DT0547 - Advanced Algorithms Laboratory — Problem set 1

Lecturer: Stefano Leucci — March 3, 2020

Exercise 1.1 *Asteroid Mining.*

You are working for a company that sends robotic probes to mine precious metal from asteroids in the Kuiper Belt and you are responsible for overseeing the mining operations on a particular asteroid.

Fortunately for you, the probes are almost completely automated and they only need assistance in determining a suitable landing site. You are given a (flattened) $S \times S$ map of the asteroid, in which the surface is split into small square-ish regions indexed with two Cartesian coordinates. For each pair of coordinates $(x, y) \in \{1, \ldots, S\}^2$ you know whether the corresponding spot on the surface contains metals that can be mined by the probe.

Before landing, a probe will contact the mission control center providing a candidate mining area. As part of your job, you need to decide whether to proceed with the landing or instruct the probe to locate a different site.

Specifically, a landing request consists of 4 integers x, y, ℓ_x, ℓ_y and refers to the rectangular area $A = \{x, x+1, \ldots, x+\ell_x\} \times \{y, y+1, \ldots, y+\ell_y\}$ having (x, y) and $(x+\ell_x, y+\ell_y)$ as opposite corners. If there are at least M spots in A containing metals, then the landing should be approved, otherwise it should be denied, where M is a parameter of the problem.

In order to keep up with the increasing number of landing requests, you would like to design an algorithm that is able to preprocess the asteroid map in order to *quickly* answer landing requests.

Input. The input consists of a set of instances, or *test-cases*, of the previous problem. The first line of the input contains the number T of test-cases. The first line of each test-case contains 4 integers S, L, M, R where L is the number of locations in $\{1, \ldots, S\}^2$ that contain metals. Each of the following L lines specifies the coordinates of a location with metals as a pair of integers x and y. Finally, each of the next R lines describes a landing request with the 4 integers x, y, ℓ_x, ℓ_y.

Output. The output consists of T lines. The i-th line is the answer to the i-th test-case and contains R characters. Specifically, the j-th character of the i-th output line is `A` if the j-th request is approved and `D` if the j-th request is denied.

Assumptions. $1 \leq T \leq 10$; $1 \leq S \leq 2^{12}$; $0 \leq L \leq S$; $R \leq 2^{18}$.

The coordinates of the locations with metals are all in $\{1, \ldots, S\}^2$. Each landing request is such that all candidate mining locations are in $\{1, \ldots, L\}^2$. More precisely, if a probe sends a request for (x, y, ℓ_x, ℓ_y), then $1 \leq x \leq x + \ell_x \leq S$ and $1 \leq y \leq y + \ell_y \leq S$.

(Continues on the next page)

Example.

The following is a representation of a map with $S = 5$ and $M = 6$ possible mining sites (marked with $\star$).

5	$\star$	$\star$			
4					$\star$
3			$\star$		
2			$\star$	$\star$	
1					
y/x	1	2	3	4	5

Input (corresponding to the above map):

```
1
5 6 3 4
1 5
2 5
5 4
3 3
3 2
4 2
2 3 2 2
2 2 1 3
3 2 2 2
1 4 4 0
```

Output:

```
DAAD
```

Requirements. Your algorithm must have a preprocessing time of at most $O(S^2)$ and must be able to answer each landing request in constant time.

Notes. Some sample inputs are available at `http://people.disim.univaq.it/~stefanoleucci/AAL_1920/material/asteroid_mining.zip`. A reasonable implementation should not require more than 1 second for each input file.

Exercise 1.2 *Number station.*

A number station is a radio station that broadcasts sequences of numbers for espionages operations. You are a secret agent and you just received a transmission with the combination of a safe where vital intelligence is held. The transmission is encrypted and consists of a $n \times n$ matrix $A = (a_i, j)_{i,j}$ of non-negative integers. As per your instructions, you are to recover combination by counting the number of *non-empty contiguous submatrices* of A whose sum of elements is odd.

Time is key: design and implement an algorithm to *quickly* compute the safe combination.

Input. The input consists of a set of instances, or *test-cases*, of the previous problem. The first line of the input contains the number T of test-cases. The first line of each test-case is the integer n. The i-th of the following n lines describes the i-th row of A and contains the n integers $a_{i,1}, \ldots, a_{i,n}$ separated by a space.

Output. The output consists of T lines, each containing a single integer. The i-th line is the answer to the i-th test-case and is the number of quadruples of integers (h, k, x, y) with $1 \leq h \leq k \leq n$ and $1 \leq x \leq y \leq n$ such that $\sum_{i=h}^{k} \sum_{j=x}^{y} a_{i,j}$ is odd.

Example. The safe combination associated with the matrix

$$A = \begin{bmatrix} 1 & 6 & 5 \\ 1 & 4 & 8 \\ 0 & 2 & 4 \end{bmatrix}$$

is 16, since A has 16 non-empty contiguous submatrices whose elements sum to an odd number. Namely: $\begin{bmatrix}1\end{bmatrix}$, $\begin{bmatrix}5\end{bmatrix}$, $\begin{bmatrix}1\end{bmatrix}$, $\begin{bmatrix}1 & 6\end{bmatrix}$, $\begin{bmatrix}6 & 5\end{bmatrix}$, $\begin{bmatrix}1 & 4\end{bmatrix}$, $\begin{bmatrix}1 & 4 & 8\end{bmatrix}$, $\begin{bmatrix}5\\8\end{bmatrix}$, $\begin{bmatrix}1\\0\end{bmatrix}$, $\begin{bmatrix}5\\8\\4\end{bmatrix}$, $\begin{bmatrix}6 & 5\\4 & 8\end{bmatrix}$, $\begin{bmatrix}1 & 4\\0 & 2\end{bmatrix}$, $\begin{bmatrix}1 & 6 & 5\\1 & 4 & 8\end{bmatrix}$, $\begin{bmatrix}1 & 4 & 8\\0 & 2 & 4\end{bmatrix}$, $\begin{bmatrix}6 & 5\\4 & 8\\2 & 4\end{bmatrix}$, and A itself. Notice that the submatrix $\begin{bmatrix}1\end{bmatrix}$ is counted twice since it appears two times in A.

Input (corresponding to matrix A in the above example):

```
1
3
1 6 5
1 4 8
0 2 4
```

Output:

```
16
```

Assumptions. $1 \leq T \leq 10$; $1 \leq n \leq 512$; $\forall i, j = 1 \ldots, n,\ 0 \leq a_{i,j} \leq 1024$.

Requirements. Your algorithm must have an asymptotic time complexity of $O(n^3)$.

Notes. Some sample inputs are available at `http://people.disim.univaq.it/~stefanoleucci/AAL_1920/material/number_station.zip`. A reasonable implementation should not require more than 1 second for each input file.