IOI 2009 Day 0
Solutions

9 August 2009

1 Area

The area of a rectangle with sides A_k and B_k is $A_k \times B_k$, so we simply need to add these products up for each of the rectangles.

As noted in the problem statement, this problem was intended purely to get contestants to try out the 64-bit integer types in their chosen languages. The only thing one must keep in mind is that multiplying two 32-bit values will yield a 32-bit result, even if assigned to a 64-bit variable. The safest thing to do is to use 64-bit variables throughout.

2 Hill

First we make an observation: starting at any point in the grid, we can “walk uphill” until reaching a cell with no adjacent higher cells, which is a hill. Furthermore, if we start from a cell with height h inside a region bounded entirely by cells with height less than h (or by the boundaries of the entire grid), then this walk cannot leave this region, and hence we know that a hill must exist inside this region.

We start the algorithm knowing that somewhere in the entire grid we have a hill. The algorithm then repeatedly finds smaller and smaller rectangles known to contain hills. At each step of the algorithm, we will have a rectangle with the following properties:

1. The highest cell seen so far falls inside this rectangle[1].
2. The rectangle is bounded by cells strictly lower than this highest cell, or by the boundaries of the original grid.

Although we do not actually perform an uphill walk, the property noted above guarantees that this rectangle will contain a hill. If the rectangle is 1×1, then it consists of just a cell which is a hill, and the problem is finished. Otherwise, we can produce a smaller rectangle as follows.

First, use the laser scanner on every cell in a line that cuts the rectangle in half (either horizontally or vertically, whichever will use the fewest scans). Let H be the highest cell that has been seen so far (including the cells that have just been scanned). Now if H does not lie on the cut, then it falls into one half of the rectangle. This half then satisfies the properties above, and we have successfully reduced the size of the rectangle. If H lies on the cut, then some additional work is required. Scan the cells immediately adjacent to H that have not yet been scanned, and let H' be the new highest cell seen. If $H' = H$ then H is a hill (since we have scanned all its neighbours), and we can immediately terminate. Otherwise, H' does not lie on the cut, and we can proceed to select one half of the rectangle as before.

By repeatedly finding smaller rectangles known to contain a hill, we must eventually find a 1×1 rectangle and the problem is solved. An upper bound on the number of scans required is

$$1002 + 502 + 502 + 252 + 252 + \cdots + 5 + 5 + 3 = 3023$$

[1]At the start of the algorithm, we have not seen any cells, but this turns out not to be very important.

Slightly tighter or looser bounds can be obtained depending on exact details of the implementation, but this is not important as full points are awarded as long as the number of scans is at most 3050.

3 Museum

To solve this problem, we start by observing that if we have three vases with heights A, B and C, such that either A is odd and C even, or A even and C odd, then no matter what B is these three vases will not violate the condition of the exhibition organisers. This is because $A+C$ must therefore be odd, and so is not divisible by two, meaning that it is impossible for B to be the average of A and C.

We therefore start by arranging the vases such that we place all the even vases first, and all the odd vases second. This gives an arrangement that looks like this:

$$E_1 \quad E_2 \quad \ldots \quad E_p \quad O_1 \quad O_2 \quad \ldots \quad O_q$$

where $E_1, E_2, \ldots, E_p$ are the even heights in some order, and $O_1, O_2, \ldots, O_q$ are the odd heights in some order.

Now consider any heights A, B, C which violate the organisers' requirements. By the observations above, either A and C are both even (in which case B is even, since it appears between A and C and all the even values are grouped together), or A and C are both odd (in which case B is also odd). In other words, we can consider the problems of ordering the even numbers and the odd numbers separately.

Now suppose that $a_1, a_2, \ldots, a_p$ is a permutation of the heights $1, 2, \ldots, p$ which satisfies the organisers' requirements (this is a smaller instance of the problem, so it can be solved by divide-and-conquer). Then simply assigning $E_i = 2a_i$ will satisfy the requirements on the even values. Similarly, given a permutation $b_1, b_2, \ldots, b_q$ of the heights $1, 2, \ldots, q$ which satisfies the requirements, we can assign $O_i = 2b_i - 1$.

Examining the properties of the resulting sequence gives another approach to generating the same solution. We can write all the heights in binary form, and then sort them according to the *reverse* of their binary form. This sorts first on the least significant bit (i.e., on whether they are odd or even), then the next least significant bit and so on. To prove that this solution is valid, note that if B is the average of A and C, then at the least significant bit that A and B first differ, A and C must have the same value for that bit, placing A and C in a separate group from B when sorting on that bit.