Approximation

Group 1 $x_i = 0$ or $x_i = 1$ — 11 points

To solve this subtask we can directly find the actual area of the convex hull, since we only have to consider at most 4 points: points with minimum y and maximum for $x = 0$ and $x = 1$. However, there is an easier way: simply find the maximum value of y coordinate and minimum value of y coordinate on range, their difference is valid answer (don't forget about the case when all points have same x coordinate). Using segment tree or sparse table problem can be solved in $O(n + q \log(n))$.

Group 2 $n, q \leq 1000$ — 9 points

This subtask can be solved directly, simply find the convex hull on the range using algorithm you like. Depending on your solution it can be done in $O(qn \log(n))$ or $O(qn)$.

Group 3 Rubber Band touches all of the nails — 13 points

In this subtask we don't have to build convex hull (since all of the points already on the convex hull). We can use the almost any formula (I will build trapezoids under the sides of the initial hull) to calculate the area of the convex hull. According to the formula area of the convex hull is equal to $\frac{1}{2}\sum_{i=1}^{i=n}(y_i + y_{i+1})(x_i - x_{i+1})$ (assume that $(x_{n+1}, y_{n+1}) = (x_n, y_n)$). We can see the this sum can be easily calculated for some range $[l, r]$ using prefix sums, but we have to consider the edge $(x_r, y_r) - (x_l, y_l)$ separately.

General Idea

Problem asks us to find the approximate area that is larger than the actual area. However, if we can find approximate area in the range $[S/2, S]$ we can multiply it by 2 and get the valid answer.

Group 4 x_i, y_i are generated randomly — 15 points

This subtask can be solved in many ways. One of them is to use the fact that randomly generated set of points has a very small number of points on the convex hull. So, we can use Divide and Conquer to directly calculate the convex hull (store convex hulls for ranges $[l, m]$ and $[m + 1, r]$ and merge them naively).

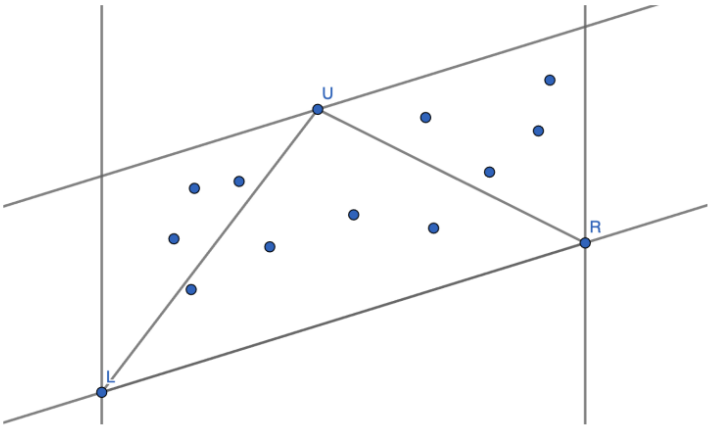
Another idea is to use first 100 points of the query, and use their convex hull. This value can be smaller than the actual area, but multiplying it by 2 gives a valid answer, since first 100 point are enough to get at least half of the required area.

Group 5 $n, m \leq 10^5$ — 34 points

Let's find the point with minimum x and call it L. We can do the same and find the point with maximum x and call it R. Build segment LR and find the point above the segment with maximum distance from it and call it U. Notice that triangle LRU has the area that is at least half of the area of the convex hull of the points above the segment LR.

Proof: build a line parallel to LR though point U, and two vertical lines through points L and R, region between this lines and line LR is parallelogram, notice that all of the points that above LR must lie inside of this parallelogram (all of the points obviously between the vertical lines and they must be under the line that goes though point U, since U is the farthest point from LR). Therefore, area of the convex hull of all points above LR is at most the area of the parallelogram, at the same time the area of the triangle

LRU is half of the parallelogram. So, its area is at least half of the area of convex hull of points above LR.



We can do the same by considering points below LR, and finding farthest point D from LR. Area of the quadrilateral $LDRU$ is at least half of the actual area.

Now to the algorithmic part of the solution. Finding points L, R is easy if you use segment tree. What we want to do is to find farthest point is some direction for some set of points (in our case direction normal to LR). Notice that for any set of points farthest one in any direction is always a point from the convex hull, and we can use the binary search on the convex hull to find this farthest point! To find point U we can do the following: let's build segment tree that would contain convex hull of the points in the subtree of the vertex, this can be done in $O(n \log(n))$ though MergeSort and Andrew's monotone chain algorithm. So, we can use this idea and do binary search in $\log(n)$ vertices of the segment tree.

Overall time complexity: $O(n \log(n) + q \log(n)^2)$

Group 6 $n, m \leq 2 \cdot 10^5$ — 20 points

This subtask is almost same compared to previous one, but we can reduce one log in the queries. Notice, that we can solve problem offline. First of all sort the queries to the segment tree in counter-clockwise order. Now, instead of binary search in vertex we can store pointer to the last answer, and use the fact that sorted queries mean that this pointer moves in one direction.

Overall time complexity: $O(n \log(n) + q \log(n))$