Lefkaritika: solution

This problem can be approached in various ways. For each (i, j) we need to find the largest square with top left corner at (i, j) that fits without containing a lamp. Suppose $W = L = N$ for clarity.

1 $O(N^2 \log N)$ time, $O(N^2)$ space (53 points)

We can binary search for the maximum side of the square, checking if it is valid by counting the number of lamps that lie inside it with 2D prefix sums. To do this, we compute $S(i, j)$ which holds the number of lamps in the rectangle $[1, i] \times [1, j]$. We can compute this using the relationship $S(i, j) = A(i, j) + S(i-1, j) + S(i, j-1) - S(i-1, j-1)$, where $A(i, j) = 1$ if there is a lamp in position (i, j) and 0 otherwise. Now, the number of lamps in the rectangle $[x_1, x_2] \times [y_1, y_2]$ is $S(x_2, y_2) - S(x_1 - 1, y_2) - S(x_2, y_1 - 1) + S(x_1 - 1, y_1 - 1)$.

2 $O(N^2 \log B)$ time, $O(N + B)$ space (66 points)

To get the next subtask we should use much less space. Note that the (maximum) side of the square with top left corner at (i, j) (provided that $A(i, j) = 0$) can be at most one larger than the sides of the squares with top left corners at $(i+1, j), (i, j+1), (i+1, j+1)$. In fact, it can always be the minimum of these 3 sides plus 1 (Try to prove it formally). Thus we have that $f(i, j) = 1 + \min\{f(i+1, j), f(i, j+1), f(i+1, j+1)\}$. We can compute these values in time $O(N^2 \log B)$ (keeping the lamp positions in a binary search tree) and space $O(N + B)$, by keeping just the last 2 rows of the DP array.

3 $O(NB + B^2 \log B)$ time, $O(B^2)$ space (100 points)

Now, if N is large and B not so large, it is better to solve this problem without traversing the whole grid. Fix a diagonal. Note that there are only $O(B)$ points on this diagonal that need to be checked, as for the intermediate points there can be no extra lamp. So for each diagonal, we can find these $O(B)$ points (they are the intersections with rows and columns that contain lamps) and devise a recurrence relation that computes the answer for all of them. Specifically, suppose that we want to compute $f(i, j)$ and we already know $f(i+c, j+c)$. If there is no lamp in rows $i \ldots i+c-1$ or columns $j \ldots j+c-1$, then $f(i, j) = c + f(i+c, j+c)$. On the other hand, if there is a lamp on row i or column j, then $f(i, j)$ could be less than that. In fact, if the first lamp on row i to the right of (i, j) is in column j' and the first lamp on column j below (i, j) is on row i', then $f(i, j) = \min\{c + f(i+c, j+c), j' - j, i' - i\}$. To compute i'

and j' efficiently, we introduce $right(i, j)$ and $down(i, j)$, where i is a row that contains a lamp and j a column that contains a lamp. Note that the number of entries is $O(B^2)$. $right(i, j)$ is the index of the first column to the right of j, such that $A(i, right(i, j)) = 1$, and respectively $down(i, j)$ for a row below i. So $i' = down(i, j)$ and $j' = right(i, j)$. If we precompute $right$ and $down$, we can compute $f(i, j)$ in $O(1)$. It is not hard to compute them, since $right(i, j) = j$ if $A(i, j) = 1$ and $right(i, j) = right(i, nxt_j)$ otherwise. Here nxt_j is just the next column after j that contains a lamp.

Now, how to also compute $\sum_{k=1}^{c-1} f(i + k, j + k)$ in $O(1)$? It is simple, since it is equal to $\sum_{k=1}^{c-1}(c - k + f(i + c, j + c)) = \frac{c(c-1)}{2} + (c - 1) \cdot f(i + c, j + c)$.

4 $O(B^2 \log B)$ time, $O(B)$ space (100 points)

It is interesting that we can make our solution even faster for large N. We will outline the general idea, so there are more details not mentionted here. Note that for every cell, we can assign exactly one lamp, that blocks it from having an even larger square. Think in reverse: For each lamp, find the sum of answers for the cells that are blocked *by it*. In fact, the pattern of these cells for a specific lamp is very well defined. It can be found by iterating over all other lamps, and then the sum can be obtained by standard algebraic formulas. The runtime is $O(B^2 \log B)$ and the space is $O(B)$ (independent of N).

In the figure below, we have three lamps, A, B, and C. The red filled area contains exactly the cells that are blocked by A. All lines are 0, 45, or 90 degrees, so it is not hard to break the red region into triangles and rectangles, compute the sum of answers for each of them, and add them up together. Note that there will also be one red solid line "above" A, corresponding to lamps on the upper right of A, as well as a third solid red line joining these two ones, corresponding to lamps on the upper left of A.