Lecture We will talk about the programming paradigm *dynamic programming*, and see two applications of it: rod cutting and longest common subsequence.

You should read CLRS introduction to chapter 15 + section 15.1 and 15.4. It is also a good idea to read section 15.3, but part of the section is about matrix chain multiplication so you do not need to understand all details of this.

Exercises

MultiPush Solve CLRS 17.1-1.

Doubling arrays In this exercise we consider dynamic tables with insertions only (no deletions) using the doubling technique. Can we show that the amortized cost of an insertion is $O(1)$ using the following potential function: $\Phi(D_i) = k$, where k is the number of elements in the array?

Queue with two stacks Solve CLRS 17.3-6.

Splay trees

1. Show the splay tree that results from inserting the keys 41, 38, 31, 12, 19 , 8 in this order into an initially empty tree.
2. Show how the tree looks after deleting 31.

Play trees Professor Bille suggests a simpler version of Splay Trees that he calls Play trees. Play Trees are similar to Splay Trees but uses only single rotations when splaying.

1. What is the amortized cost of splay(x) if we only use single rotations? Analyze it with the same potential function as we used for Splay Trees.
2. What is the total (actual) cost of first n inserting elements with keys $1, 2, 3, \ldots, n$ in that order in a Play Tree and then searching for $1, 2, 3, \ldots, n$ (in that order)?
3. Professor Bille claims that the amortized cost of the operations insert, search and delete in Play trees is $O(\log n)$. "You just need to use a more sophisticated potential function" he says. Could he be correct?

Splay trees Let Φ be the potential function used to analyze splay trees. I.e. $\Phi = \sum_{v \in T} \text{rank}(v)$. Prove that the potential of a complete binary tree is $O(n)$ and that the potential of a rooted path is $O(n \log n)$.

Dynamic hashtable Explain how to make a dynamic hashtable with insertions using the doubling technique. Your solution should use $\Theta(n)$ space. What is the insertion time of your solution?

Deamortization of dynamic tables It is sometimes possible to *deamortize* data structures, i.e., getting the same bounds worst case as amortized by doing the work in the "background". Show that you can get worst case constant insertion time in dynamic tables, while still having space usage $O(n)$, where n is the number of elements currently in the table.

Hint. Use the doubling technique, but spread out the work on all the insertions.

Puzzle of the week: Princesses You are a young Prince from the country Algo. The King in the neighboring country Logic has 3 daughters. The oldest one always tells the truth, the youngest one always lies, and the middle one sometimes lies, sometimes tells the truth.

You want to marry either the oldest one or the youngest one (since you know she *always* lies that is as good as the one always telling the truth). The only one you don't want to marry is the middle one.

The king is a sneaky man, and he tells you, that you can ask *one* of the daughters *one* question. The question should be one with a yes/no answer. After that you have to choose which one to marry. They all look alike, so it is not possible for you to determine which one is which by looking at them.

What question should you ask, and which one should you then pick?

Implementation of dynamic tables [CJ] Implement your own dynamic table for integers (without using the built-in versions). Your dynamic table must support insertion, deletion, printing of inserted elements and reporting of the table size.

Mandatory exercise: Binary heap Solve CLRS 17.3-3.