Tutorial 3

Data Structures 1, 7. 3. 2025 https://iuuk.mff.cuni.cz/~chmel/2425/ds1en/

Leftovers from last time

Exercise 1 (Flexible arrays have potential!)
In the lecture, you have seen the analysis of the doubling flexible arrays. There, you used the aggregate method to argue that doubling the array when full and halving the array when at the quarter of its capacity gives amortized constant time.
Prove the same result using a potential defined as $\phi(\text{array}) = |\frac{\text{capacity of array}}{2} - \text{number of elements in array}|$.

Splay trees

Exercise 2 (Splay trees might not have logarithmic depth)
Create a sequence of operations that builds a splay tree with linear depth that works for both naive and standard splay operation.

Exercise 3 (Why isn't naive splay enough?)
What breaks when the Splay operation is implemented naively, that is by only rotating the node up using only single rotations on the node? It is best if you construct a sequence of $m \geq n$ operations that has total cost larger than $m \log n$, but just saying what breaks in the proof of the amortized complexity also works.

Exercise 4 (Splay trees, potentially)
Ensure that you understand the definition of the potential in splay trees and that you understand the main ideas of the amortized analysis of the splay operation.

a) What is the definition of the potential of a splay tree?

b) What is the amortized cost of a rotation (a single and a double)?

c) What is the amortized cost of the full splay operation and how does it follow from the amortized costs of rotations?

d) What is the *real* cost of the splay operation (and what units do we measure it in)?

e) What is the potential of a perfectly balanced BST? (A reasonable upper and lower bound are sufficient, assume the tree has $n = 2^k - 1$ nodes.)

f) What is the potential of a path? (Again, reasonable bounds are sufficient.)

Bonus exercises

Exercise 5 (Statically optimal BST)
We have a set of n keys $\{k_1, \ldots, k_n\}$ from which we want to build a static BST. At the same time, we know the probabilities of accessing each key $w_1, \ldots, w_n$ with $w_i > 0$ for all i.
Create an algorithm for building a statically optimal BST, that is a BST that minimizes the sum $\sum_{i=1}^{n} d(i) \cdot w_i$, where $d(i)$ is the depth of the key k_i and the root has depth 1. (That is, we are looking for such a tree such that the expected length of accessing a key is as small as possible.)
Hint: dynamic programming might be useful.

Exercise 6 (Analysing the successor using a potential)
During the first exercise, we showed that using $n - 1$ successor operations on any BST, when starting in the node with minimum key, has total cost $\mathcal{O}(n)$. Prove the same result using a potential.