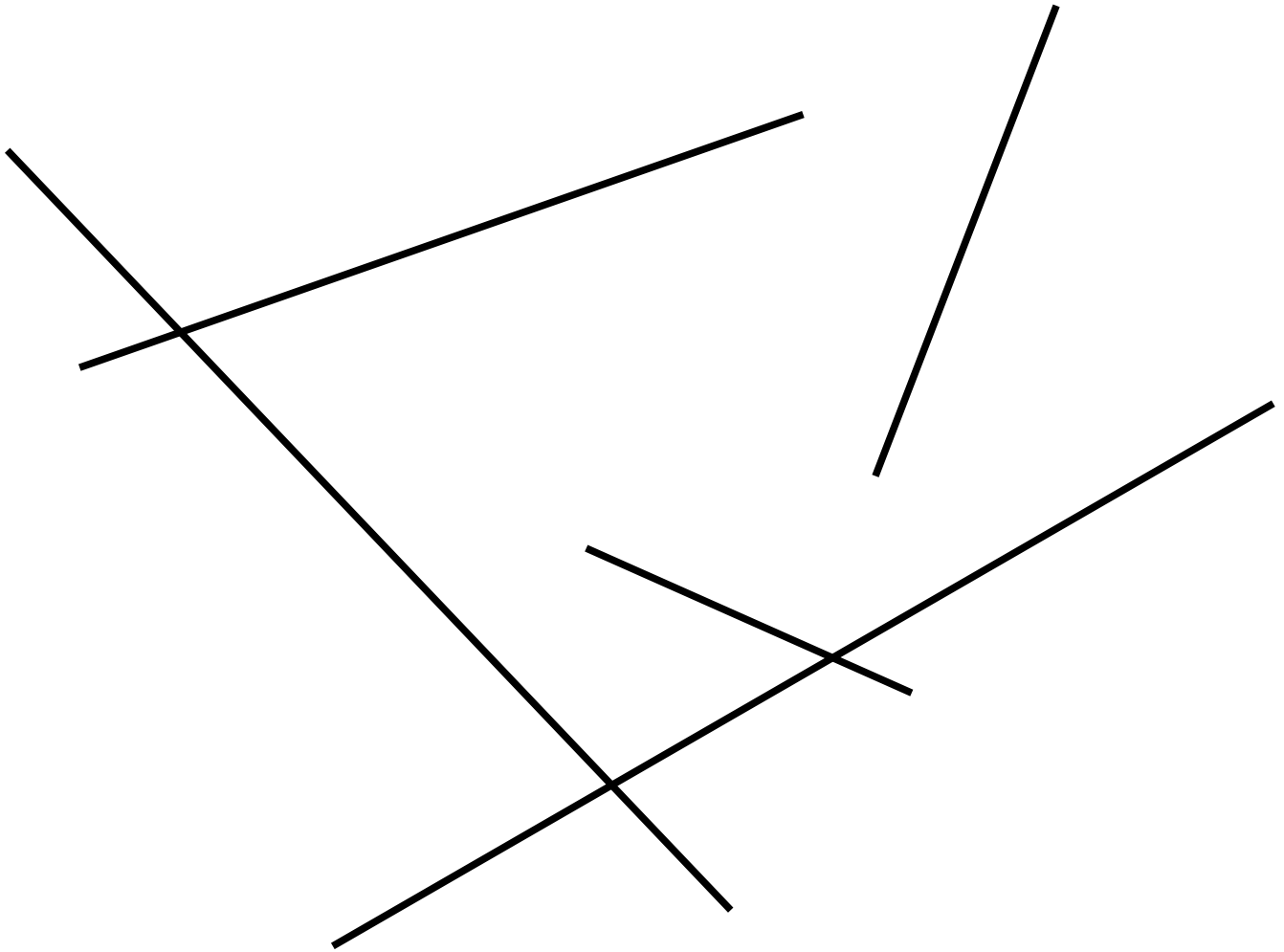
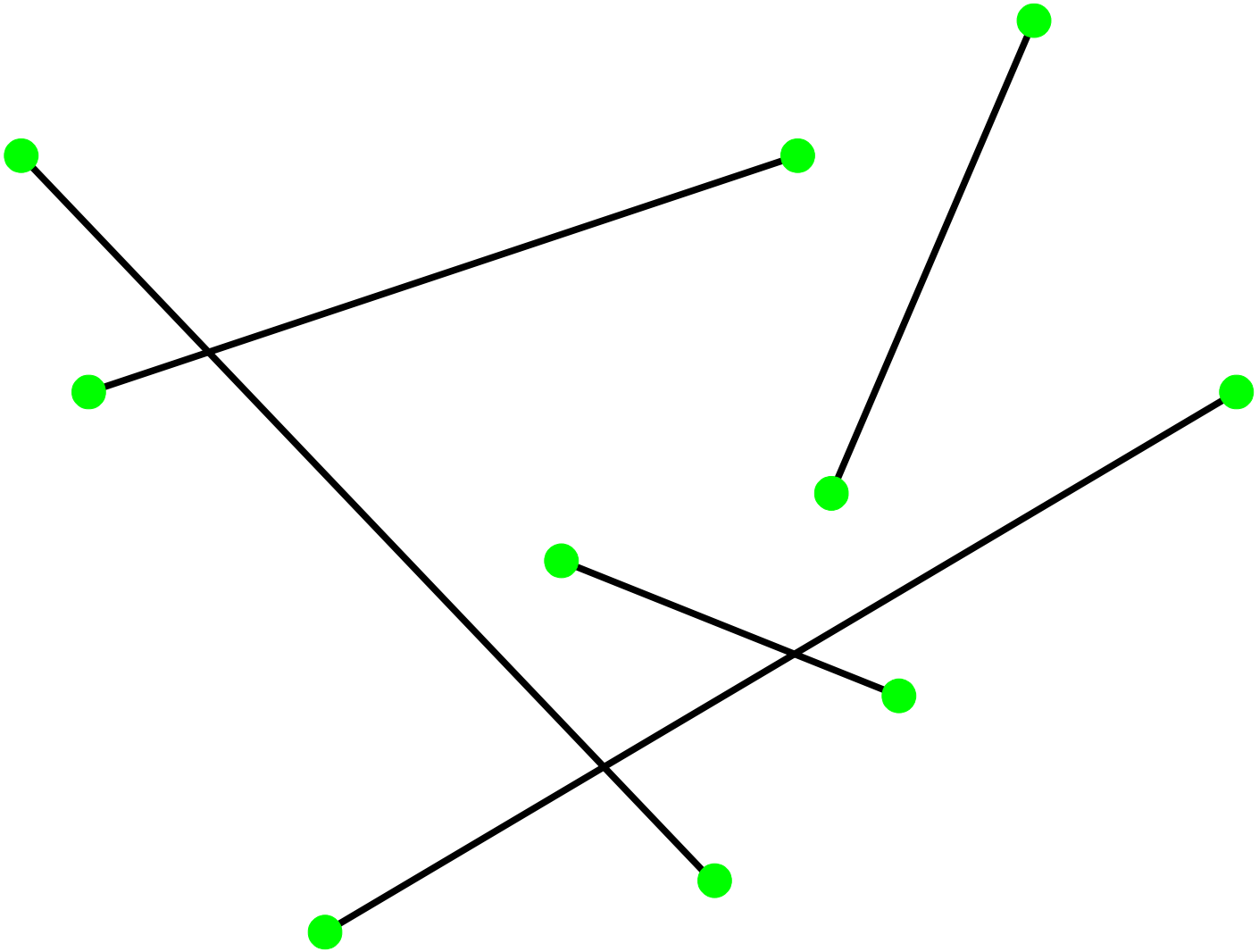
Schnittpunkte einer Menge von Strecken

