

## Лекция 9. Алгоритмы пересечения геометрических объектов

### 1. Задача о пересечениях геометрических объектов

Задача о пересечениях состоит в установлении факта расположения нескольких объектов в одном и том же месте. Проблемы с ее решением связаны обычно с большой размерностью задачи. Примеры областей, где возникает эта задача:

- архитектурное проектирование (объем базы более  $10^6$  элементов);
- машинная графика ( $\geq 10^5$  векторов);
- проектирование интегральных схем ( $\geq 10^6$  компонентов).

При такой размерности даже алгоритмы сложности  $O(N^2)$  являются неприемлемыми.

Как правило, задача пересечения объектов сводится к проверке пересечения многоугольников или прямолинейных отрезков.

**Задача 1** (пересечение отрезков). Даны  $N$  прямолинейных отрезков на плоскости. Надо определить факт пересечения хотя бы двух из них.

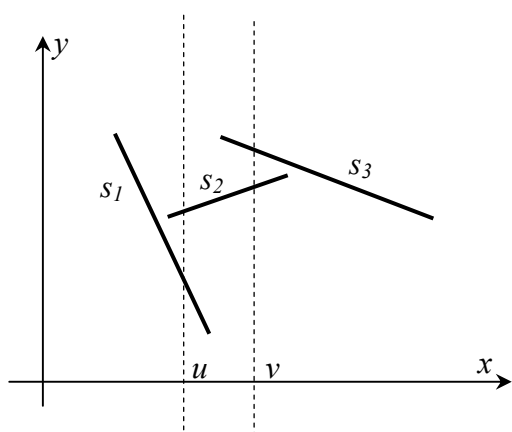
**Задача 2** (пересечение простых многоугольников). Даны два простых многоугольника. Требуется определить, пересекаются ли они.

**Задача 3** (проверка простоты многоугольника). Дан многоугольник на плоскости. Проверить, прост ли он.

Очевидно, что задачи 2 и 3 сводимы за линейное время к задаче 1.

### 2. Упорядочение отрезков на плоскости

Рассмотрим пару непересекающихся отрезков  $s_1$  и  $s_2$  на плоскости. Будем говорить, что эти отрезки сравнимы в абсциссе  $x$ , если существует вертикальная прямая, пересекающая их обоих. При этом введем отношение порядка для сравнимых в  $x$  отрезков (обозначаемое  $>_x$  и называемое «выше»). Отрезок  $s_1$  лежит выше отрезка  $s_2$  в абсциссе  $x$  (т.е.  $s_1 >_x s_2$ ), если ордината точки пересечения  $s_1$  с вертикальной прямой больше, чем ордината точки пересечения  $s_2$  с этой прямой. В примере на рисунке имеет место  $s_2 >_u s_1$  и  $s_3 >_v s_2$ .



Если рассмотреть движущуюся слева направо вертикальную прямую, то каждый отрезок из заданного множества отрезков в течение какого-то времени пересекается этой прямой. В каждый момент времени отрезки, пересекаемые прямой, полностью упорядочены в смысле введенного отношения порядка. По мере перемещения прямой упорядочение может меняться в одном из трех

случаев:

1. Встретился левый конец отрезка  $s$  (этот отрезок добавляется к множеству пересекаемых в данный момент отрезков);
2. Встретился правый конец отрезка  $s$  (этот отрезок удаляется из этого множества);
3. Обнаружена точка пересечения отрезков  $s_1$  и  $s_2$  (тогда эти отрезки меняются местами в смысле их упорядоченности).

Предположим, что множество отрезков устроено таким образом, что одновременное пересечение трех или более отрезков в одной точке невозможно. (Это предположение не очень сильно нарушает общность рассматриваемых алгоритмов, но упрощает их описание.) Тогда легко видеть, что справедливо следующее

**Необходимое условие пересечения пары отрезков:** Если отрезки  $s_1$  и  $s_2$  пересекаются, то существует абсцисса  $x$  такая, что  $s_1$  и  $s_2$  являются смежными в упорядочении  $>_x$ .

На этой идее строится эффективный алгоритм решения задачи пересечения множества отрезков.

### 3. Алгоритмическая парадигма плоского заметания

Будем рассматривать динамический процесс, состоящий в перемещении вертикальной прямой слева направо (из  $-\infty$  в  $+\infty$ ). Прямая называется «заметаящей прямой». С этой прямой связана некоторая структура данных  $L$ , называемая «статусом заметающей прямой», которая описывает упорядоченное множество отрезков, пересекаемых прямой в каждый момент времени. Очевидно, что статус представляет собой динамическую (т.е. изменяющуюся во времени) структуру данных.

Отметим, что изменения в статусе заметающей прямой происходят в некоторые дискретные моменты времени, когда происходят упомянутые выше следующие события: прямая пересекает левый или правый конец отрезка, либо точку пересечения двух отрезков. В эти моменты меняется множество пересекаемых отрезков, либо их упорядоченность. Значения абсциссы  $x$ , в которых происходят эти события, называются «точками событий». Поскольку с каждой точкой события связано значение  $x$  множество точек событий упорядоченно естественным образом по возрастанию  $x$ . Множество точек событий будем описывать структурой данных  $E$ , называемой «перечнем событий».

В алгоритмах, использующих идею плоского заметания, при появлении точки события происходит выполнение трех видов действий:

1. Редактирование перечня событий: занесение только что обнаруженных точек событий, удаление текущей точки события, а также всех других, которые стали ненужными.
2. Редактирование статуса заметающей прямой для представления изменений его состояния из-за нового положения прямой.
3. Решение остальных вопросов задачи, зависящих от конкретной постановки.

