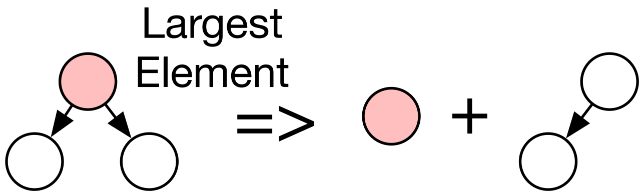
Each exploration: at most O(n) deviation edges

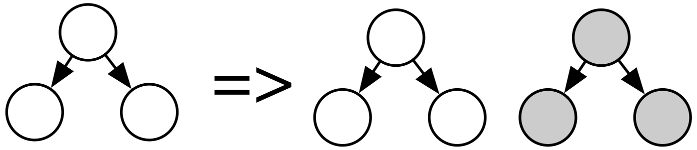
	Step 1: Suffix tree	Step 2: Prefix tree
Baseline	O(n)	= O(kn log$n$)
Our Algorithm	Prepare deviation edges O($n$log$n$)	Use precomputed edges O($k$log$k$)

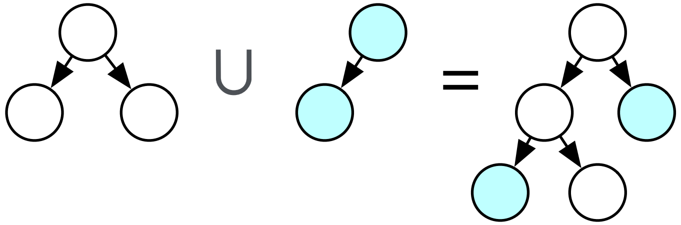
Overall: O($n$log$n$+$k$log$k$)

