Merge Convex Hulls Algorithm

The main idea of this algorithm is that vertices that are almost collinear should end up a connected polygonal chain. To achieve this the algorithm recursively builds up convex hulls of the points. The result is a set of layers of convex hulls, similar to an onion. Then the convex hulls are merged to build a polygon that consists of all the given points.

Pseudocode

```
function merge_convex_hulls(PointSet point_set) -> Polygon {
    List convex_hulls := create_convex_hulls(point_set);
    Polygon polygon = convex_hulls.pop_front();
    while (convex_hulls is not empty) {
        Polygon new_hull = convex_hulls.pop_front();
        polygon = merge(polygon, new_hull);
    }

}

/**
 * Iteratively creates a convex hull for the given point set and
 * removes the points from the set.
 * Repeats this process until the set is empty.
 * Returns a list of the created convex hulls
 * The hulls in the list are sorted from most outside to most inside.
 */
function create_convex_hulls(PointSet point_set) -> ListOfConvexHulls {
    List convex_hulls := {};
    while (point_set is not empty) {
        Polygon new_hull = create_convex_hull(point_set);
        convex_hulls.add(new_hull);
        point_set.remove_all(new_hull.get_vertices());
    }
    return convex_hulls;
}

function merge(Polygon polygon, ConvexPolygon new_hull) -> Polygon {
    // TODO add implementation
    for (Point vertex : new_hull.vertices()) {
        Segment nearest_segment = find_nearest_segment(polygon.edges(), point);
        polygon.insert_point_at_segment(nearest_segment, point);
    }
    return polygon;
}
```

bla