Code Jam to I/O for Women 2020 - Code Jam to I/O for Women

Analysis: Impromptu Outdoor Gallery

We can start by making the observation that for every valid quadrilateral, $p_1p_2p_3p_4$, at least one of the two diagonals divides the quadrilateral into two triangles. We can order the points in the quadrilateral such that the diagonal formed by points p_1 and p_3 splits the quadrilateral into two triangles. This means that points p_2 and p_4 are on opposite sides of the infinite line through p_1 and p_3 or vice versa. Without loss of generality, we can assume the former, since it is just a renaming of the same set of points in the same order. The area of $p_1p_2p_3p_4$ is equal to the sum of the areas of the two triangles $p_1p_3p_2$ and $p_1p_3p_4$. These areas are equal to half of the product of the distance between p_1 and p_3 and the distance between p_2 (or p_4) and the line through p_1 and p_3. Therefore, if we choose two points to be p_1 and p_3 in our quadrilateral, we can find the minimum area using those two points by getting the closest point to the line through p_1 and p_3 on each side of the line.

Test set 1

With only 25 points, we can test every set of 4 points in every order. To make sure a specific order defines a simple polygon, it suffices to check the sign of the signed area of both triangles is positive. This can be done by checking the sign of the cross product when computing the area of the two triangles that make up the quadrilateral.

We could also can loop over all possible pairs of points to use as p_1 and p_3, and then, for each pair, loop through the remaining points to find the closest on each side. There are $O(\mathbf{N}^2)$ unique pairs of points and for each pair, we check $O(\mathbf{N})$ other points to find the closest on each side. This gives an overall time complexity of $O(\mathbf{N}^3)$.

Test set 2

In test set 2 we have significantly more points to choose from that a time complexity of $O(\mathbf{N}^3)$ will not work for the given time limit. We can, however, precompute all lines formed by unique pairs of points and do a circular sweep line, processing those lines in order of their slopes; this will make it much faster to find the closest points to the lines. As we rotate, we keep the points in a list sorted by their distance to the line.

We can add the horizontal line to the set of stop points, and use it as the starting angle for the sweep line. This simplifies initialization because sorting the points by distance to this line is simply sorting by y-coordinate. Then, for any p_1 and p_3, the closest points to the line p_1p_3 must be ones which are adjacent to either p_1 or p_3 in the sorted list (because p_1 and p_3 have a distance of 0). There is a constant number of candidates, so we can check them all in constant time to find the best p_2 and p_4.

As we rotate our orientation to line up with the next smallest sloped line, we can see that the only points that change their ordering in the sorted list are the two points of the line segment being looked at. Taking advantage of this, we can maintain our sorted list of points by just swapping the adjacent pairs of each line after we consider them in order of slope.

Because we still have $O(\mathbf{N}^2)$ unique lines and we need to sort them by their slopes, the initial time complexity to generate the sorted list of lines and the initial ordering of the points is $O(\mathbf{N}^2 \log \mathbf{N})$. After we have all the line segments sorted, we can maintain the sorted list and go through each line in order and find the closest points in O(1) per line. This results in an overall time complexity of O($\mathbf{N}^2$ log **N**).