Eidgenössische
Technische Hochschule
Zürich

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

Markus Püschel
Peter Widmayer
Thomas Tschager
Tobias Pröger
Tomáš Gavenčiak

Department of Computer Science
11th January 2017

Data Structures & Algorithm Solutions to the Test exam AS 16

Solution 1 *Minheap implementation.*

The task was quite straightforward and we refer the students to the solution sources and lecture notes on heaps for details. The main advantage of the heap nodes being numbered from 1 (rather than 0) is that the children of the node at position i are the nodes at positions $2i$ and $2i + 1$, and the parent of the node j is at position $j/2$ (integer division rounding down).

Solution program On the lecture website, you can find an example solution source with further comments on the implementation.

Data The judge data were mostly random and tested the correctness of the heap functionality. Program speed was not very important in this task.

Solution 2 *Maximal submatrix.*

We describe the fastest solution with $\mathcal{O}(n^3)$ running time that was hinted in the task text: Trying all possible rows for `top` and bottom is straightforward and there are $\mathcal{O}(n^2)$ such choices. For every one of those choices, we need to compute the sums of the partial columns C_j where

$$C_j = C_{\texttt{top:bottom},j} = \sum_{i=\texttt{top}}^{\texttt{bottom}} A_{i,j}.$$

If we would recompute all n values of C_j from scratch, this would take $\mathcal{O}(n^2)$ time for every choice of `top` and `bottom`, or $\mathcal{O}(n^4)$ time in total. We can reuse the old values of C_j we computed for `top` and `bottom` $- 1$ when we move to `top` and `bottom`:

$$C_{\texttt{top:bottom},j} = C_{\texttt{top:bottom-1},j} + A_{\texttt{bottom},j}.$$

This update takes only $\mathcal{O}(n)$ time for every choice of `top` and `bottom`.

When we have the partial column sums C_j, we need to find the maximal subarray sum on C_j to decide which column range to take as the optimal submatrix with chosen `top` and `bottom`. This is exactly the task P4.2 and was also covered in the lecture notes.

Solution programs On the lecture website, you can find two example solution sources with further comments on the implementation.

The "`slower`" solution program implements the hinted $\mathcal{O}(n^4)$ solution that precomputes the areas of all the submatrices starting at $(0, 0)$. It then tries all the $\mathcal{O}(n^4)$ submatrices, calculating the sum of each submatrix in time $\mathcal{O}(1)$ using the precomputed sums.

Data The judge data were mostly large random matrices, balancing the positive and negative numbers to get large submatrices as the optimal solution (but not the entire input).