Divide and conquer

(in main)

sort (points based on x coordinate) → n log n

divide and conquer (Points)
{
  if no of points <= 3
    sort and return as (base case)

  int mid = n/2

splits it into two halves of size n/2 recursively

  (divide) left points (points (begin) → points [mid])
  (divide) right points (points [begin + mid] → points [end])

  return merge hull (left, right)
}

merge hulls ( )
{
  points ans = left
  ans. insert (ans. begin) right. begin, right. end)

  points hull
  hull (all points in ans)

lower hull
O(n)

  while (hullsize() >= 2 and cross product (hull (hullsize() - 2), hull (hullsize - 1), p) <= 0)
    hull. pop ()
  hull. push

int lower = hull.size
for (int i = arr.size - 2 to 0)
{
while hull.size() > lower size and cross product
(hull[hullsize - 2], hull[hullsize - 1], merged) ≤ 0
hull.pop()
hull.push(arr[i])
}

upper hull
$O(n)$

hull.pop back() → remove duplicate

Thus we can say that every recurrence is

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

left + right — mergehull

$$T(n) = 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n = 4T\left(\frac{n}{4}\right) + n + n$$

$$= 8T\left(\frac{n}{8}\right) + 3n$$

$$= 2^k T\left(\frac{n}{2^k}\right) + kn$$

at $2^k = n$

$$= n \times T(1) + \log_2 n \cdot n = O(n \log_2 n)$$