Описание алгоритма за $O(n^4)$

Алгоритм принимает множество S из n непересекающихся отрезков и состоит из следующих шагов:

1. P - множесто концов отрезков из S. Строится множество L, прямых, определяющихся любой парой $p_i, p_j \in P$.

   Данная операция требует $O(n^2)$ времени, $|L| = 2n^2 - n$.

   Для дальнейшей обработки множество точек P сохраняется. Для каждого отрезка запоминаются индексы его концов. Контейнер занимает $2n$ памяти.

2. Строится упорядочение прямых $A(L)$ с помощью инкрементального алгоритма. Его сложность для m прямых есть $O(m^2)$. Соответственно в данном случае потребуется $O(n^4)$.

   Результатом работы данного алгоритма является РСДС, занимающий линейную память от числа ребер, вершин и граней. Так как это число для $A(L)$ квадратично зависит от количества прямых в L, то расход по памяти на данном этапе алгоритма достигает $O(n^4)$.

   Для каждого ребра сохраняется информация об отрезках из S, на точках которых была построена прямая, частью которой является данное ребро.

3. Находится любая непустая грань f РСДС, производится упорядочивание P по возрастанию полярного угла относительно любой вершины q данной грани.

   В случае равенства дополнительно производится следующая операция:

   (a) Из двух сходящихся в q ребер f находится то, которое является частью прямой, проходящей через равные точки.

   (b) Производится сдвиг от него на ε по нормали в сторону f.

   (c) Значения углов пересчитываются, точки заносятся в получившемся порядке.

   После сортировки все отрезки проверяются на «правильность» (индексы i и j точек отрезка в контейнере должны быть соседними, p_iqp_j должны образовывать левый поворот). Для каждого отрезка запоминается «правильный» ли он. Запоминается количество правильных отрезков. Сортировка занимает $O(nlog(n))$, проверка на правильность - $O(n)$.

4. Производится обход РСДС (например в ширину), начиная с грани f. На каждом шаге, зная «перешагиваемое» ребро, через обращение к отрезкам находятся индексы, порождающих его точек.

   Если точки относятся к одному отрезку, они меняются местами в контейнере. В противном случае для всех точек, задающих отрезки (их 4), производится упорядочивание из прошлого пункта для граней до и после «перешагивания». Если сортировки порождают разные перестановки данных точек, то точки отрезков, которые задают ребро, меняются местами.

   Все отрезки, которые затронула прошлая операция, проверяются на «правильность». Запоминаются текущие состояния отрезков. Вносится изменение в число «правильных»

отрезков (если надо). Если при проверке угла оказалось, что выбранная вершина q совпадает с одной из точек p_i, p_j, на время проверки за q принимается другая вершина грани, несовпадающая с p_i, p_j. В случае пустых граней такой вершины может не быть, тогда без проверки считается, что «правильность» отрезка изменилась.

Обработка каждой грани требует $O(1)$ времени и памяти.

5. Обход продолжается до тех пор пока не найдется *непустая* грань, в которой все отрезки окажутся «правильными», или пока не останется непосещенных граней. В первом случае ответом алгоритма - «да» с предоставлением любой точки внутри найденной грани, во втором случае ответ - «нет».

   Обход требует $O(N^4)$ времени.