Implementation documentation

Tracing algorithm was largely underestimated in the requirements documentation. Algorithm's rough progression:
1. Choose a vertex v from which point of view other vertices are traced.
2. Choose another point p. Store the direction and distance from v to p. Also store direction and distance from v to p.right, assuming that p is a part of a polygon and thus has a connection with a left and right neighbour which are a part of the same polygon. This data can be represented as a sector of two directions.
3. Repeat 2 for every point in the field. Sectors are stored in a heap, where a sector with the smallest left direction value is placed on top of the heap. Time requirement: O(p*log(p)) Memory requirement: O(p), where p is amount of geometry points.
4. Go through the sectors and remove redundant and overlapping sectors. Each stored sector is pulled from the heap (Time: O(p*log(p))) and stored in a tree (Time: O(p*log(p))), where sectors with smaller destination value are stored as a left child. Using this technique, amount of stored sectors is reduced significantly. However the factor of sector reduction is hard to estimate. Memory requirement: O(p).
5. All vertices are checked whether they are obstructed by any sector. Time requirement: O(w*s), where w is amount of vertices in a field (which is less than p) and s is amount of usable sectors (also less than p). Memory requirement: O(q), where q is amount of points that are unobstructed to the v. That way q <= p.
6. Repeat this for every v in the field.
Overall time requirement: O(w * ((p*log(p)) + (w*s))) $< O(p^3)$. Memory requirement: O(p).