1.

Given that the crane does not intersect with obstacles, the shapes of C-obstacles can traced in following cases:

* When p3 lies on the top edge of the obstacle; p2 lies above the obstacle; p1 lies to the left of the obstacle, a part of C-obstacle is defined by:
  + r-(x-, y-) that 0 <= x- <= w
  + r+(x+, y+) that h – 1 <= y+ <= w
  + This C-obstacle part surrounds the upper-left of the obstacle
* When p2 lies on the left edge of the obstacle and p1 lies to the left of the obstacle, a part of C-obstacle is defined by:
  + r-(x-, y-) that 0 <= x- <= w
  + r+(x+, y+) that y+ >= h
* When p1 lies on the bottom edge of the obstacle, a part of C-obstacle is defined by:
  + r-(x-, y-) and r+(x+, y+) that 0 <= y- <= h
* When p3 lies on the top edge of the obstacle; p2 lies above the obstacle; p1 lies to the right of the obstacle, a part of C-obstacle is defined by:
  + r-(x-, y-) and r+(x+, y+) that 0 <= x+ <= w and h-1 <= y+ <= h
* When p2 lies on the right edge of the obstacle and p1 lies to the right of the obstacle, a part of C-obstacle is defined by:
  + r-(x-, y-) and r+(x+, y+) that 0 <= x+ <= w and y- <= h <= h+

2a.

Given that every rectangle is defined by 3 points A, B, and C that: A and B lie on the bottom horizontal strip; B and C lie on the right vertical strip.

This problemed can be transformed into the orthogonal range searching problem in a 3-dimenstional space because:

* The query is a rectangle.
* All rectangle S completed in a query rectangle Q can be defined by 3 dimensions:
  + X-coordinates of A, B, and C
  + Y-coordinates of A, B, and C
  + The angle between two segments AB and BC such that A, B and C form a rectangle if and only if the angle equals to 90 degrees.

Similar to the ordinary orthogonal range tree, our proposed data structure is a balanced binary tree T storing sets of 3 points defining a rectangle such that:

* In the 1st or x-coordinate dimensional range tree, all points A and B (C shares the y-coordinate with B) are sorted by x-coordinate. The query interval is Q = [x-lo, x-hi]. This dimension takes O(logn) space.
* In the 2nd or y-coordinate dimension, all points B and C (A shares the x-coordinate with B) in the canonical set corresponding to an internal node u in the 1st dimension are sorted by y-coordinates. The query interval is Q = [y-lo, y-hi]. This dimension takes O(logn) space.
* In the 3rd or angle dimension, for every combination of 3 points A, B, and C of a canonical set of points selected from the 2nd dimension, their corresponding angles formed by AB and BC are sorted by degree from 0 to 180. We assume C lies above segment AB. The query interval is Q = [0, 180]. This dimension takes O(logn) space. The final combination must form a 90 angle.

Since each dimension is a binary tree, it takes O(logn + k) to report k rectangles. Totally, the query time complexity is O(log^3n + k) that k is the reported rectangles. The space complexity of each binary tree is O(nlogn). Hence, the total space complexity of the 3D orthogonal range tree is O(nlog^3n).

2b.

The triangle is guaranteed to have one horizontal edge, vertical edge, and one edge of slope -1 or 1. This information refers to a right and equal triangle.

Every right and equal triangle is defined by:

* 3 points A, B and C such that AB = AC
* The angle formed by AB and AC equals 90 degrees.
* The two angles formed by AB-BC and AC-BC equals 45 degrees.

Similar to 2a, we can transform this problem into a 3D rectangle range searching problem that points are queried by right and equal triangles.

Our proposed data structure is a balanced binary tree T:

* In the 1st dimension, all points are sorted by x-coordinate. The query interval is Q = [x-lo, x-hi]
* In the 2nd dimension, all points (belongs to a canonical set in the 1st dimension) are sorted by y-coordinate. The query interval is Q = [y-lo, y-hi] that |y-hi – y-lo| = |x-hi – x-lo|
* In the 3rd dimension, all points P (belong to a canonical set in the 2nd dimension) are sorted by angles formed by AB and a segment connecting B with each point in P. The final point set forms angles with B <= 45. In other words, the corresponding slopes are either -1 or 1.

Since each dimension is a binary tree, it takes O(logn + k) to report k points. Totally, the query time complexity is O(log^3n + k) that k is the reported points. The space complexity of each binary tree is O(nlogn). Hence, the total space complexity of the 3D orthogonal range tree is O(nlog^3n)

Extra credit:

2 right and equal triangles sharing the same edge of slope -1 or 1 will form a square. This common edge is defined by two points A and C which are lower-left and upper-right points of the square respectively.

Similar to 2b, we can transform this problem into a 3D orthogonal range searching problem such that all reported points belong to a query square. Then, we can split the square into 2 right and equal triangles by the common edge that contains the final points.

We can achieve the O(nlog^2n) space and the O(log^2n +k) query time by removing the 3rd dimension. The proposed data structure is similar to 2b such that:

* In the 1st dimension, all points are sorted by x-coordinate. The query interval is Q = [x-lo, x-hi]
* In the 2nd dimension, all points (belongs to a canonical set in the 1st dimension) are sorted by y-coordinate. The query interval is Q = [y-lo, y-hi] that |y-hi – y-lo| = |x-hi – x-lo|
* Final points can be found by linearly determining the xy-coordinate of each point belongs to either right and equal triangle halves of the query square (explained above). The time complexity to trace points in each triangle half is k/2.

By removing the 3rd dimension, the total space complexity is O(nlog^2n). The total time complexity is O(log^2n + k/2) = O(log^2n + k)