Pointer-based Data Structures (1/5)

Elena Arseneva

PDMI CS Club,
Sep-Oct 2019

Course Info: Pointer-based Data Structures

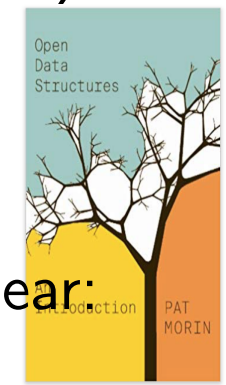
Schedule:

- Lec. 1: Intro, BSTs review, Scapegoat trees.
- Lec. 2: Splay trees, dynamic optimality.
- Lec. 3: Link-cut trees.
- Lec. 4 & Lec. 5: Dynamization of static pointer-based DSs. Persistence for dynamic pointer-based DSs.

Course Info: Pointer-based Data Structures

Some books:

- R. Cormen, C. Leiserson, R. Rivest, C. Stein "Intoduction to Algorithms, 3rd Edition"(The MIT Press) 2009.
- Pat Morin "Open Data structures"https://opendatastructures.org



Conferences where most work on data structures appear:

- Foundations of Computer Science (FOCS),
- Symposium On Discrete Algorithms (SODA),
- International Colloquium on Automata, Languages and Programming (ICALP),
- European Symposium on Algorithms (ESA),
- International Symposium on Algorithms and Computation (ISAAC),
- Symposium on Theoretical Aspects of Computer Science (STACS),
- Scandinavian Symposium and Workshops on Algorithm Theory (SWAT),
- Algorithms and Data Structures Symposium (WADS)

Dynamic Sets

	BST	RB-tree	Scapegoat tree	Splay tree
SEARCH(S,k)	$O(h)$	$O(\log n)$	$O(\log n)$	$O(\log n)$
MINIMUM(S)	$O(h)$	$O(\log n)$	$O(\log n)$	$O(\log n)$
MAXIMUM(S)	$O(h)$	$O(\log n)$	$O(\log n)$	$O(\log n)$
SUCCESSOR(S,x)	$O(h)$	$O(\log n)$	$O(\log n)$	$O(\log n)$
PREDECESSOR(S,x)	$O(h)$	$O(\log n)$	$O(\log n)$	$O(\log n)$
INSERT(S, x)	$O(h)$	$O(\log n)$	$O(\log n)$	$O(\log n)$
DELETE(S, x)	$O(h)$	$O(\log n)$	$O(\log n)$	$O(\log n)$

Lec. 1 & 2

— worst-case time

— amortized worst-case time

Exercises

- (An alternative in-order walk in BST) Prove that the following procedure in a BST of size n can be carried out in $\Theta(n)$ time: Find the minimum element in the tree, write it down, then $n-1$ times find the successor of the last element written and write the result down.
- How do we Join two BSTs? (Join(T_1, x, T_2) takes two BSTs, such that all the keys of T_1 are at most x, and all the keys of T_2 are greater than x, and returns a BST whose keys are union of the sets of keys of T_1 and T_2).
- How do we Join Red-Black Trees efficiently? (here both initial and final tree must be red-black)
- How do we Join Scapegoat trees efficiently?
- (*) Describe a Rebuild procedure (used in Scapegoat trees) that takes $o(n)$ space in addition to the space taken by the tree itself.

Reading on BSTs, RB trees, Scapegoat trees

- BSTs and Red-Black trees:
  - R. Cormen, C. Leiserson, R. Rivest, C. Stein "Intoduction to Algorithms, 3rd Ed."(The MIT Press) 2009 + Russian translation
  - original1: R. Bayer and E. M. McCreight. Organization and maintenance of large ordered indexes. Acta Informatica, 1(3):173–189, 1972.
  - original2: Leo J. Guibas and Robert Sedgewick. A dichromatic framework for balanced trees. In Proc. 19th Annual Symposium on Foundations of Computer Science, pages 8–21, 1978.
- Scapegoat trees:
  - Pat Morin "Open Data Structures"
  - Jeff Erikson's lecture notes: http://jeffe.cs.illinois.edu/teaching/datastructures/notes/02-scapegoat-splay.pdf
  - original1: Arne Andersson. General balanced trees. J. Algorithms 30:1–28, 1999.
  - original2: I. Galperin and R. Rivest. Scapegoat trees. Proc. 4th Annu. ACM-SIAM Sympos. Discrete Algorithms, 165–174, 1993.