1. Beschreibung der Aufgabenstellung

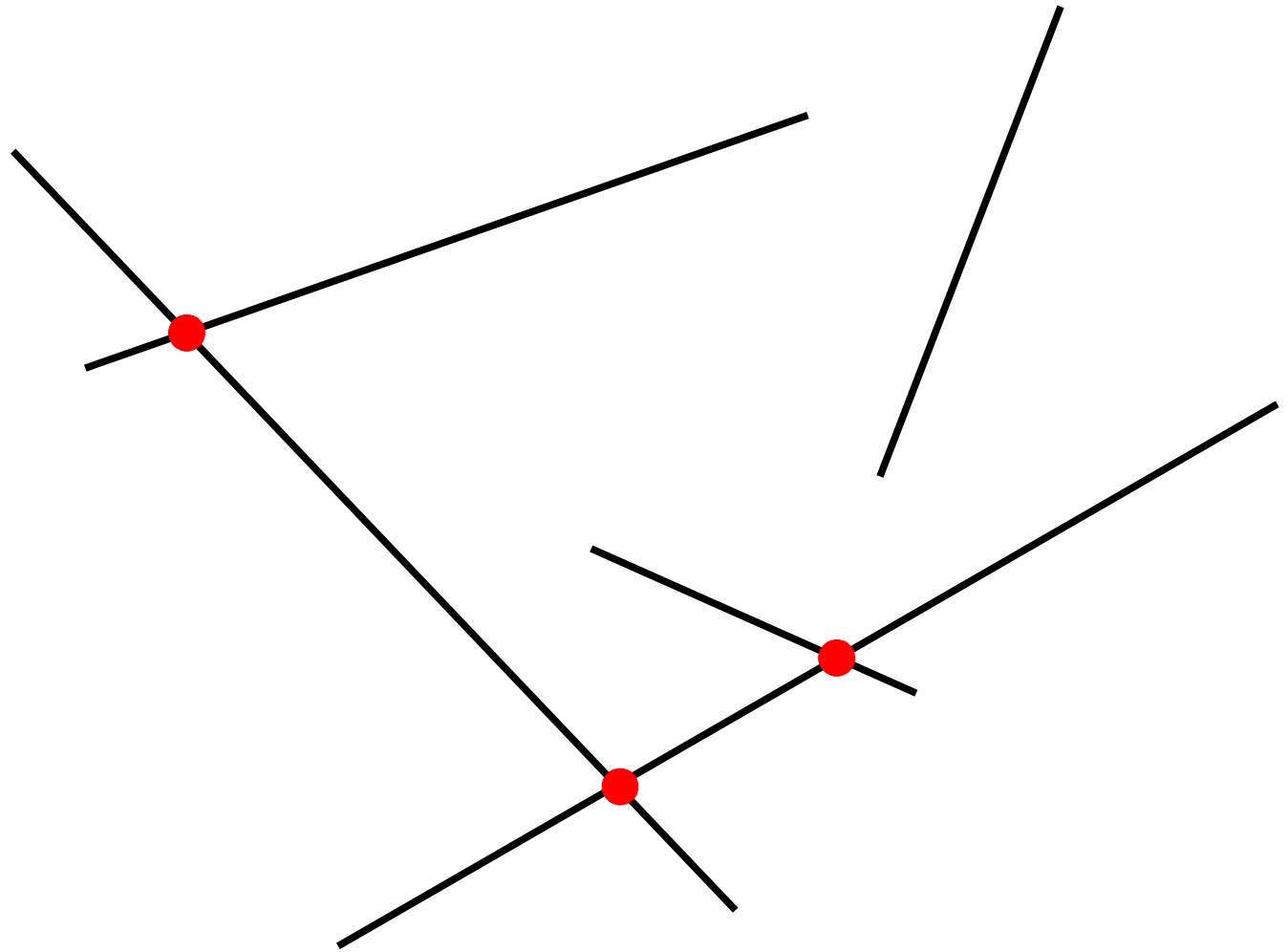
Gegebene Menge von Strecken



Strecke = zwei Endpunkte



Gesucht: Schnittpunkte



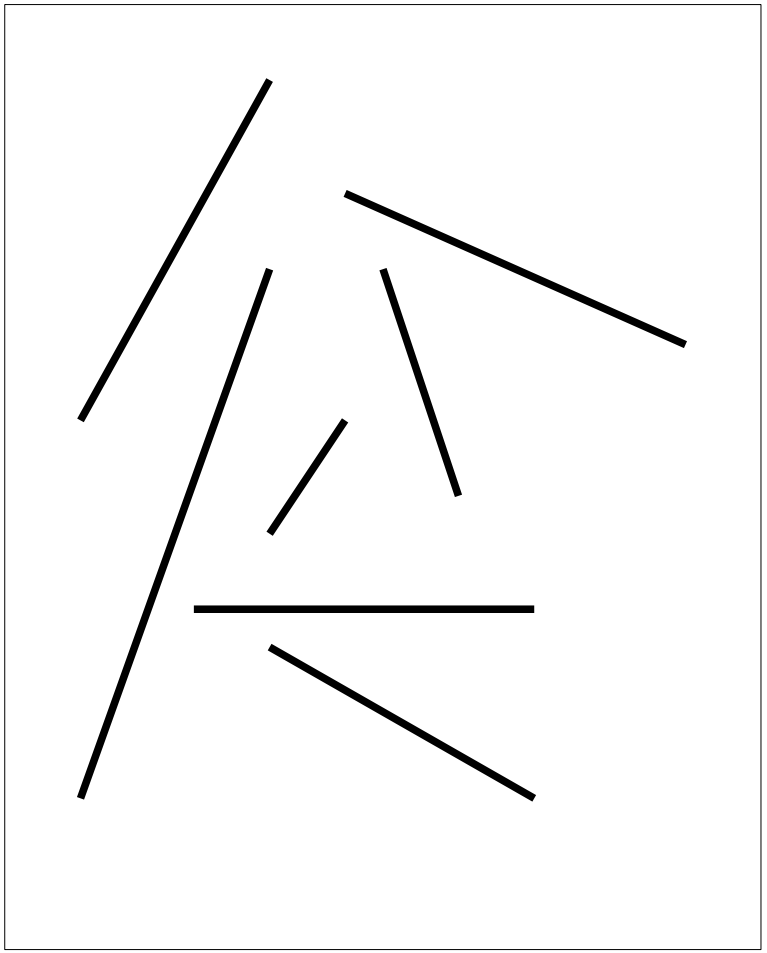
Berechtigte Fragen

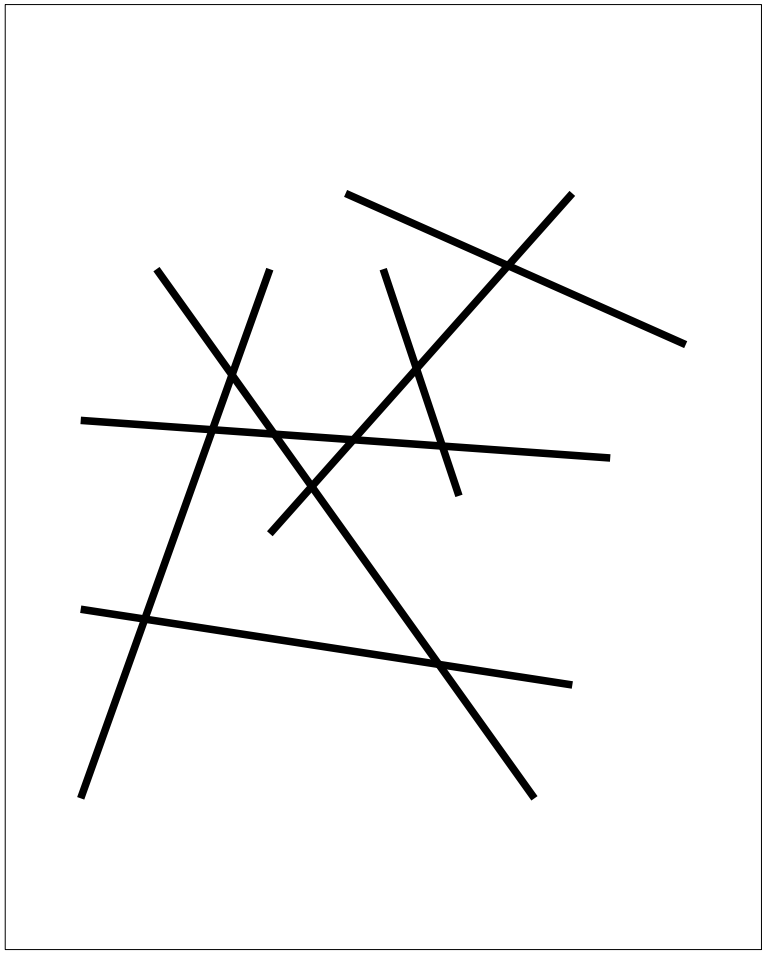
Wie kann man testen, ob sich zwei Strecken schneiden?

Warum geht man nicht einfach alle Paare von Strecken durch?

Ausgabe-Sensitivität

Wenn die Ausgabe klein ist, soll die Rechenzeit nicht unnötig lang sein.





Einige vereinfachende Annahmen

Ein Endpunkt einer gegebenen Strecke ist niemals Element einer anderen gegebenen Strecke.

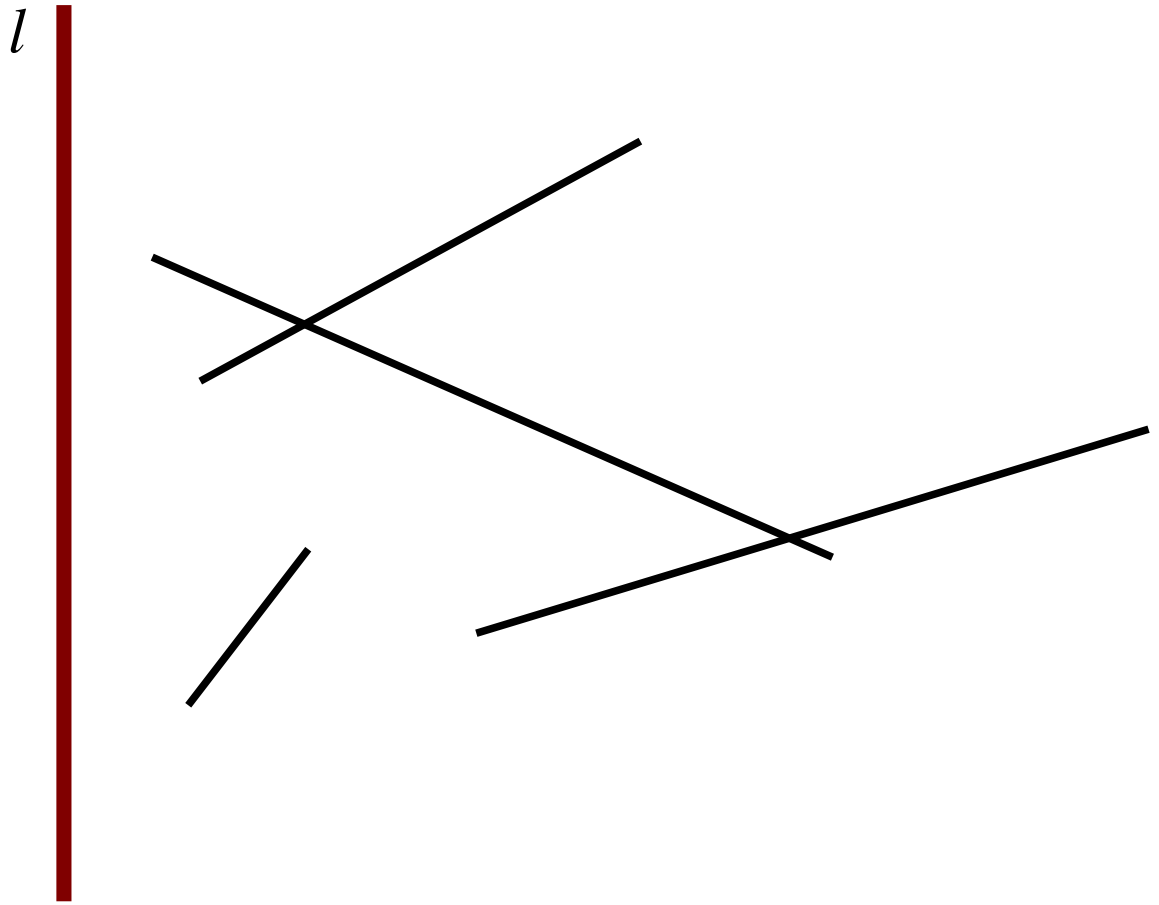
In einem Punkt schneiden sich höchstens zwei Strecken.

