1. You are given a sorted array $A = (a_1, a_2, \ldots, a_n)$ possibly containing duplicates. Develop a $O(1)$ time algorithm to find all representatives $i$ such that $a_i$ is the start of a sequence of identical values. For example, the representatives of $A = (a_1 = 1, a_2 = 3, a_3 = 3, a_4 = 5, a_5 = 6, a_6 = 6, a_7 = 6, a_8 = 6)$ are 1, 2, 4, and 5.

2. Develop a ranking algorithm for the elements of $A$. We do this by modifying the algorithm to replace all $a_i$ with the array index of the representative of the value. For example, the algorithm should produce $A = (a_1 = 1, a_2 = 2, a_3 = 2, a_4 = 4, a_5 = 5, a_6 = 5, a_7 = 5, a_8 = 5)$ afterwards. How is this problem related to the pointer jumping algorithm? What is the time $T(n)$ and work $W(n)$ of your algorithm?

3. Consider the non-recursive prefix sum algorithm discussed in class and show on slides 20–22. Develop an optimal $T(n) = O(\log n)$, $W(n) = O(n)$ non-recursive prefix sum algorithm that does not use the auxiliary $B$ and $C$ arrays. The input array $A$ should hold the prefix sums when the algorithm terminates.
Solutions

1. EREW PRAM algorithm:

Input: Array \( A[1, \ldots, n] \) of size \( n = 2^k \)
Output: \( A[i] = i \) or \( A[i] = 0 \) for \( 2 \leq i \leq n \)
\[
\text{if } i = 1 \lor A[i-1] \neq A[i] \text{ then }
A[i] = i
\]
\[
\text{else }
A[i] = 0
\]

2. The EREW PRAM algorithm:

Input: Array \( A[1, \ldots, n] \) of size \( n = 2^k \) as defined above
Output: ranking such that \( A[i] = j \) and \( A[j] \) is the representative of the rank of \( A[i] \) for \( 1 \leq i \leq n \)
\[
\text{if } A[i] = 0 \text{ then }
A[i] = i - 1
\]
for \( 1 \leq i \leq n \)

while \( A[i] \neq A[A[i]] \) do
\[
\]

runs in \( T(n) = \mathcal{O}(\log n) \) parallel time with \( W(n) = \mathcal{O}(n) \) operations.

3. The first solution performs a bottom-up and top-down phase, where we just changed the indexing to store the results directly in \( A[] \):

Input: Array \( A[1, \ldots, n] \) of size \( n = 2^k \)
Output: Prefix sums in \( A \)
\[
\text{for } h = 1 \text{ to } \log n \text{ do }
\]
\[
\text{for } 1 \leq i \leq n/2^h \pardo
A[i \cdot 2^h] = A[i \cdot 2^h] + A[i \cdot 2^h - 2^{h-1}]
\]
\[
\text{for } h = \log n \text{ to } 1 \text{ do }
\]
\[
\text{for } 1 \leq i \leq n/2^h \pardo
A[i \cdot 2^h + 2^{h-1}] = A[i \cdot 2^h + 2^{h-1}] + A[i \cdot 2^h]
\]

An alternative algorithm is obtained by combining the computations into a single loop which yields a EREW PRAM algorithm that also runs in \( \mathcal{O}(\log n) \) parallel time, but requires more work \( \mathcal{O}(n \log n) \) (for the if-statement operations):
Input: Array $A[0, \ldots, n - 1]$ of size $n = 2^k$
Output: Prefix sums in $A$
for $h = 0$ to $\log n - 1$ do
  for $0 \leq i \leq n/2 - 1$ pardo
    if $i \mod 2^h = 0 \land 2^i + 2^h < n$ then