Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Department of Computer Science
Markus Püschel
Peter Widmayer
Thomas Tschager
Tobias Pröger
Tomáš Gavenčiak

10rd November 2016

Data Structures & Algorithm Solutions to Sheet P7 AS 16

Solution P7.1 *Presents and ribbons.*

First, having to match ribbon lengths and box side areas is just an additional complication to the core matching problem, since for every ribbon of length l, we can compute the only matching area as $b = (l/4)^2$ if l is divisible by 4, and otherwise we know there can be no such box with a natural number as its side area. Even after this preprocessing of ribbon lengths, the remaining m' "ribbon areas" $b_0 \leq b_1 \leq \ldots b_{m'-1}$ are still sorted in ascending order.

We now have to solve the problem of matching the values in two sorted arrays A and B by equality. We can use an algorithm very similar to `merge(A,B)` from mergesort: We keep one index i into array A and index j into array B such that all numbers $a_0 \ldots a_{i-1}$ and $b_0 \ldots b_{j-1}$ have been either already matched or it is sure they do not have any match.

At every step, compare a_i to b_j. If $a_i = b_j$, we have a new matching pair and we advance both i and j by 1 (as neither can be used in another match). Otherwise if $a_i < b_j$, then all $b_j \ldots b_{m'-1}$ are larger than a_i and therefore a_i does not have a match, so we advance i by 1. Symetrically we advance j by 1 if $a_i > b_j$.

We start the loop with $i = j = 0$ and finish when $i = n$ or $j = m'$, as then we ran out of ribbons or boxes (not necessarily at the same time, of course).

On the lecture website, you can find a solution running in time $O(n+m)$ that does the lenght-to-matching-area transformation (skipping ribbons with length not divisible by 4) directly within the main loop. The solution source contains further comments on the implementation.

Data

`judge1` $m = 0$, $n = 1$, a special case with no ribbons.

`judge2` $m = 10\,000$, $n = 20\,000$, random numbers with extra random matches and repetitions.

`judge3` $m = 30\,000$, $n = 30\,000$, random numbers with extra random matches and repetitions.

Notes on submitted solutions. Apart from programs using some slow search, there were several other problems: Some people used `double` values in stead of `int`, but floating point calculations might lose precision for higher values, and there might be rounding errors. In this case, this was easily avoidable. Another problem was dividing the ribbon length by 4, as this gives `12/4=13/4=3`, while ribbon of length 13 can never match. Note that using the `double` would not really help, as you need to somehow round it afterwards.