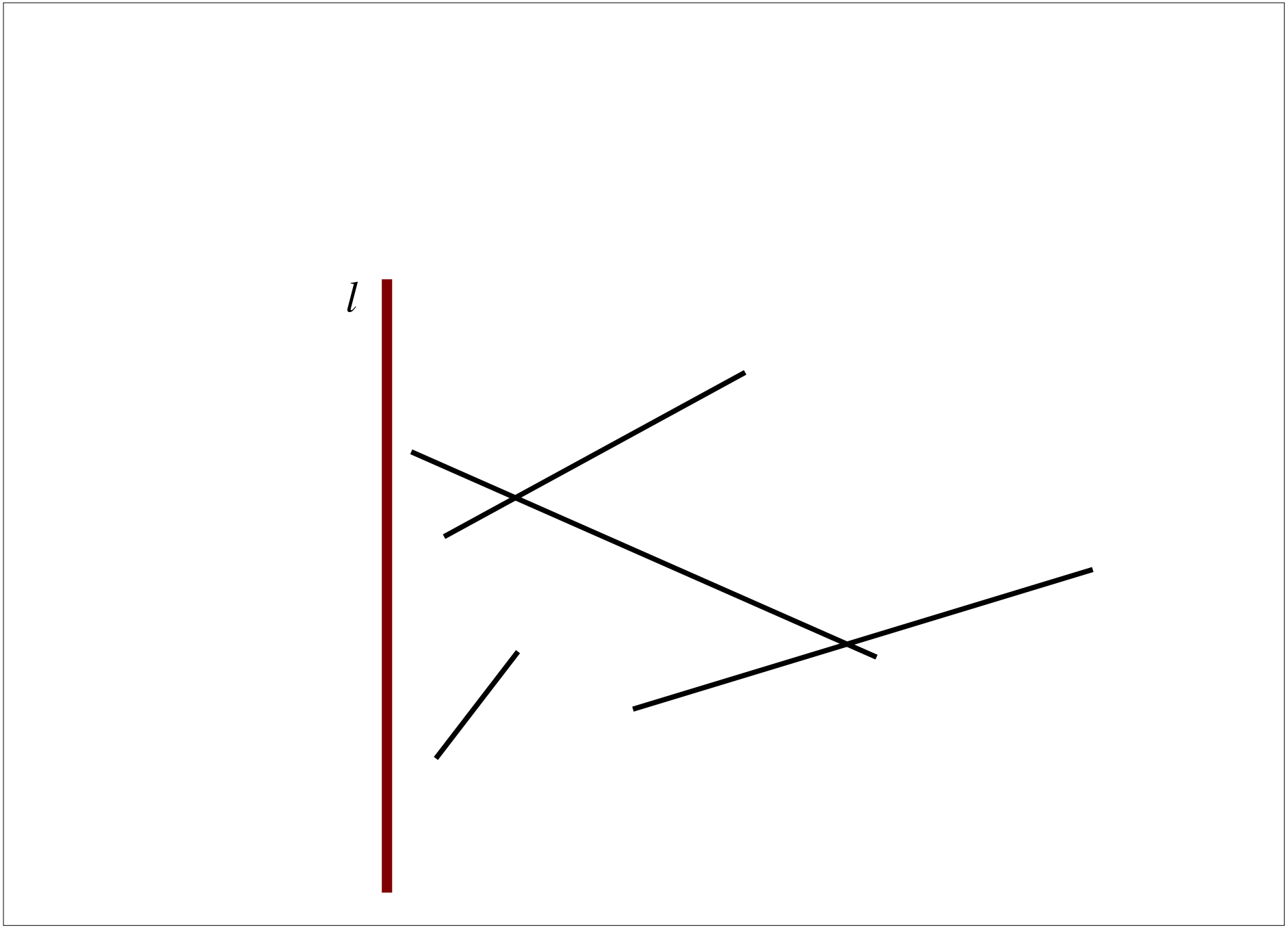
In der Menge, die alle Endpunkte und alle Schnittpunkte enthält, haben alle Elemente paarweise verschiedene x-Koordinaten.

