Weekplan: Persistent Data Structures and the Plane Sweep Method

Inge Li Gørtz

References and Reading

[1] Topics in Data Structures, section 5.1 and 5.5, G. F. Italiano and R. Raman.

[2] Planar Point Location Using Persistent Search Trees, N. Sarnak and R. E. Tarjan, CACM, 1986.

[3] Making Data Structures Persistent, J. R. Driscoll, N. Sarnak, D. D. Sleator, R. E. Tarjan, JCSS, 1989.

We recommend reading [1] in detail before the lecture (excluding the part after Thm 5.16 up to Thm 5.17).

[2] provides background on planar point location, and [3] provides background on persistent data structures.

Exercises

1 Fat node method Show the data structures used when we implement partial persistence using the fat node method for the following example. We have an array of length 5. Initially, all values in the array are 0. Update(i,v) updates position i to value v. Show how the data structure looks after the following upates:

Update(2,5), Update(3,4), Update(2,3), Update(1,8), Update(2,2), Update(3,3), Update(1,1), Update(2,7).

2 Node copying method In this exercise we use the node copying method to make an unbalanced binary search tree partially persistent. Show how the partial persistent unbalanced binary search tree look after the following insertions: 3, 5, 7, 9, 11, 10, 12, 2.

3 Persistent hash tables Which technique would you use to construct a partially persistent hash table?

4 Partially persistent heaps Describe an implementation of a partially persistent binary heap. The heap should support the operations find-min, extract-min and insert.

4.1 A binary heap can be represented either with pointers as a tree or as an array. Describe how to make it partially persistent in each case.

4.2 What are the time and space complexities for each of the operations?

4.3 Suppose you insert n elements into the partially binary heap. What is the total time and space usage?

5 Path copying If you have a linked data structure where all modifications occur on a path from the root, then it is enough to copy the path where the changes occur to get a partially persistent version of the data structure. Explain how this would work in a binary search tree and give an example. What is the time slowdown and the extra space used?

6 Temporal databases You are working as a consultant for the company "*Boxes, Boxes and Boxes*", that sells boxes. They want a database containing information about all their boxes. Each box has an id and a price. They want to be able to update the database with insertions and deletions of boxes. The database should—in addition to the updates—support the following query:

- report(a, b, t): Return the id of all boxes with a price between a and b that was in the database at time t.

Give a data structure supporting the required updates and queries. Analyse the space and time complexity of your data structure.

7 Temporal databases 2 In this exercise you have a database with data of the following form: [id, production date, sales date, price].

The database is not updated but stores a large amount of data. It must efficiently support queries of the following form:

- *sum*(prodfrom, prodto, sold): Return the sum of the prices of all products that have production date between prodfrom and prodto and have sales date before sold.

Give a solution that uses space $O(n)$, preprocessing time $O(n \log n)$ and has query time $O(\log n + k)$, where n is the number of items in the database and k is the number of reported items.

* Change your solution to report the number of items with the right product date, i.e., support the query

- *reportNum*(prodfrom, prodto, sold): Return the number of products that have production date between prodfrom and prodto and have sales date before sold.

Your data structure must have query time $O(\log n)$. What are the best space and preprocessing times you can get?