При плоском заметании общая задача на плоскости сводится к конечной серии аналогичных задач в одномерном пространстве на заметающей прямой. Эти задачи обычно решаются проще. А поскольку различия задач, относящихся к соседним событиям, обычно невелики, то при решении одномерных задач можно использовать одни и те же данные, что позволяет конструировать эффективные алгоритмы.

### 4. Структуры данных

Рассмотрим структуру  $L$ , описывающую статус заметающей прямой. Для нее необходимо обеспечить выполнение следующих операций над отрезком  $s$ :

- 1) **ВСТАВИТЬ** ( $s, L$ ) – установка отрезка в статус;
- 2) **УДАЛИТЬ** ( $s, L$ ) – удаление отрезка из статуса;

- 3) *НАД* ( $s, L$ ) – найти отрезок в статусе, смежный с  $s$  и лежащий выше его;
- 4) *ПОД* ( $s, L$ ) – найти отрезок в статусе, смежный с  $s$  и лежащий ниже его.

Структура данных, позволяющая выполнять эти операции, известна как «словарь». Она может быть реализована в виде 2-3-дерева или АВЛ-дерева и в этом случае каждая из указанных операций будет выполняться за время  $O(m)$ , где  $m$  – максимальное число отрезков, находящихся одновременно в статусе.

Структура данных  $E$  для работы с перечнем событий должна обеспечить выполнение следующих операций над токами событий  $x$ :

- 1) *MIN* ( $E$ ) – определить минимальный элемент в  $E$  и удалить его;
- 2) *ВСТАВИТЬ* ( $x, E$ ) – вставить абсциссу  $x$  в упорядоченное множество точек событий;
- 3) *ЧЛЕН* ( $x, E$ ) – узнать, является ли абсцисса  $x$  членом множества  $E$ .

Это множество операций обеспечивается структурой данных «очередь с приоритетом». Она также может быть реализована в виде 2-3-дерева или АВЛ-дерева и тогда эти операции будут выполняться за время  $O(m)$ , где  $m$  – максимальное число событий в перечне событий.

## 5. Алгоритм

Алгоритм включает следующие элементы:

- 1) Упорядочение концов отрезков (сортировка  $2N$  точек) за время  $O(N \log N)$ .
- 2) Корректировка структуры  $L$  в каждой точке события за время  $O(\log N)$ .
- 3) Все пары отрезков, которые становятся смежными при корректировке  $L$ , проверяются на пересечение за время  $O(1)$ .
- 4) Если обнаруживается пересечение, то оно регистрируется и его абсцисса вставляется в перечень событий за время  $O(\log M)$ , где  $M$  – общее число событий, т.е.  $2N+K$ ,  $K$  – число пересечений отрезков.

Общее время, необходимое в этом случае для решения задачи составляет

$$O(N \log N) + (2N+K) * O(\log N) + (2N+K) * O(\log (2N+K))$$

или  $O((N+K) \log N)$ .

Эта оценка относится к тому случаю, когда необходимо найти все  $K$  пересечений в множестве отрезков. В том случае, когда нужно лишь установить факт пересечения, время решения составит  $O(N \log N)$ .

Ниже приводится текст алгоритма на псевдокоде для нахождения всех пересечений множества отрезков.

PROCEDURE Пересечение Отрезков;

1. BEGIN сортируем  $2N$  концов отрезков лексикографически по  $x$  и  $y$  и помещаем их в очередь с приоритетом  $E$ ;
2.  $A := \emptyset$ ; //  $A$  – множество пересечений отрезков
3. WHILE  $E \neq \emptyset$  DO
4. BEGIN  $p := \text{MIN}(E)$ ; // Очередное «текущее» событие
5. IF ( $p$  – левый конец отрезка) THEN
6. BEGIN  $s := \text{отрезок}$ , концом которого является  $p$ ;
7. ВСТАВИТЬ ( $s, L$ );
8.  $s_1 := \text{НАД}(s, L)$ ;
9.  $s_2 := \text{ПОД}(s, L)$ ;
10. IF ( $s_1$  пересекает  $s$ ) THEN поместить ( $s_1, s$ ) в  $A$ ;
11. IF ( $s_2$  пересекает  $s$ ) THEN поместить ( $s, s_2$ ) в  $A$ ;

```

12.   END
      ELSE IF (p - правый конец отрезка) THEN
13.   BEGIN s:=отрезок, концом которого является p;
14.         s1:=НАД(s,L);
15.         s2:=ПОД(s,L);
16.         IF (s1 пересекает s2 справа от p) THEN
              поместить (s1,s2) в A;
17.         УДАЛИТЬ(s,L);
18.   END
      ELSE // p - точка пересечения отрезков
19.   BEGIN
20.         (s1,s2):=отрезки, пересекающиеся в p;
              // здесь s1=НАД(s,L) s2=ПОД(s,L) слева от p
21.         s3:=НАД(s1,L);
22.         s4:=ПОД(s2,L);
23.         IF (s3 пересекает s2) THEN поместить (s3,s2) в A;
24.         IF (s1 пересекает s4) THEN поместить (s1,s4) в A;
25.         поменять местами s1 и s2 в L;
26.   END;
27.   // теперь найденные пересечения нужно обработать
28.   WHILE (A≠∅) DO
29.   BEGIN взять очередную пару (s1,s2) из A;
30.         x:=абсцисса точки пересечения (s1,s2);
31.         IF ЧЛЕН(x,E)=FALSE THEN
32.         BEGIN вывести(s1,s2); //например, на печать
33.               ВСТАВИТЬ(x,E);
34.         END;
35.   END;
36. END;
37. END;
38. END;

```