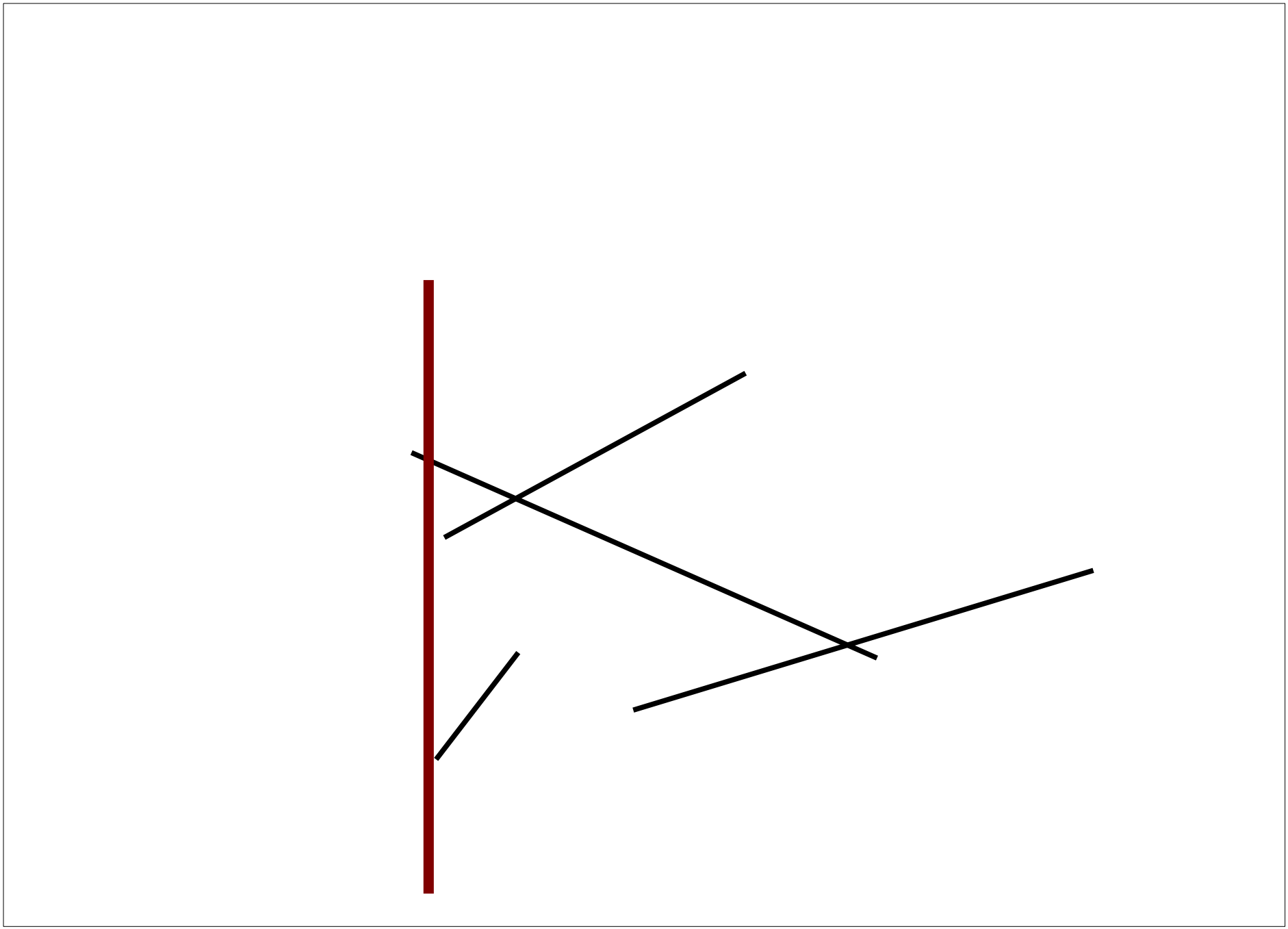
2. Beschreibung eines Algorithmus

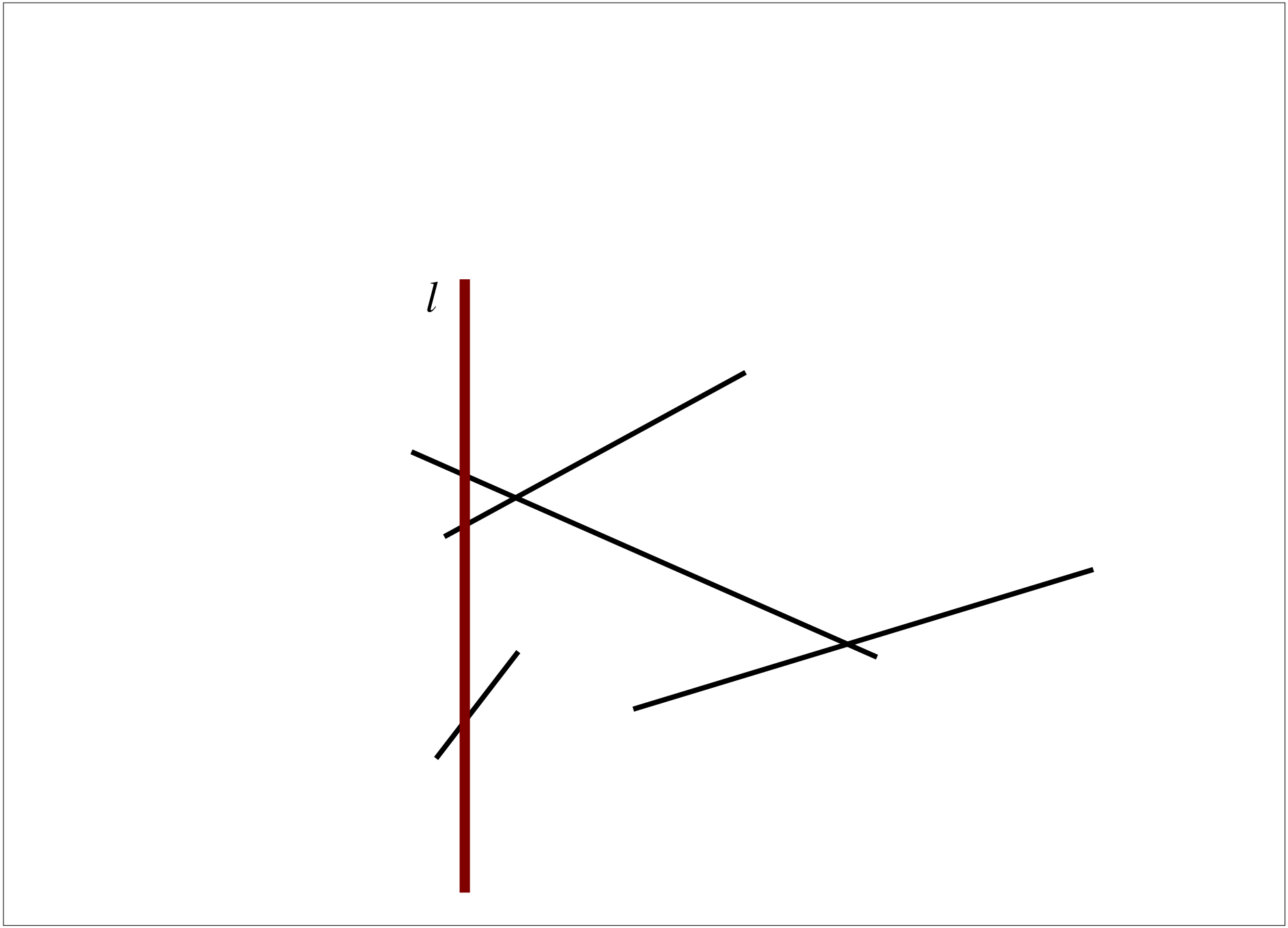
Idee: Gleitgeradenalgorithmus

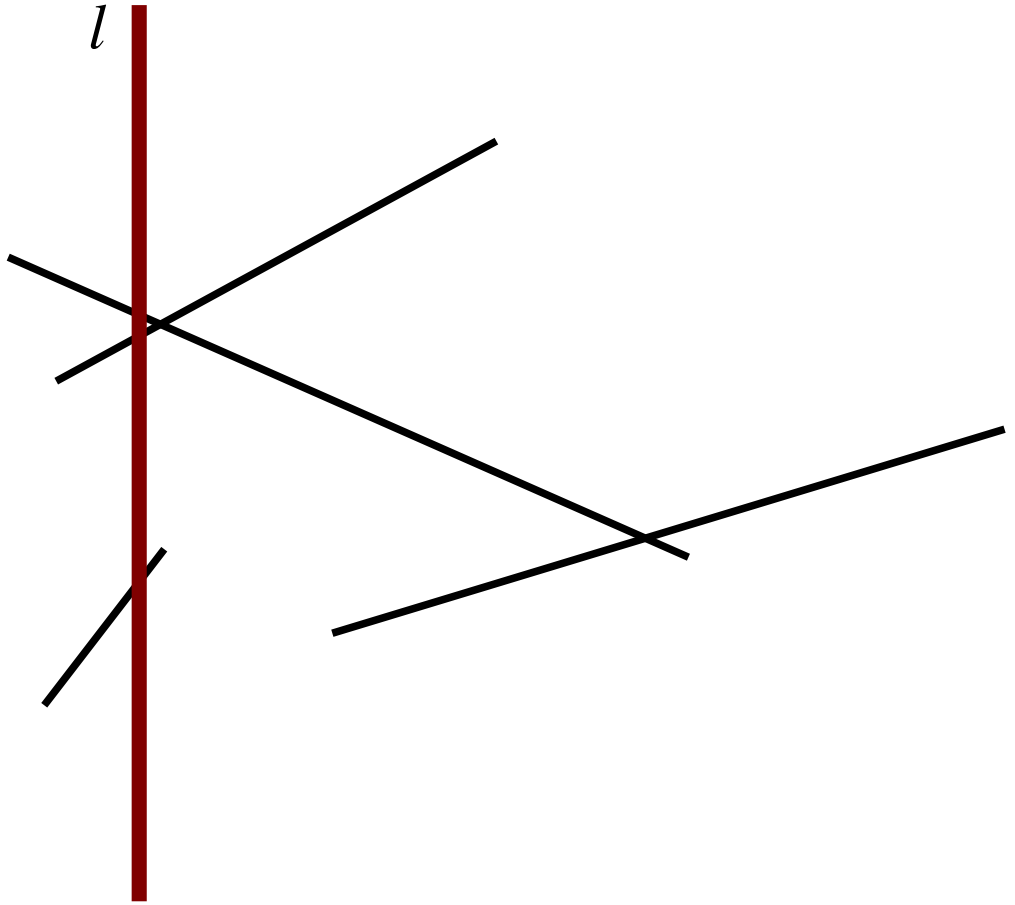
Eine senkrechte Gerade l bewegt sich kontinuierlich von links nach rechts über die Ebene.



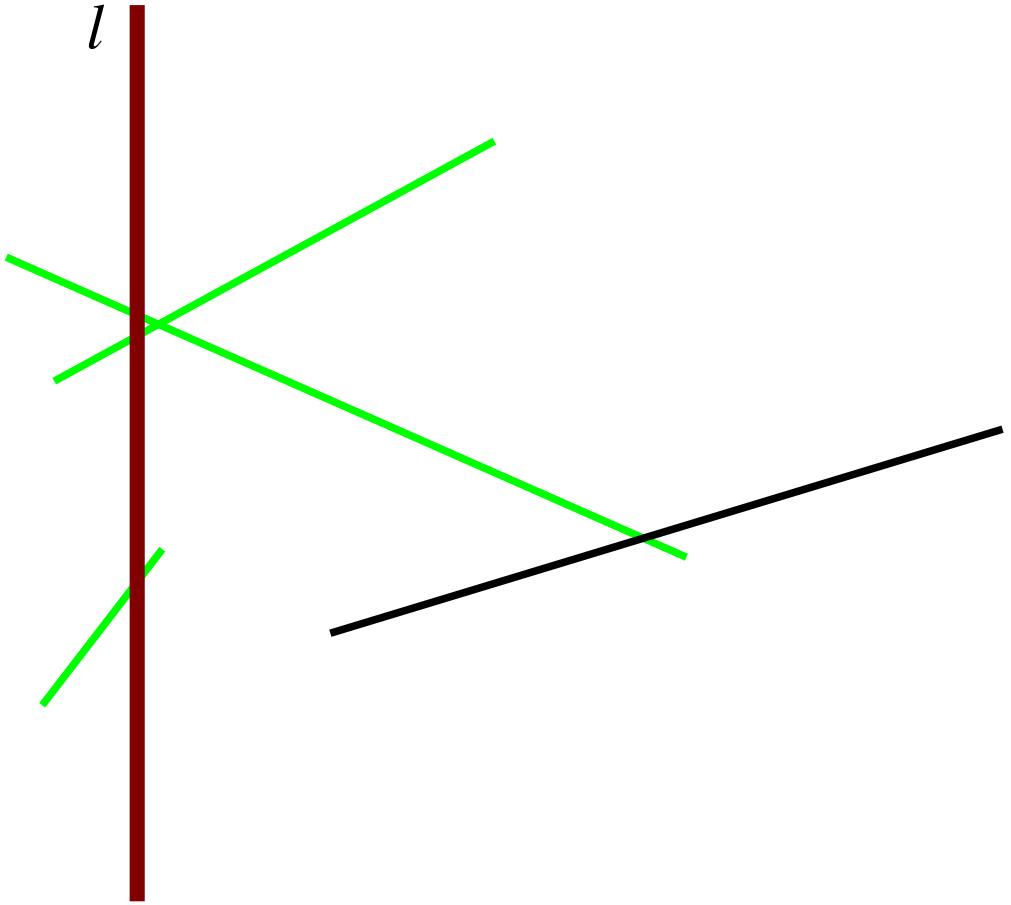

l



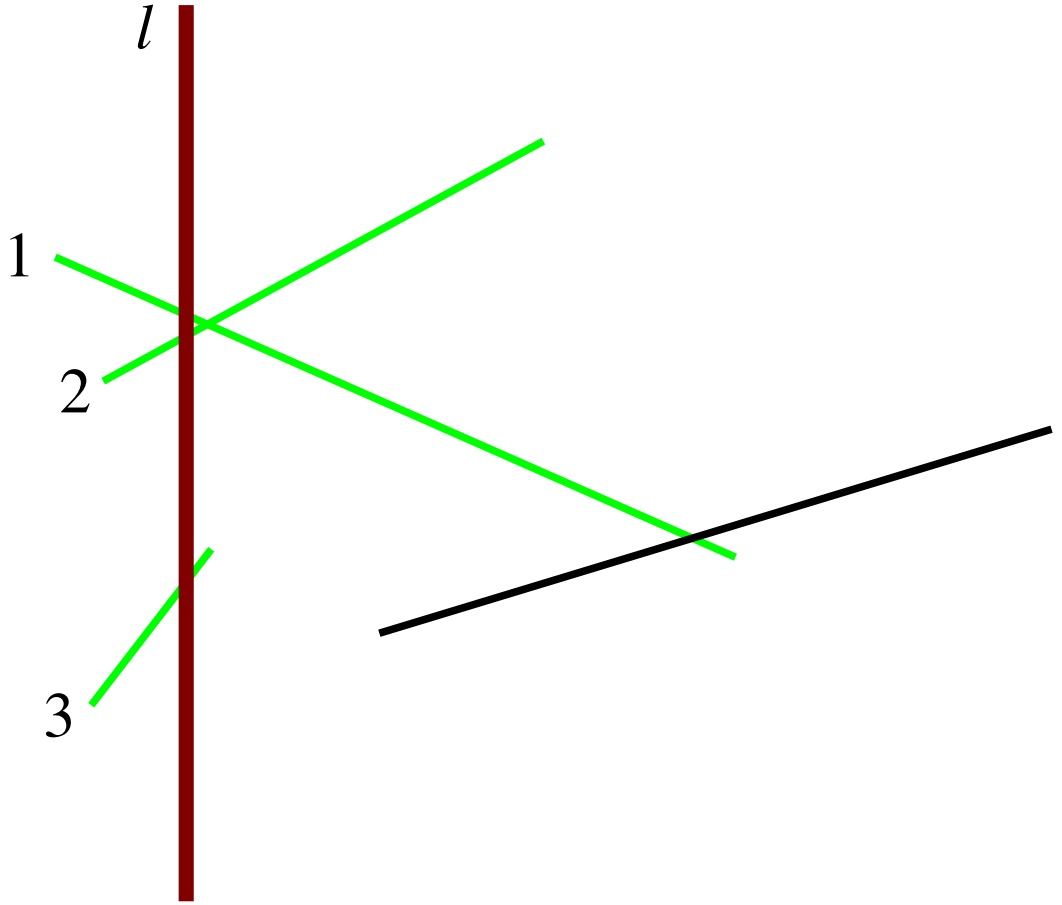

l


l

Zu jedem Zeitpunkt verwalten wir die Menge S der Strecken, die von l geschnitten werden.



Die Menge S ist immer bezüglich der Abfolge der Schnittpunkte auf l geordnet.

