"Nordic Camping" Editorial / Writeup

Hamish Starling *

July 2022

1 Introduction

This is a write-up for the problem Nordic Camping from NOI 2018. The problem is available on Kattis here. I couldn't find an existing one anywhere online so I decided that I would solve the problem myself and write a brief one. The solution I outline is without a doubt not the fastest solution possible, but it passes all test cases for 100pts. This problem does not require that many complicated techniques, just many component steps. Consider whether you want to think about it for a little longer before reading on!

2 Abridged & Abstracted Problem Description

Given an N by M binary matrix **B**, and a set S of Q points within that matrix, determine for each point P in S, the largest area for a square sub-matrix selected from **B** which includes point P and contains only 0 valued cells, no 1s. (If we consider empty campsite cells to be 0 and rocks to be 1 in the context of the problem.)

For example, as in the second sample case:

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

If $P = (3, 2)$[1] the answer is 3 since the 3x3 sub-matrix from $(3, 2)$ to $(5, 4)$ is the largest possible 3x3 matrix containing point P and no 1s. Similarly the answer for $P = (4, 3)$ is also 3, while the answer for $(2, 1)$ is 2.

3 Solution

As most Olympiad problems do, this one uses the sub-task marking scheme. The test cases are divided into a number of subsets with distinct bounds on Q,N and M. [2]. Solving all test cases within each subset scores a certain number of points, out of a total of 100 for a fully correct solution passing all test cases.

Subtask	Score	Constraints
1	20	$N \leq 50, M \leq 50, Q \leq 1000$
2	25	$N \leq 800, M \leq 800, Q \leq 100000$
3	20	$N \leq 10, M \leq 2000, Q \leq 500$
4	35	$N \leq 2000, M \leq 2000, Q \leq 100000$

[1] Using row-first coordinates, starting from (1,1) in the top left of the matrix.

[2] The sub-tasks in the Kattis version refer to K but this is not defined in the problem text so one assumes we ignore K.

The following solution will pass all 4 sub-task test sets.

This is just one solution I thought of after some time considering the problem; it's not the best.

Seeing as there are a large number of queries we do not really want to have to process the answer for every query one at a time as this is likely to be TLE [3]. Rather, we would prefer to preprocess the answer for every element in the grid (in such a way that we perceive efficiency gains from doing this, to make it fast enough), and then simply query into the 2D array **Answer** for every point in S, which could be done in constant time per point. Therefore, we use the following method:

1. Read in the campsite description (matrix **B**) and simultaneously create a partial sums array **G** where **G**[i][j] gives the sum of all values in **B** above and to the left of position [i][j] including the value at position [i][j]. This can be built in $O(1)$ per item in G using bottom up dynamic programming and thus this step has complexity $\mathbf{O(NM)}$.

2. For each element (i,j) in the matrix in row major order, consider making it the top left corner of a sub-matrix. Use Binary Search / Bisection Method to obtain the largest possible square sub-matrix that could be placed with its top left corner at (i,j) and store this in a new array **TopLeftCorner**. We find out whether a given size of sub-matrix can be placed at a given point in constant time by using the partial sums array **G** to obtain the sum of the elements in the sub-matrix. If 0, we can do it; if 1 we cannot. This step has complexity $\mathbf{O(NMlog_2M)}$ since for all NM elements in the grid we do a binary search over possible square sub-matrix sizes, which can go up to M and thus the complexity of the binary searches is each $\mathbf{O(log_2M)}$.

3. But this is not sufficient since the water source (point P) doesn't have to be the top left corner of the tent (square sub-matrix); it can be placed at any point within the tent. Therefore, we must propagate these values from the top left corner throughout the entirety of each sub-matrix to get the array **Answer**. The simplest way to do this would be to, for each point (i,j) in **TopLeftCorner**, for each point (k,l) in the sub-matrix with TLC at (i,j), set the value at (k,l) to be the value at (i,j) if and only if it is bigger than the current value at (k,l). Unfortunately this is potentially $O(n^4)$ and so will be TLE. Instead, we use a greedy/DP algorithm. First we propagate leftwards, and then downwards on the resultant array, **leftpropped**.

   (a) For each row in the matrix,

   (b) Create a priority queue which will store sizes and top left corners of the sub-matrices, ordering by size.

   (c) For each item in the row, add it to the priority queue. Then, remove from the priority queue until the current top item extends as far as the current square in the row (i.e. has not "expired").

   (d) Set the value in **leftpropped** for the current element to be the sub-matrix size value at the top of the PQ.

4. We then repeat this process on the columns to propagate the sizes down the matrix using **leftpropped** like **TopLeftCorner** and **Answer** like **leftpropped**. This step and the above are both $\mathbf{O(NMlog_2M)}$ since the priority queue can be implemented with logarithmic complexity using a heap.

5. At this point we finally read in the water sources required, and read off the answers from the **Answer** array, which is overall $\mathbf{O(Q)}$

This gives an overall complexity for the process of $\mathbf{O(NMlog_2M)}$ which is AC.

The code for this solution is available in the Kattis folder of this repo.

[3] Using the 100M operations/sec bound suggested in CP Book, we would only have 3000 operations to solve for each point P and there isn't an efficient enough algorithm to achieve this.