Binary and Informatics Olympiads

Translation Compiled by Kirito Feng

Wu Sen

CTSC 2009

Contents

§1 Binary and Data Structures

Example 1.1

Matrix. We are given a $N \times M$ binary matrix, initialized to 0. We have Q queries, where we can flip all values in a subrectangle of the grid. After each query, we query for the value of (x, y).

As we are dealing with a binary grid, each entry is either a 1 or a 0.

Firstly, let us consider an easier problem: we are given a binary string of length N, with the same updates and queries.

Thus we can add one to the left point, and add one to the point after the right point. The parity of a point (x, y) is equal to the parity of its prefix sum. This can be maintained with a Fenwick tree or similar data structure in $\mathcal{O}(\log N)$.

This can be easily extended in to two dimensions to solve the given problem, and generalizes to N-dimensional variants on this problem.

Remark. The author proceeds to spend the rest of the the analysis of this problem explaining and fanboying over Fenwick trees.

§2 Binary and Problem Solving

Example 2.1

Sudoku. Given a 3×3 Sudoku board, solve it.

Firstly, we observe that pure brute force is far too inefficient to solve the problem.

Thus we need to use very efficient pruning. Unfortunately, trying various pruning techniques while iterating the valid sequences still leads to a TLE.

We can use an optimization where each each square is associated with a binary number from 1 to 511, where the ith bit is true if i is a possible value for the square, and false otherwise. We can then try to place numbers, update the masks, and repeat, and this type of backtracking will pass.

Example 2.2

Requirements. You are given N ($1 \leq N \leq 100\,000$) 5-tuples. We define the "dissimilarity" of two 5-tuples $(a_1, a_2, a_3, a_4, a_5)$ and $(b_1, b_2, b_3, b_4, b_5)$ as $|a_1 - b_1| + |a_2 - b_2| + |a_3 - b_3| + |a_4 - b_4| + |a_5 - b_5|$. Find the maximal dissimilarity over all pairs of 5-tuples.

We observe that the $\mathcal{O}(N^2)$ brute force algorithm will obvious time out.

At the same time, we observe that the number of values per tuple is significantly smaller than the number of points. Is this information useful?

Let us consider an easier version of this problem. What if instead of 5-tuples, we were dealing with 2-tuples?

The problem then turns into one of finding the maximal Manhattan distance between points on a Cartesian grid.

We see that this is equivalent to taking the maximum over several cases, of which the coefficient of each term is either positive or negative. Thus we can iterate all of these possibilities with a bitmask and find the maximum quite easily.

Example 2.3

Cow XOR. You are given an array of integers in the range $[0, 2^{21} - 1]$. Find the subarray with the maximal XOR.

The solution is to use a prefix XOR array combined with a trie. While this might have been a novel problem back in the day, it has aged poorly and is now a template implementation question...

§3 Conclusion

In conclusion, we have discussed the Fenwick tree, and shown some new problem solving and optimization techniques based on the ideas of binary numbers.

Remark. No, I did not translate that literally, I'm tired of this paper. This paper did not age well.