The Building Block: Persistent Mergeable Heaps





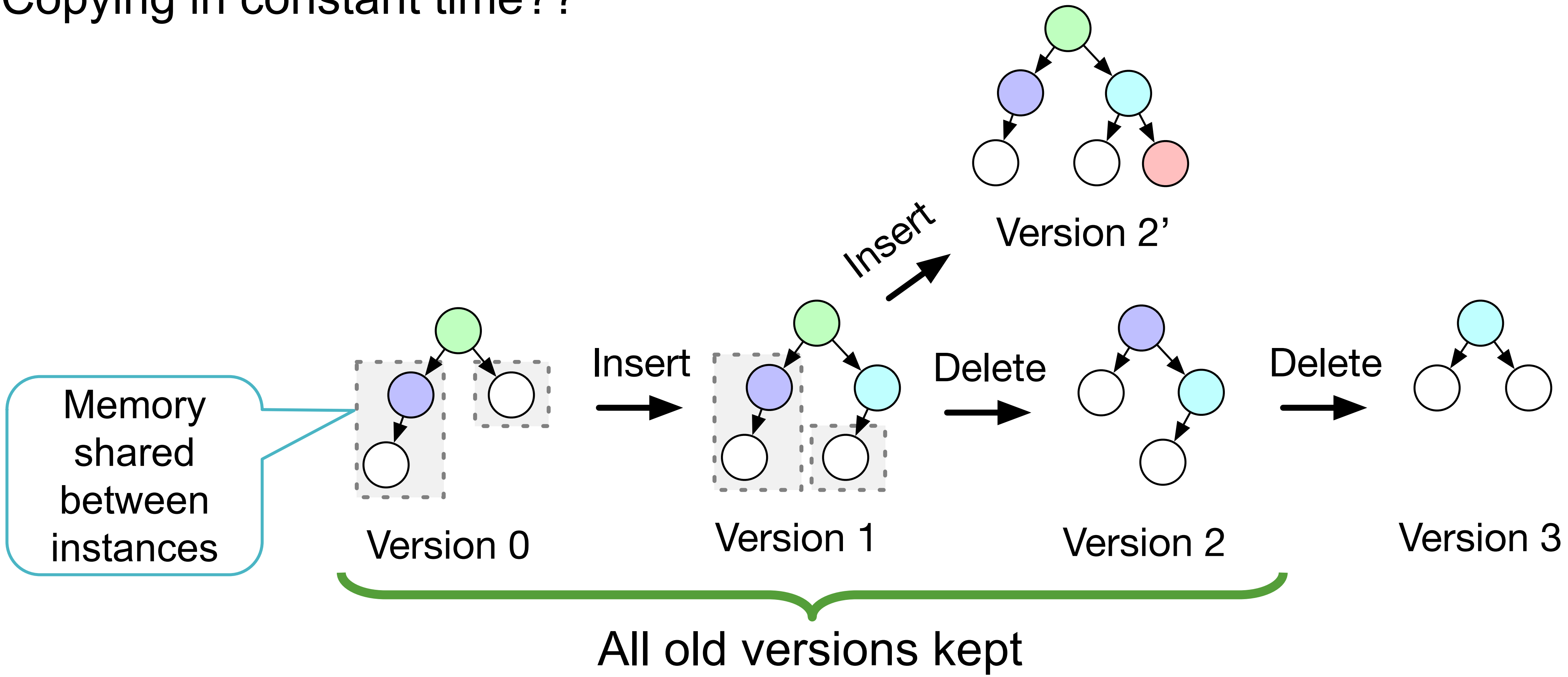




	Ordinary Heaps (e.g. Binary heap)	**Persistent Mergeable Heaps** (e.g. Leftist tree)
Insertion	O(log*n*)	O(log*n*)
Deletion (largest element)	O(log*n*)	O(log*n*)
Copying	O(*n*)	O(1)
Merging	O(*n*)	O(log*n*)

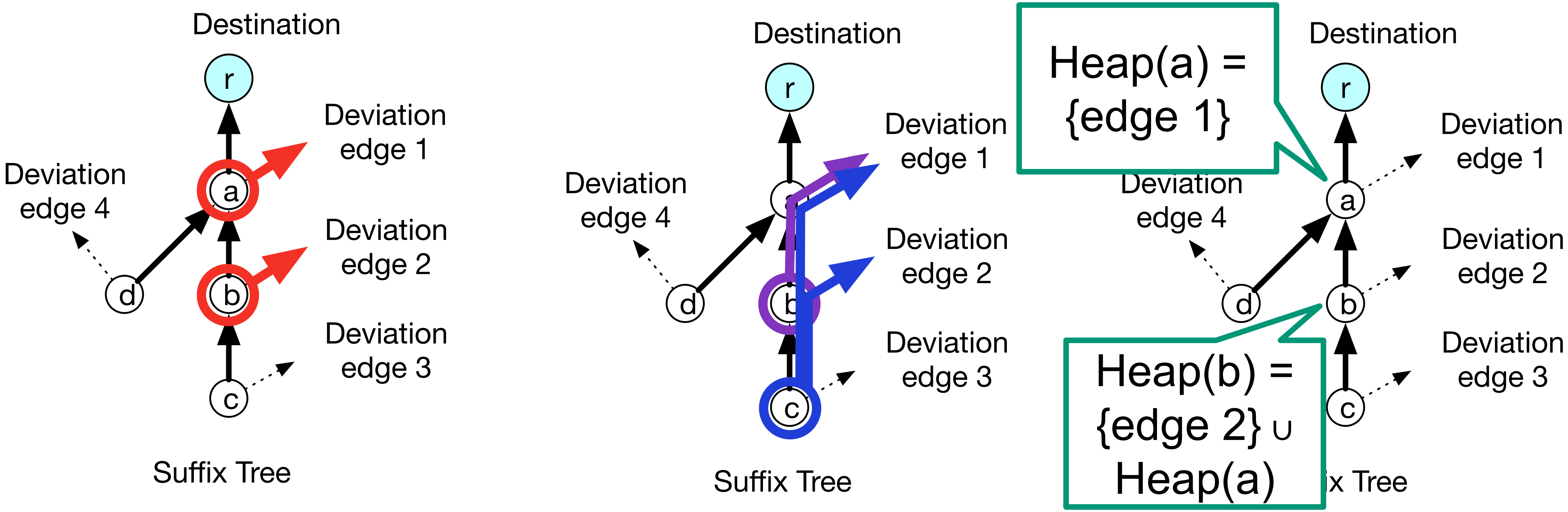
Magic Behind Leftist-Tree: **Persistency**

Copying in constant time??



