Exercise 2

Task 1
Let x and y be natural numbers in binary representation. Decide in NC, whether $x < y$.

Solution
Let $x = x_1 x_2 \cdots x_n$ and $y = y_1 y_2 \cdots y_n$ with $x_i, y_i \in \{0,1\}$. Question: Does $x < y$ hold?
Initiate n processors $p_1, \ldots, p_n$. Each p_i compares the bits x_i and y_i. If $x_i < y_i$, then p_i starts $i-1$ new processors $p_{i,1}, \ldots, p_{i,i-1}$. In the next step, $p_{i,j}$ tests if $x_j = y_j$. If there is one i (more precisely the smallest one), where all $p_{i.j}$ ($j = 1, \ldots, i-1$) answered *yes*, then return *true*, otherwise return *false*.
We showed that we can solve the problem in constant time with less than n^2 processors and hence we found an NC algorithm.

Task 2
Let x and y be natural numbers in binary representation. Compute the subtraction $x - y$ in NC, where $x - y = 0$ if $x < y$.

Solution
Let $x = x_1 x_2 \cdots x_n$ and $y = y_1 y_2 \cdots y_n$ with $x_i, y_i \in \{0,1\}$. Task: Compute

$$x - y := \begin{cases} 0, & \text{if } x < y, \\ \mathrm{sub}(x,y), & \text{otherwise.} \end{cases}$$

First wie need to test, if $x < y$. This can be done in NC via Task 1. If yes, the output is 0. To compute the subtraction, we use a very nice trick: The *ones' complement*. With that we are able to turn the subtraction into an addition. And we know that addition is in NC (see lecture).
We will start a little bit more general and consider the subtraction of two numbers x and y ($x \geq y$) in base B, which means $x_i, y_i \in \{0, \ldots, B-1\}$.
Claim: We have $x - y = x + \bar{y} + 1$, where $\bar{y} = (B-1-y_1)(B-1-y_2)\cdots(B-1-y_n)$. Here, the sum is interpreted as a number modulo B^n. That means in particular $-y = B^n - y$. Hence

$$\begin{aligned} B^n - y &= B^n - \sum_{i=1}^{n} y_i \cdot B^{n-i} \\ &= 1 + (B-1) \cdot \sum_{i=1}^{n} B^{n-i} - \sum_{i=1}^{n} y_i \cdot B^{n-i} \\ &= 1 + \sum_{i=1}^{n} (B - 1 - y_i) \cdot B^{n-i}, \end{aligned}$$

which proves the claim. Setting $B = 2$ means we have $\bar{y} = \bar{y_1} \cdots \bar{y_n}$, where $\bar{y_i} = 1 - y_i$ (which swaps 0 and 1). Also $\bar{y}$ can be computed in constant time with n processors.

Task 3

Compute the number of paths between two nodes in a directed acyclic graph in NC.

Solution

Let $G = (V, E)$ be a directed acyclic graph (V = vertices, E = edges) and let A be its adjacency matrix, which means

$$A[i,j] = \begin{cases} 1, & (i,j) \in E, \\ 0, & \text{otherwise.} \end{cases}$$

Claim: $A^k[i,j]$ is exactly the number of paths from i to j of length k. We show this statement by induction on k. The case $k = 1$ is trivial by definition of A. Now we show the induction step. Every path of length $k+1$ in G is (obviously) a path of length k plus one additional edge. On the other hand, $A^{k+1} = A \cdot A^k$, where A^k encodes the number of paths of length k by the induction hypothesis. Therefore,

$$A^{k+1}[i,j] = \sum_{l=1}^{m} A[i,l] A^k[l,j], \quad m = |V|$$

encodes exactly the number of paths of length $k+1$ from i to j. This proves the claim.
Since G is acyclic, we know that the longest path has at most length $m - 1$. This means that the number of all paths between two given nodes i and j can be computed via the sum $\sum_{k=1}^{m-1} A^k[i,j]$. From the lecture we know that addition is in NC, hence is suffices to compute each $A^k[i,j]$ on a processor p_k. Each sum and product (see lecture) can be computed in NC with polynomially many processors. The number of substeps is $m - 1$, so if the input length is m (which means $|V|$ is encoded in unary representation), then the whole algorithm works in NC. But if $|V|$ is encoded in binary representation, we need exponantially many substeps compared to the input length. Therefore it is crucial for $m = |V|$ to be given in unary representation.