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 точки q из внутренности данной грани.

   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, кроме индексов его концов, запоминается поворот. 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 i и j, порождающих его точек.

   Точки меняются местами в контейнере, если:

   (a) Они относятся к одному отрезку

   (b) Они относятся к разным отрезкам и «перешагиваемое» ребро не является подмножеством отрезка p_ip_j

   Все отрезки, точки которых изменили свое положение в контейнере, проверяются на «правильность». Для проверки поворота требуется выбрать точку на грани:

   (a) Если грань непустая выбирается любая внутренняя точка.

   (b) Если грань пустая, но ее ребра не лежат на прямой порождаемой точками отрезка, выбирается внутренняя точка любого ее ребра (например середина)

   (c) Если грань пустая и ее ребра лежат на прямой порождаемой точками отрезка, без проверки считаем, что поворот отрезка сменился.

Запоминаются текущие состояния отрезков (положения точек в контейнере, поворот) Вносится изменение в число «правильных» отрезков (если надо).

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.