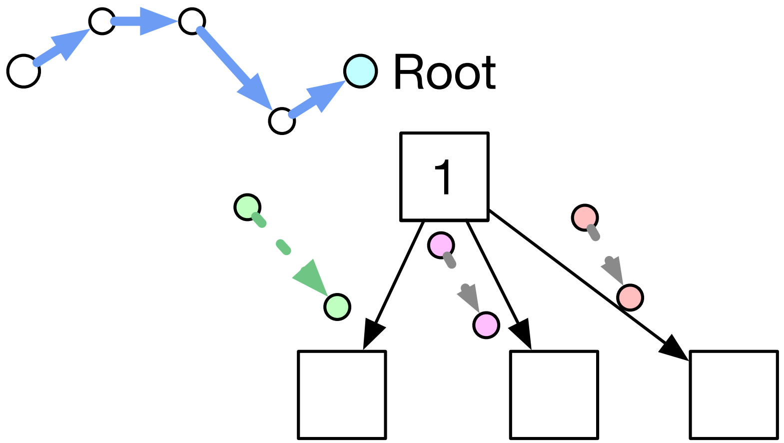
Copy data structure ⇔ doing *NOTHING* (only keep the old version pointer), O(1)

Our Algorithm (1/2): Deviation Preprocessing

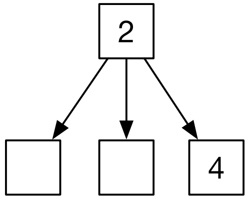


BFS on suffix tree and build up heaps => O(*n*log*n*)

Our Algorithm (2/2): Efficient Path Searching



Overall: $O(n\log n + k\log k)$

k^* 

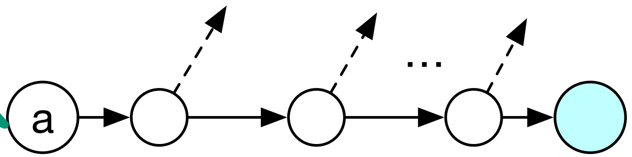
$= O(k\log k)$

Preprocessing: $O(n\log n)$

Heap of unexplored nodes { □, □, □ }

Each exploration: $O(n)$ deviation edges, **but only 1 preprocessed heap**

Heap(a) = {edge 1, edge 2, …}



Merge! $O(\log k)$

Experimental Results

- Implementation on OpenTimer
- 2.1GHz Intel Xeon & 512GB mem
- Compared with:
  - OpenTimer
  - CPU version of the *suffix forest* algorithm [Guo, ICCAD'21]
- On: TAU 2015 benchmarks

On benchmark `leon2`: 4M pins Time (ms)

k=	100	100,000	1,000,000
Ours	1481	**5476**	**19472**
OpenTimer	4469	218463	1009337
Speed-up	**3x**	**39x**	**51x**
Suffix forest	**1441**	6834	34572
Speed-up	0.97x	**1.24x**	**1.77x**

Conclusions and Future Work

- Near linear-time $O(n\log n + k\log k)$ novel *k*-shortest path searching algorithm
- **persistent mergeable heaps** and **deviation preprocessing** step to precompute path deviations and speed up path searching.
- **1.7~50x** faster than OpenTimer and other baselines.

- Path constraints
- GPU acceleration
- Common path pessimism removal (CPPR)

Thanks!
Questions are welcome

Website: https://guozz.cn
Email: gzz@pku.edu.cn