Описание алгоритма за $O(n^4)$

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

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 грани, обозначим точку q.

   После сортировки все отрезки проверяются на «правильность». (индексы i и j точек отрезка в контейнере должны быть соседними) 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.

   Если на следующем шаге оказывается необходимым перешагнуть через ребро, которое является частью одного из исходных отрезков, то данный шаг пропускается. Определение этого факта занимает константное время.

   Существует конечное количество случаев того, как может изменится сектор заметания, заданный точками p_i, p_j после «перешагивания». Все случаи подробно рассмотрены в приложении.

   С учетом имеющихся данных, случай определяется за константное время. Столько же требуется на внесение изменений в порядок точек в контейнере, в количество «правильных» отрезков. Таким образом, обработка каждой грани требует $O(1)$ времени и памяти.

5. Обход продолжается до тех пор пока не найдется 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.

Источники:

1. de Berg, Mark; Cheong, Otfried; van Kreveld, Marc; Overmars, Mark (2008). Computational Geometry, Algorithms and Applications (3rd ed.). Springer. pp. 172–177. ISBN 978-3-540-77973-5.