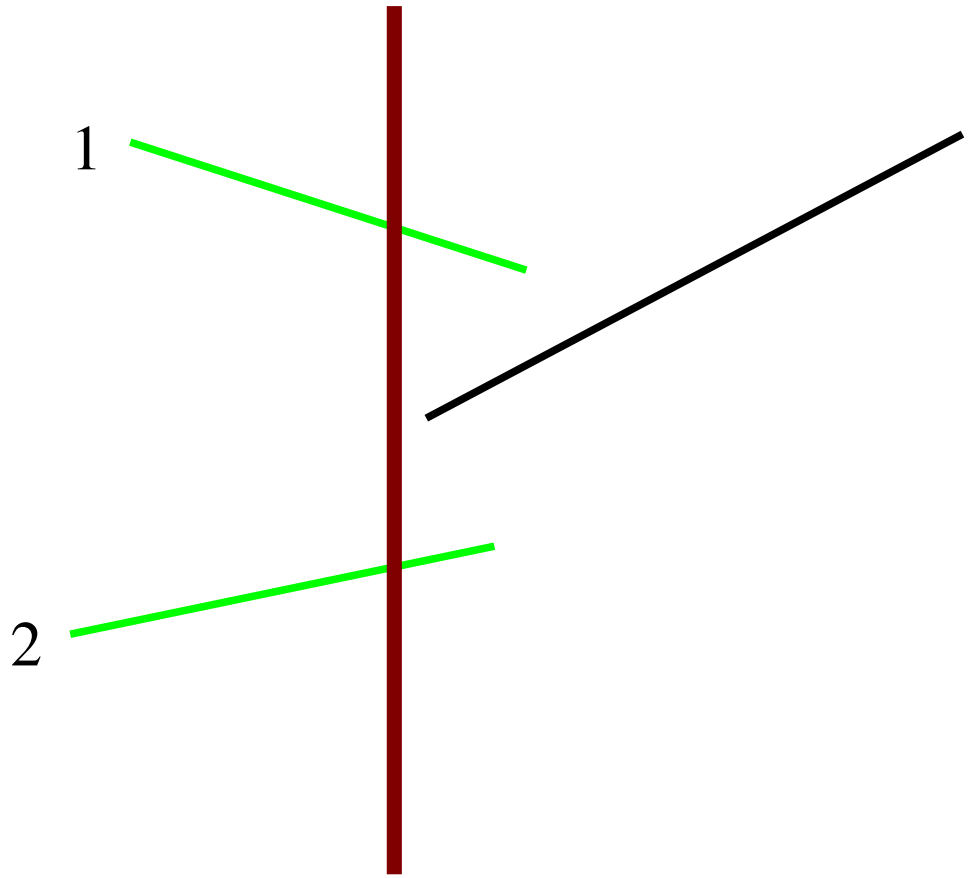


Wichtige Beobachtung

Offenbar ändert sich die Menge S nur an einer endlichen Menge E von **Ereignispunkten**.

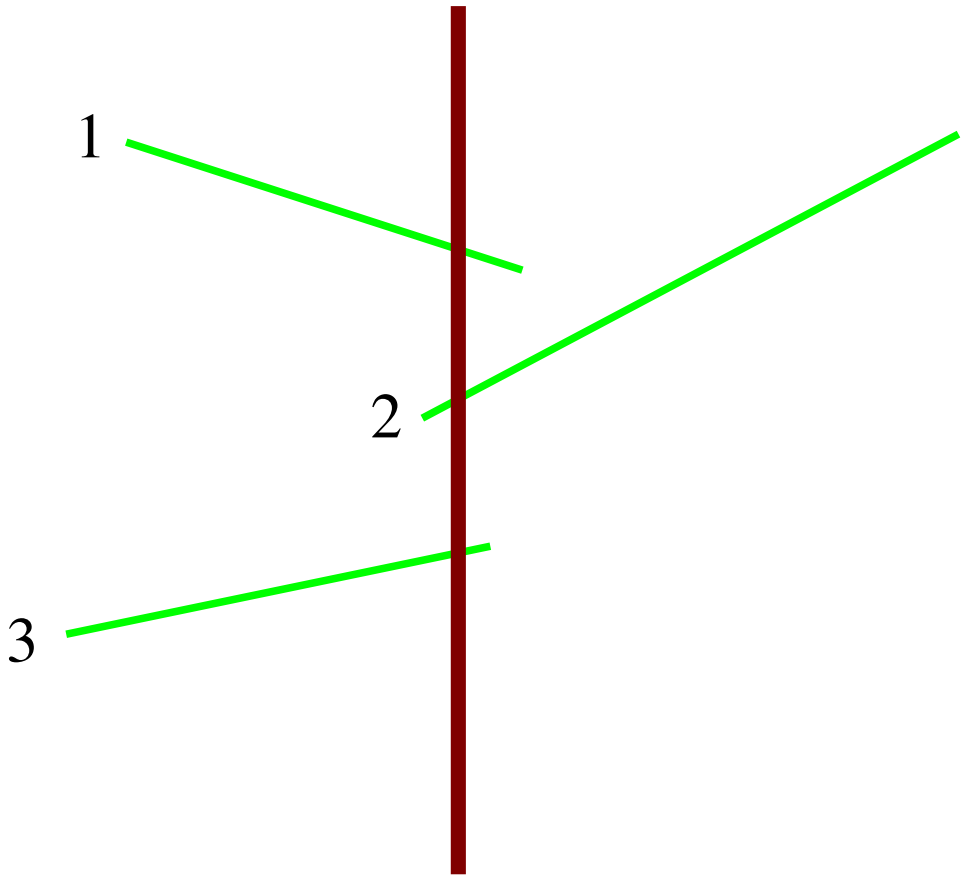
1. Typ von Ereignispunkt

Dort wo eine Strecke beginnt.



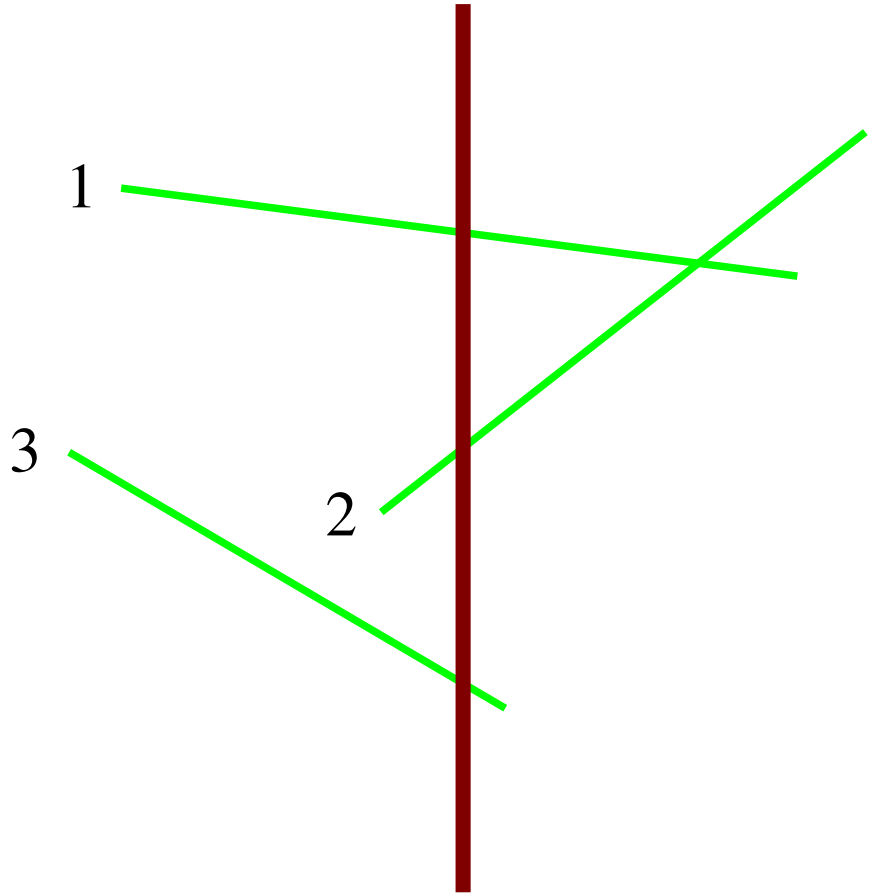
1. Typ von Ereignispunkt

Dort wo eine Strecke beginnt.



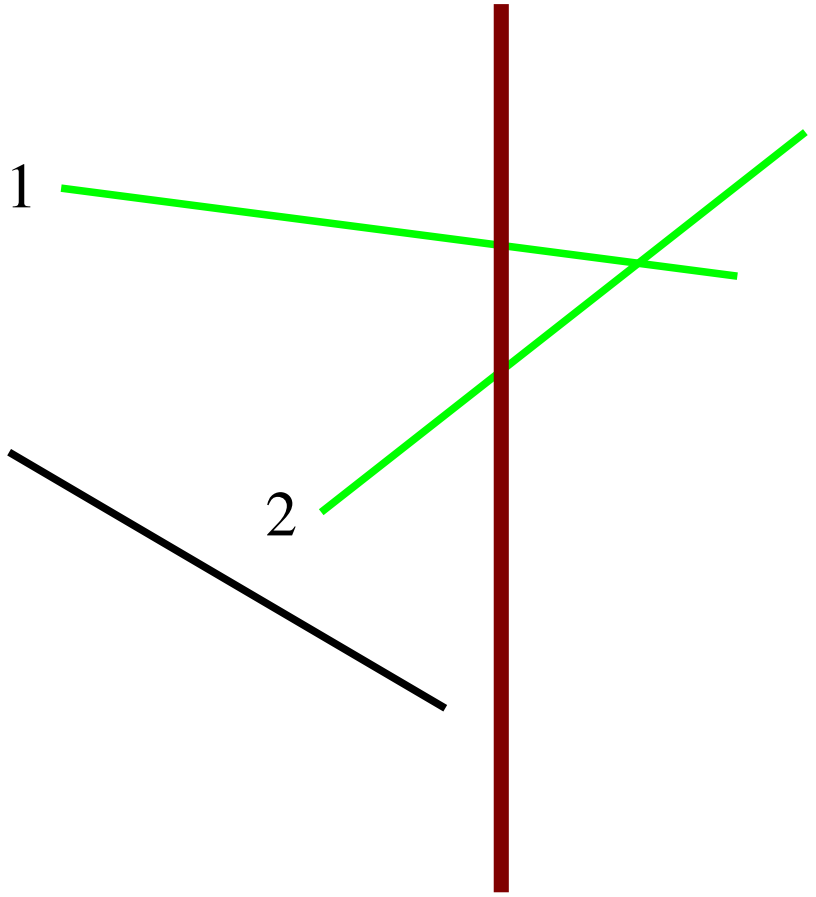
2. Typ von Ereignispunkt

Dort wo eine Strecke endet.



