2.1. Build Fenwick tree from a given array in $O(n)$ time.

2.2. Build Fenwick tree from a given array without additional memory in $O(n)$ time.

2.3. Given a Fenwick tree, restore the original array without additional memory in $O(n)$ time.

2.4. How will the running time and the amount of memory used in a sparse table change if you store segments of length not 2^k, but x^k $(x > 2)$?

2.5. Add an operation to the Fenwick tree that finds the maximum prefix of an array on which the sum is at most x (all values are non-negative) in $O(\log n)$ time.

2.6. There is a chessboard $n \times n$. Process requests: 1) add / remove a rook, 2) find the number of squares in a given rectangle that are not beaten by any rook. Both requests in $O(\log n)$.

2.7. The sequence f_i is calculated according to the following rules: $f_{-1} = f_0 = 1$, $f_i = (a_i \cdot f_{i-1} + b_i \cdot f_{i-2}) \pmod M$. You need to process requests: 1) for a given i, change the numbers a_i and b_i (and recalculate the sequence), 2) find the value of f_i. Both requests in $O(\log n)$.

2.8. There are two arrays a and b. You need to process requests: 1) copy a segment array a into the array b (that is, do $b_{y+q} = a_{x+q}$ for all q from 0 to $k-1$) 2) find the value of b_i. Both requests in $O(\log n)$.

2.9. There is a city of n houses in a row, the height of the i-th house is a_i. M bombs fall on the city in succession. The bomb with the force p_j, hitting the house x_j, destroys all houses i, for which $a_i \le p_j - |x_j - i|$. Find for each house which bomb will destroy it. Time $O((n+m)\log n)$.