Algorithms and Data Structures Week 14

Georg Hasebe, 17th December.

Matrices and Graphs

In section **2.2.3 Erreichbarkeit** of part 9 of the script, it is explained how we can determine whether it is possible to reach every vertex from every other vertex in a directed graph G.

The solution to this question is based on the matrix A_G^{n-1} for a certain matrix A (see script for details). At this point, we asked ourselves: how can we efficiently raise a matrix to the power of $n-1$?

The naive approach would involve performing $n-2$ matrix multiplications:

$$A_G \cdot A_G \cdots A_G = A_G^{n-1}.$$

If each matrix multiplication takes $O(n^3)$ time, this approach requires $O(n^4)$ time in total.

A more efficient method is **exponentiation by squaring**, which allows us to reduce the number of matrix multiplications:

$$A_G^k = \begin{cases} A_G^{k/2} \cdot A_G^{k/2} & \text{if } k \text{ is even}, \\ A_G^{(k-1)/2} \cdot A_G^{(k-1)/2} \cdot A_G & \text{if } k \text{ is odd}. \end{cases}$$

This method reduces the runtime to $O(n^3 \log n)$, again assuming each matrix multiplication operation takes $O(n^3)$ time.

To illustrate the difference, consider the case $n = 4$:

$$A^4 = (((A \cdot A) \cdot A) \cdot A) \qquad (\text{Naive method}) \quad (1)$$
$$A^4 = (A \cdot A) \cdot (A \cdot A) \qquad (\text{Exponentiation by squaring}) \quad (2)$$

Notice that this rearrangement of parentheses requires the **associativity** property of matrix multiplication, which clearly holds since A is a square $n \times n$ matrix.

Next, consider the recursive formulas from section **2.1.1 Rekursionen**:

$$L_{i,j}^{(k)} = \bigvee_{s=1}^{n} \left(L_{i,s}^{(k-1)} \wedge L_{s,j}^{(1)} \right),$$
$$M_{i,j}^{(k)} = \min_{s=1}^{n} \left(M_{i,s}^{(k-1)} + M_{s,j}^{(1)} \right),$$
$$N_{i,j}^{(k)} = \sum_{s=1}^{n} \left(N_{i,s}^{(k-1)} \cdot N_{s,j}^{(1)} \right).$$

To analyze these formulas, recall **Definition 5.18** of a **ring** from the *Diskrete Mathematik* script. A **semiring** is a generalization of rings, where we drop the requirement that every element must have an additive inverse. More precisely, a semiring $\langle R; +, 0, \cdot, 1 \rangle$ is an algebra where $\langle R; +, 0 \rangle$ is a commutative **monoid** rather than a commutative group.

From **Definition 5.5** of the *Diskrete Mathematik* script, a **monoid** is an algebra $\langle M; *, e \rangle$ where the operation $*$ is **associative**.

Now observe that the pairs of operations $(\vee, \wedge)$, $(\min, +)$, and $(+, \cdot)$ each form a semiring. Since semirings inherently satisfy the associativity property for multiplication, the recursive formulas $L_{i,j}^{(k)}$, $M_{i,j}^{(k)}$, and $N_{i,j}^{(k)}$ can be computed efficiently using **exponentiation by squaring**.

Thus, we conclude that $L_{i,j}^{(n)}$, $M_{i,j}^{(n)}$, and $N_{i,j}^{(n)}$ can all be computed in $O(n^3 \log n)$.

If you don't believe this, just compute A^3 for a 2×2 matrix and see if $(A \cdot A) \cdot A = A \cdot (A \cdot A)$ using any recursive formula of your choosing.