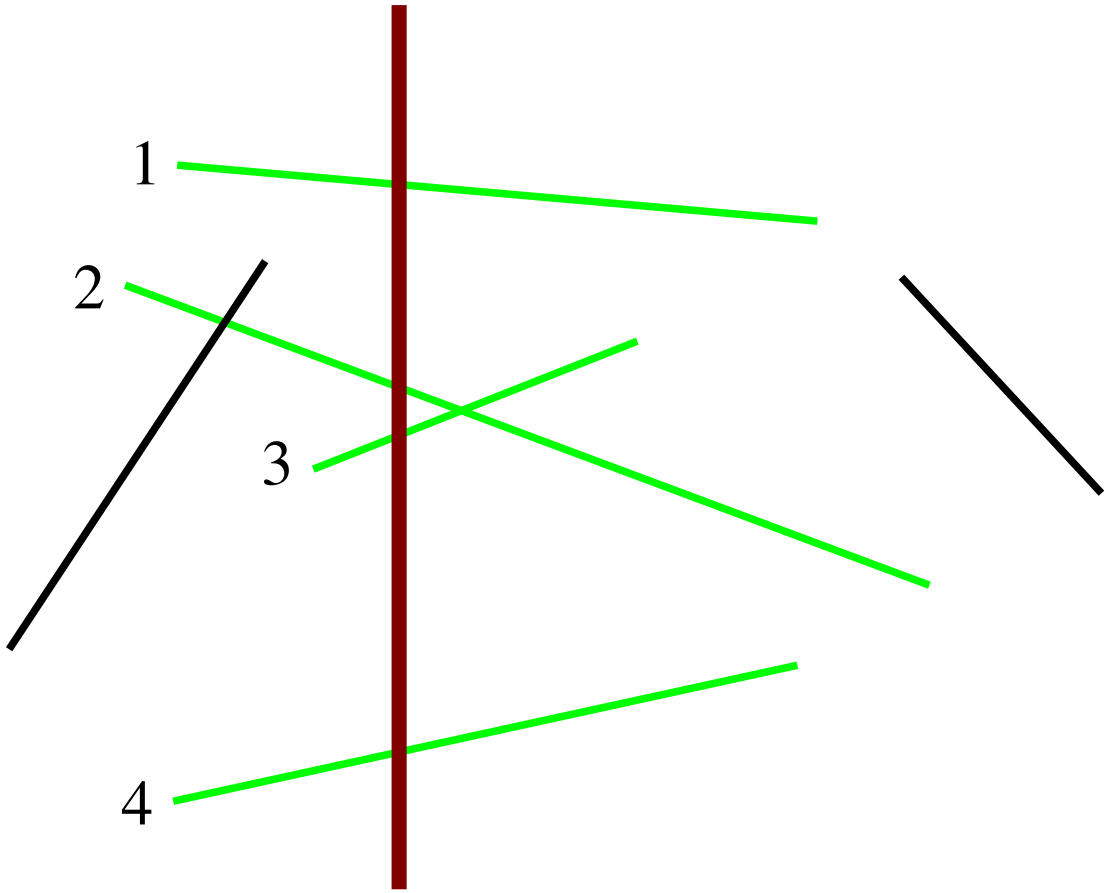
2. Typ von Ereignispunkt

Dort wo eine Strecke endet.



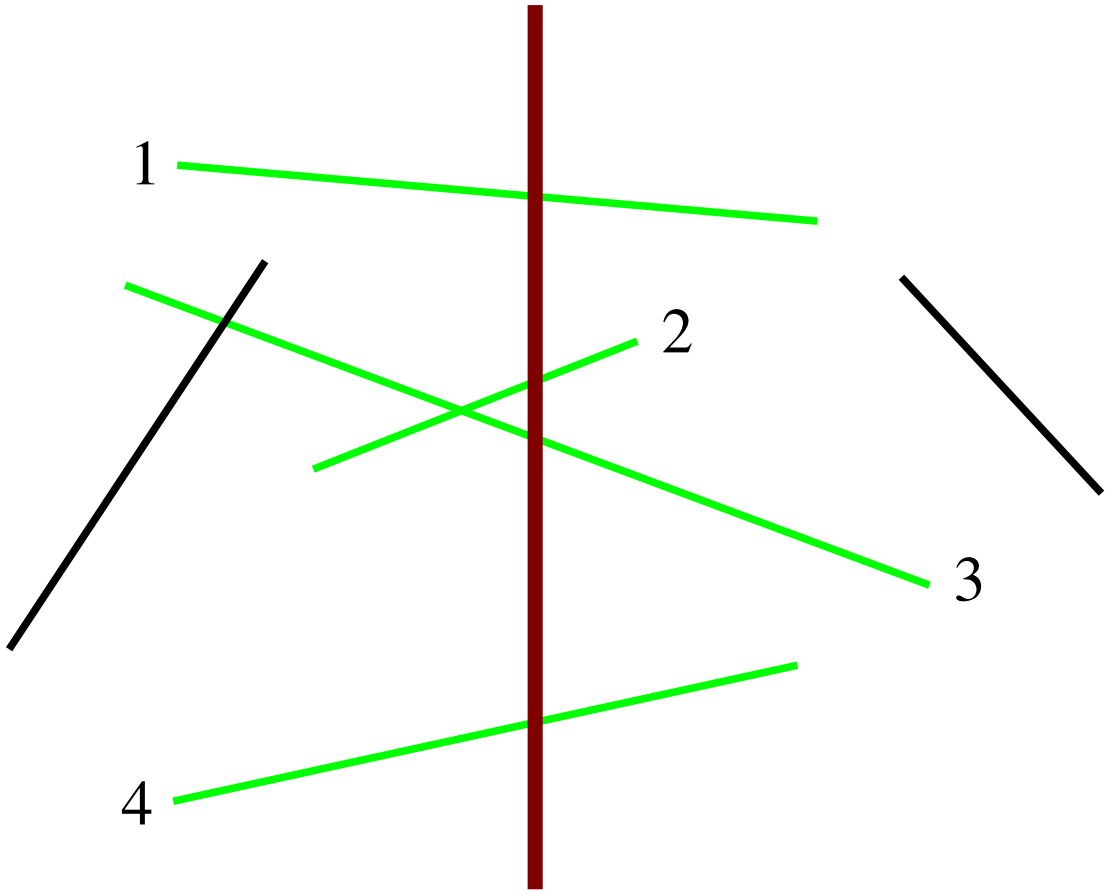
3. Typ von Ereignispunkt

Dort wo sich zwei Strecken schneiden.



3. Typ von Ereignispunkt

Dort wo sich zwei Strecken schneiden.



Verwaltung der Ereignispunkte

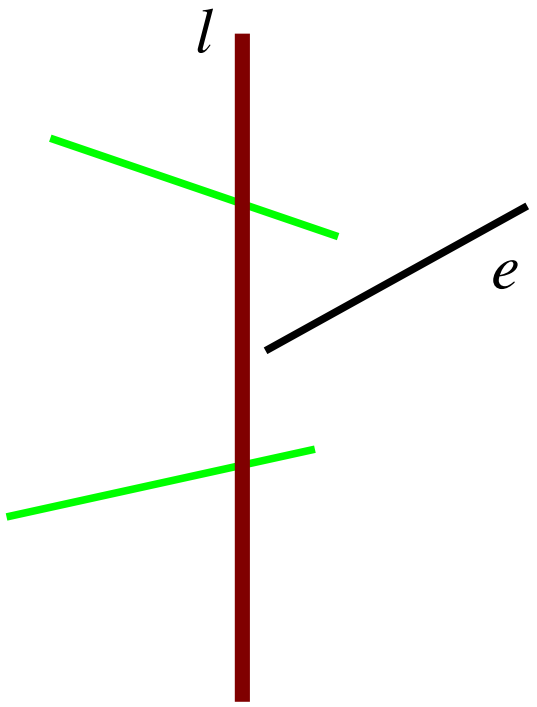
Problem: Wir kennen nicht die gesamte Menge E der Ereignispunkte von Anfang an, denn die Schnittpunkte zwischen den Strecken wollen wir ja gerade berechnen.

Aber: Wenn der nächste Ereignispunkt ein Schnittpunkt ist, dann sind die beiden sich schneidenden Strecken in S benachbart.

Deshalb verwalten wir sowohl die Menge S als auch die Menge E in einer Datenstruktur, in der wir effizient Elemente einfügen, suchen und löschen können.

Verarbeitung des 1. Typs von Ereignispunkten

1. Typ: eine Strecke beginnt



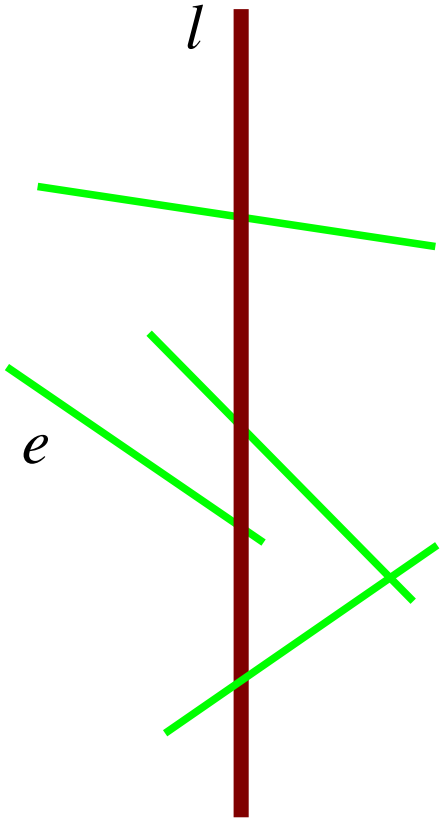
e in *S* einfügen

Auf Schnittpunkt zwischen *e* und dem Vorgänger von *e* rechts von *l* testen. Schnittpunkt ggfs. in *E* einfügen.

Auf Schnittpunkt zwischen *e* und dem Nachfolger von *e* rechts von *l* testen. Schnittpunkt ggfs. in *E* einfügen.

Verarbeitung des 2. Typs von Ereignispunkten

2. Typ: eine Strecke endet

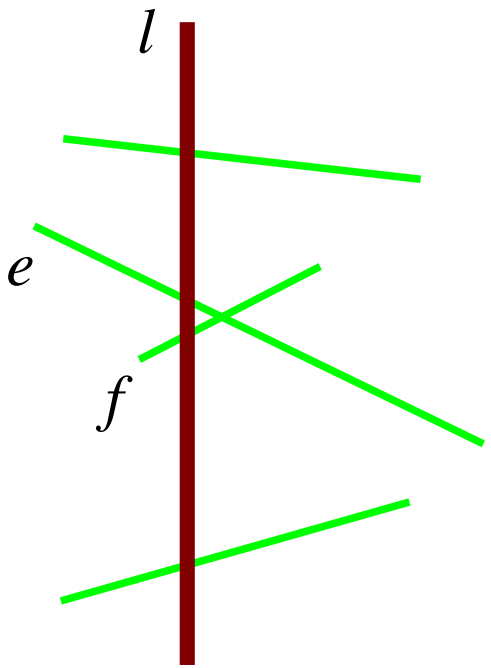


Auf Schnittpunkt zwischen dem Vorgänger und dem Nachfolger von *e* rechts von *l* testen.
Schnittpunkt ggfs. in *E* einfügen.

e aus *S* löschen

Verarbeitung des 3. Typs von Ereignispunkten

3. Typ: Schnittpunkt zwischen zwei Strecken



e und *f* aus *S* löschen.

e und *f* in *S* entsprechend der neuen Reihenfolge einfügen.

Auf Schnittpunkt zwischen *e* und dem Nachfolger von *e* rechts von *l* testen. Schnittpunkt ggfs. in *E* einfügen.

Auf Schnittpunkt zwischen *f* und dem Vorgänger von *f* rechts von *l* testen. Schnittpunkt ggfs. in *E* einfügen.

3. Analyse des Algorithmus

Laufzeit des Gleitgeradenalgorithmus

Sei n die Anzahl der gegebenen Stecken und k die Anzahl der Schnittpunkte.

