Описание алгоритма за $O(n^4)$

Алгоритм принимает множество S из n непересекающихся отрезков. P - множесто концов отрезков S, является МТОП-1.

1. 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. Произвольно выбирается грань f РСДС. Производится упорядочивание P по возрастанию полярного угла относительно любой внутренней точки f.

   Производится заметание по углу, в процессе которого получаем кольцевой список, хранящий следующие данные:

   (a) Индексы точек, задающих сектор.

   (b) Статус, содержащий информацию о нахождении в секторе для каждого отрезка (bool контейнер размера n)

   Статус для нулевого сектора находится честной проверкой на пересечение с помощью построения бесконечной прямой по нулевому углу. Статусы остальных секторов определяются за константу, исходя из события: меняем знак в статусе того отрезка, чья точка - новое событие. Также в процессе заметания подсчитывается и запоминается число секторов (k), у которых в статусе > 1 отрезка (со значенем $True$).

   Сортировка занимает $O(nlog(n))$, заметание - $O(n)$.

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. Зная точки, можем найти секторы, которые оказываются затронутыми на этом шаге. Изменения, которые претерпевает список определяется конечным набором возможных случаев. Все случаи отдельно рассмотрены в приложении.

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

5. Обход продолжается до тех пор пока не найдется грань, для которой после пересчета статусов число k примет значение 0, или пока не останется непосещенных граней. В первом случае ответом алгоритма - «да» с предоставлением любой точки внутри f, во втором случае ответ - «нет».

   Обход требует $O(N^4)$ времени.