Pointer-based Data Structures (5/5)

Elena Arseneva

PDMI CS Club,
Sep-Oct 2019

Persistence

	Overhead	Source (see next page)
Partial (read-only access to previous versions)	$O(1)$	[1,2]
Full (read/write access to previous versions)	$O(1)$	[1] deamortization: OPEN
Confluent (full+merging versions)	$\log$(# updates)+ $\log$(#distinct paths)	[3-6]
Functional (confluent+ unchangeable nodes)	no general technique known	[7,8]

Assumptions of the original DS:

- pointer-based
- constant # fields per node
- constant in-degree

— worst-case time

— amortized worst-case time

Reading on Persistence

- Erik Demaine's lecture notes (and video): lec 1
- (order maintenance) P. Dietz, D. Sleator: Two Algorithms for Maintaining Order in a List STOC 1987: 365-372

1 J. Driscoll, N. Sarnak, D. Sleator, R. Tarjan: Making Data Structures Persistent. J. Comput. Syst. Sci. 38(1): 86-124 (1989)

2 G. Brodal: Partially Persistent Data Structures of Bounded Degree with Constant Update Time. Nord. J. Comput. 3(3): 238-255 (1996)

3 A. Fiat, H. Kaplan: Making data structures confluently persistent. J. Algorithms 48(1): 16-58 (2003)

4 S. Collette, J. Iacono, S. Langerman. Confluent Persistence Revisited. In Symposium on Discrete Algorithms (SODA), 593-601, 2012.

5 H. Kaplan, C. Okasaki, R. Tarjan: Simple Confluently Persistent Catenable Lists. SIAM J. Comput. 30(3): 965-977 (2000)

6 E. Demaine, S. Langerman, and E. Price: Confluently Persistent Tries for Efficient Version Control. Algorithmica (2008).

7 C. Okasaki: Purely Functional Data Structures. NY: Camb. Pr., 2003.

8 G. Brodal, C. Makris, K. Tsichlas: Purely Functional Worst Case Constant Time Catenable Sorted Lists. ESA 2006: 172-183