Anzahl der Ereignispunkte: $O(n+k)$ mit $k \in O(n^2)$

Aufwand pro Ereignispunkt: $O(\log n)$

Laufzeit: $O((n+k)\log n)$

Speicherbedarf des Gleitgeradenalgorithmus

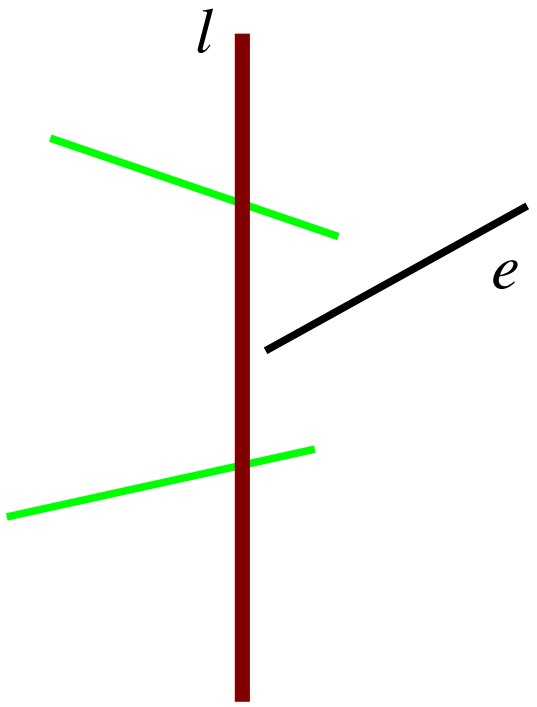
unschön: Speicherbedarf zur Verwaltung von E kann in $\Omega(n^2)$ liegen.

Idee: Wir verwalten in E nur die Schnittpunkte zwischen Strecken, die momentan entlang l benachbart sind.

Das sind dann natürlich immer nur $O(n)$.

Modifizierte Verarbeitung des 1. Typs von Ereignispunkten

1. Typ: eine Strecke beginnt



e in *S* einfügen

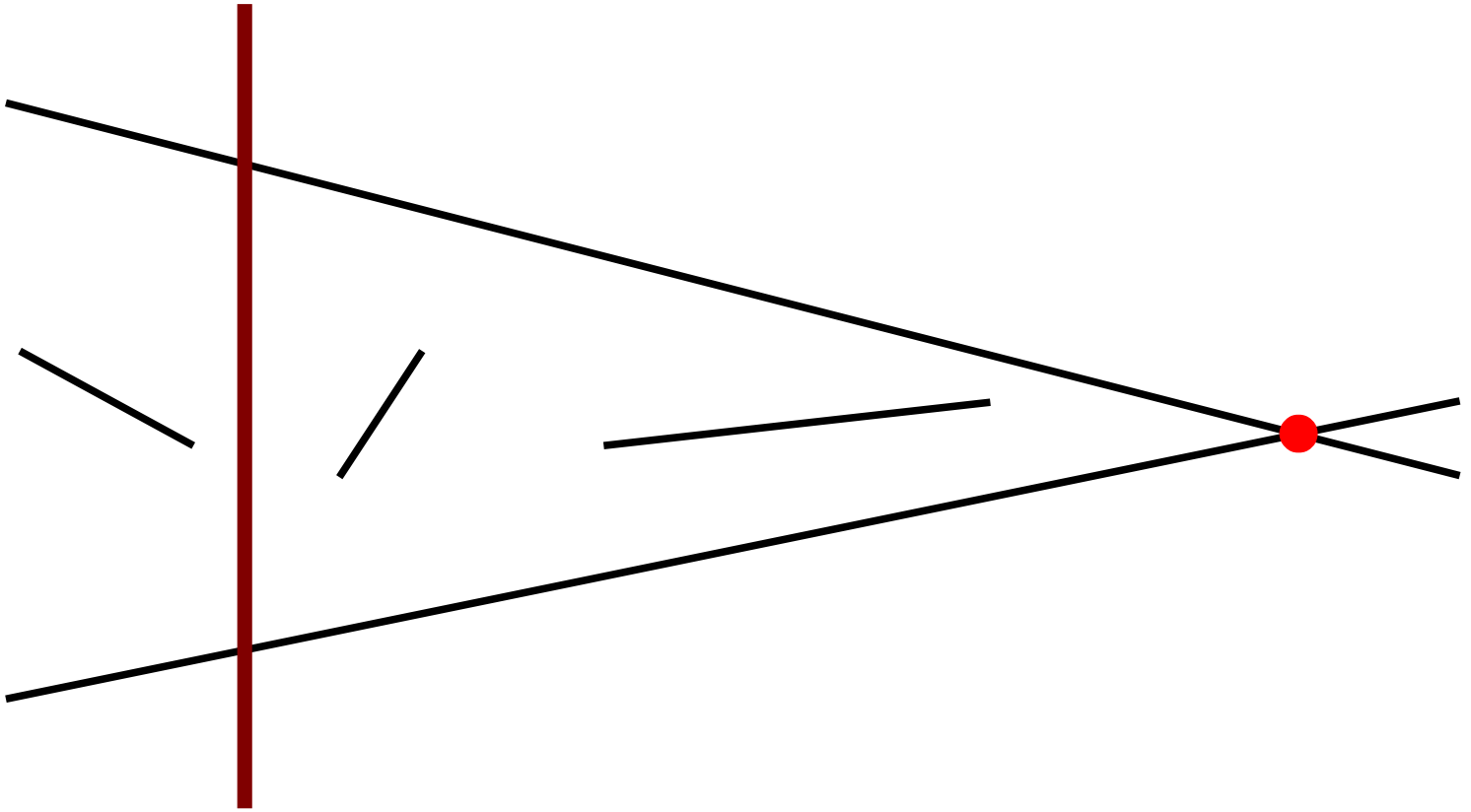
Falls es Schnittpunkt zwischen Vorgänger und Nachfolger von *e* rechts von *l* gibt, wird dieser aus *E* gelöscht.

Auf Schnittpunkt zwischen *e* und dem Vorgänger von *e* rechts von *l* testen. Schnittpunkt ggfs. in *E* einfügen.

Auf Schnittpunkt zwischen *e* und dem Nachfolger von *e* rechts von *l* testen. Schnittpunkt ggfs. in *E* einfügen.

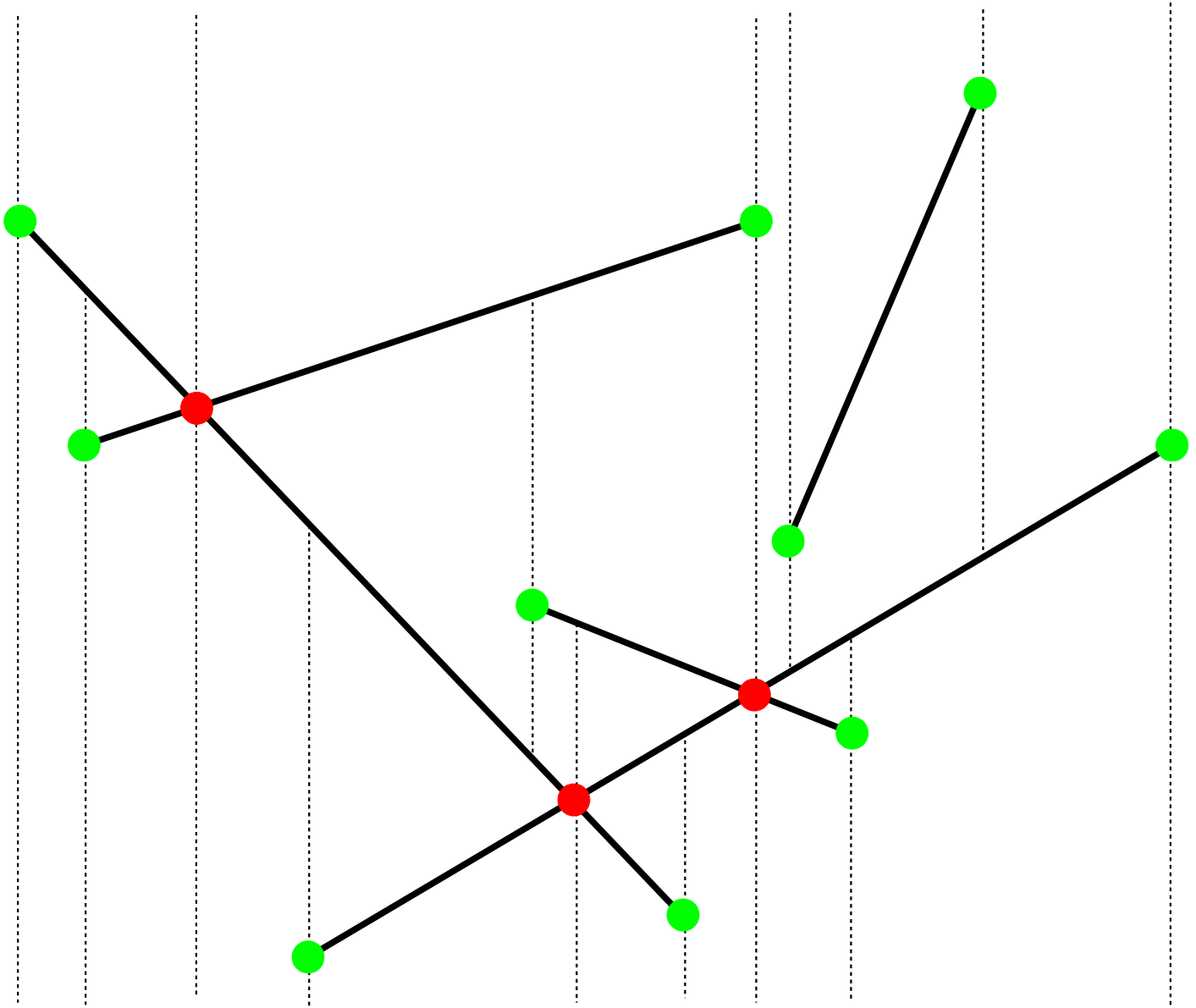
Wirkung der Modifikation

Jetzt wird ein Schnittpunkt unter Umständen mehrmals in E eingefügt und wieder gelöscht.



