PYRAMID (Expected solution)

Author: Hugo Ryckeboer (Argentina)

SOLUTION 1: For every possible position of the pyramid, add up the heights of all the squares in the pyramid base; after that for every possible position of the chamber within that pyramid, substract the heights of the squares in the chamber to the number obtained for the pyramid base. Keep in memory the maximum height for that position of the pyramid and proceed to the next possible position. This algorithm has a running time of $O(n^3\ m^3)$ and should score at most 30 points.

SOLUTION 2: Instead of recalculating the sum of the heights for each possible position of the pyramid within the field, or the chamber within the pyramid, use the previous result. That is, calculate the first position, after that in order to move, for example, to the left, just add the new column of squares and substract the rightmost one. This can be optimized by pre computing the sum of all the rectangles of size *a* x *b* and of size *c* x *d* in time *O(n m)* and storing them into an array. This algorithm has a running time of $O(n^2\ m^2)$ and should score between 50 and 59 points depending on how good is the implementation.

SOLUTION 3: Pre compute the sum of all rectangles of size *a* x *b* and of size *c* x *d* as described in solution 2. For each row and for each column in the grid create a binary tree that allows you to search for the smallest number within any interval in time log time, populate these trees with the values of the rectangles of size *c* x *d*. Instead of checking every possible position of the chamber within the pyramid, use the binary trees to search for the minimum value of the chambers within the desired range. This algorithm has a running time of *O(n m (logm + logn))*. If efficiently coded this algorithm scored 100 points.

OPTIMAL SOLUTION: Pre compute the sum of all rectangles of size *a* x *b* and of size *c* x *d*. Now we need a way to find the minimums for each row and column in linear time. Suppose that *a-c-1 = 8*, that is, the number of different columns where the chamber can start within a given pyramid is 8. That means that for every group of 8 consecutive rectangles of size *c* x *d* in a row, we have to find the minimum among them to know which is the best position to locate a chamber within that row.
Look at the following figure

...	23	45	56	32	34	27	32	43	56	45	67	23	12	11	34	43	25	14	...
	23	27	27	27	27	27	32	43											
								43	43	43	43	23	12	11	11				
	23	27	27	27	23	12	11	11											

The first line contains the sums of the rectangles of size *c* x *d* for some row.
The second line contains a vector of length 8 that was constructed growing from right to left taking the minimum value between the corresponding position in the first line and the previous value in the vector.
The third line is a vector constructed as that of the second line but growing from left to right.
Finally the *i-th* position of the fourth line contains the minimum value of every group of eight consecutive squares starting at position *i*. To compute each minimum take the minimum value of the *i-th* position of the vector in the second line and the *i-th* position of the vector in the third line.

Constructing these vectors for each row and column will give you the minimum value of every possible chamber in time *O(n m)* thus resulting in an algorithm with running time *O(n m)*.

Test run #	Group/Value	*m*	*n*	*a*	*b*	*c*	*d*
1	1/0	8	5	5	3	2	1
2	2/10	8	8	5	5	2	2
3	2/10	8	8	5	5	2	2
4	2/10	8	8	5	5	2	2
5	2/10	8	8	5	5	2	2
6	3/10	8	8	5	5	1	2
7	3/10	8	8	5	5	2	2
8	3/10	8	8	5	5	2	2
9	3/10	8	8	5	6	2	3
10	4/10	5	5	3	4	1	2
11	5/20	10	10	5	4	2	1
12	5/20	100	100	50	50	25	25
13	5/20	115	115	50	50	25	25
14	6/3	130	130	50	50	25	25
15	7/3	150	145	75	50	40	30
16	8/3	250	250	75	125	40	80
17	9/3	400	400	300	200	100	100
18	10/3	750	745	375	350	140	230
19	11/3	850	850	467	345	245	198
20	12/3	950	950	426	550	200	300
21	13/29	1000	1000	500	500	250	250
22	13/29	1000	1000	500	500	250	250
23	13/29	1000	1000	500	500	250	250
24	13/29	1000	1000	500	500	250	250