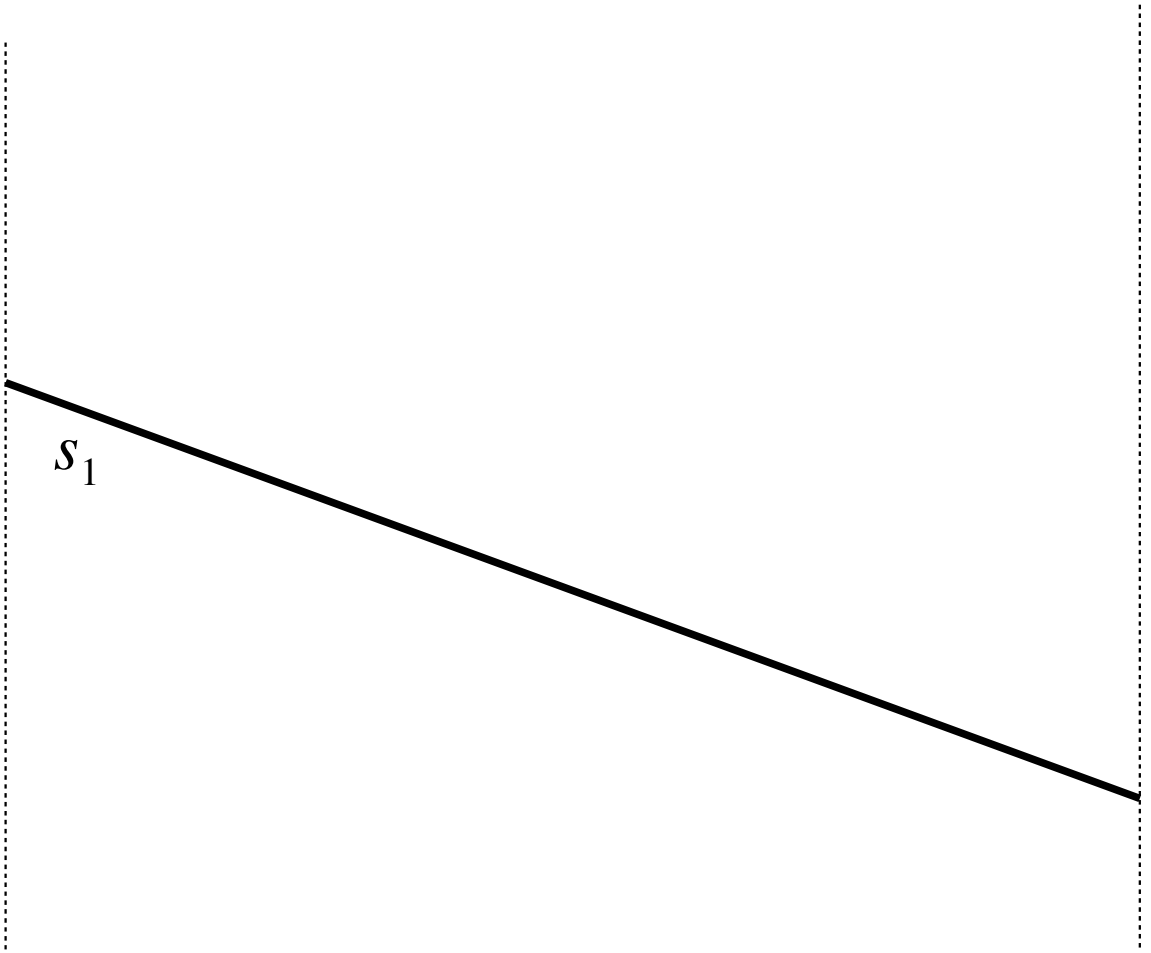
4. Lösung mit Hilfe einer Trapezzerlegung

Trapezzerlegung für eine Menge von Strecken

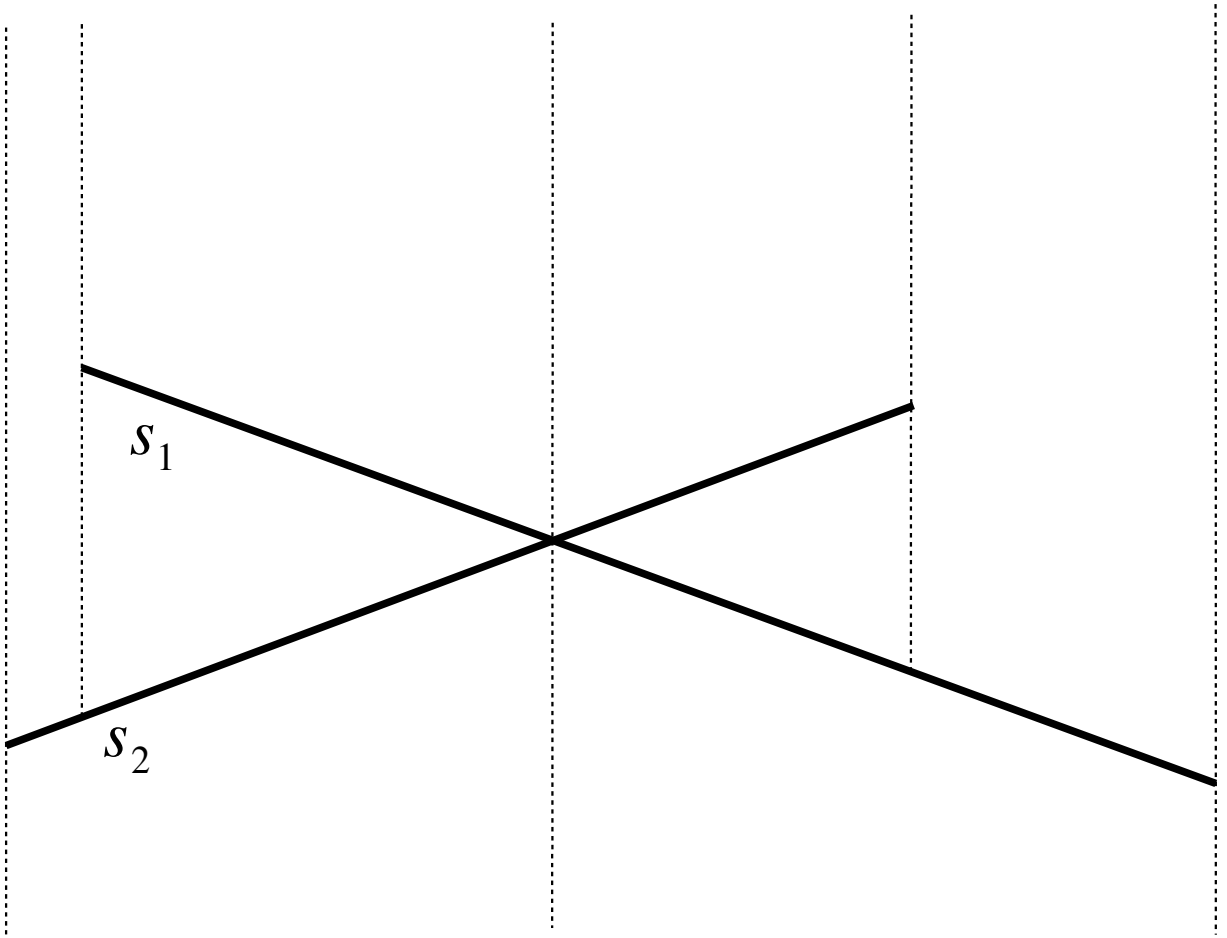


Inkrementelle Berechnung der Trapezzerlegung

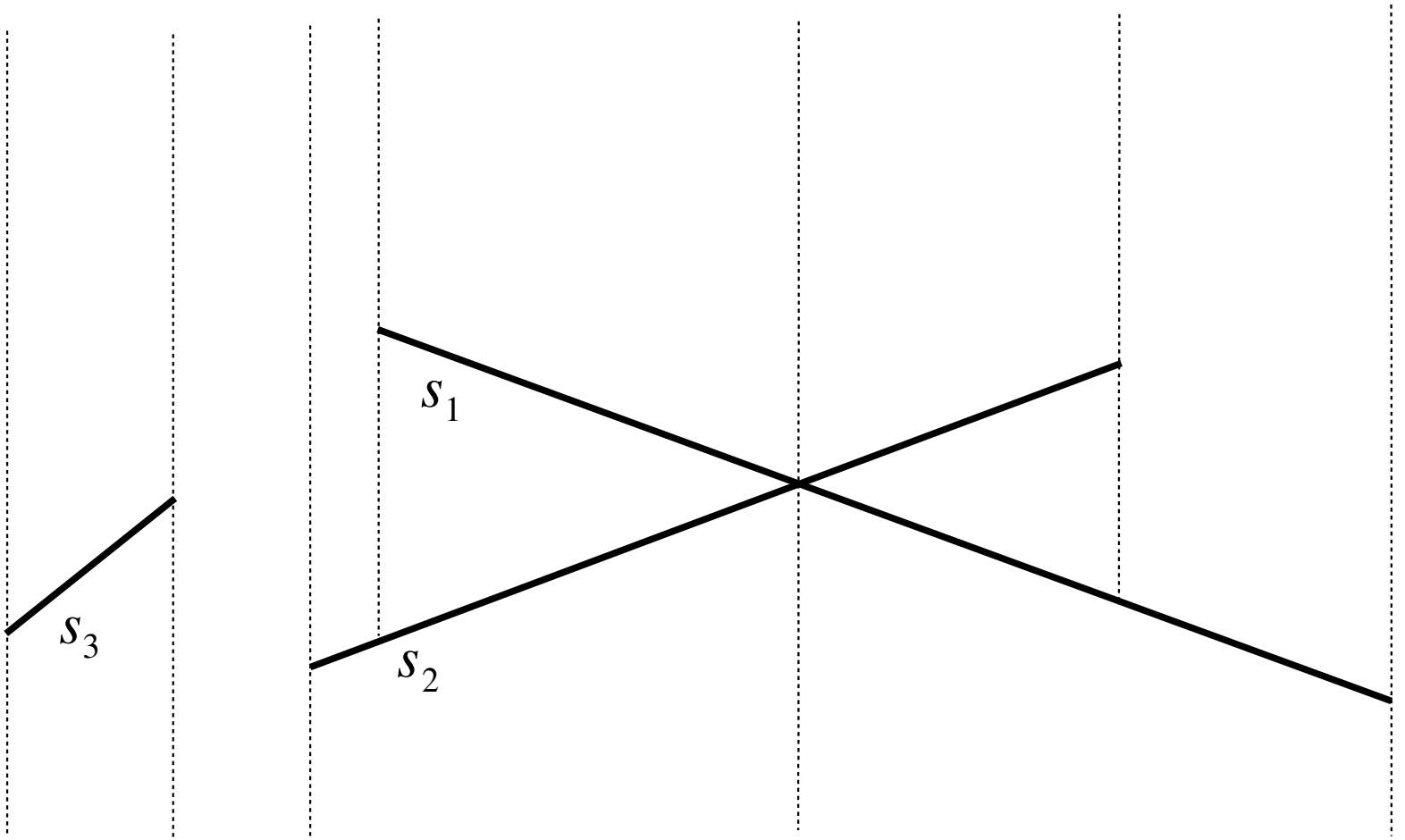
Zuerst wird die Trapezzerlegung von s_1 berechnet.



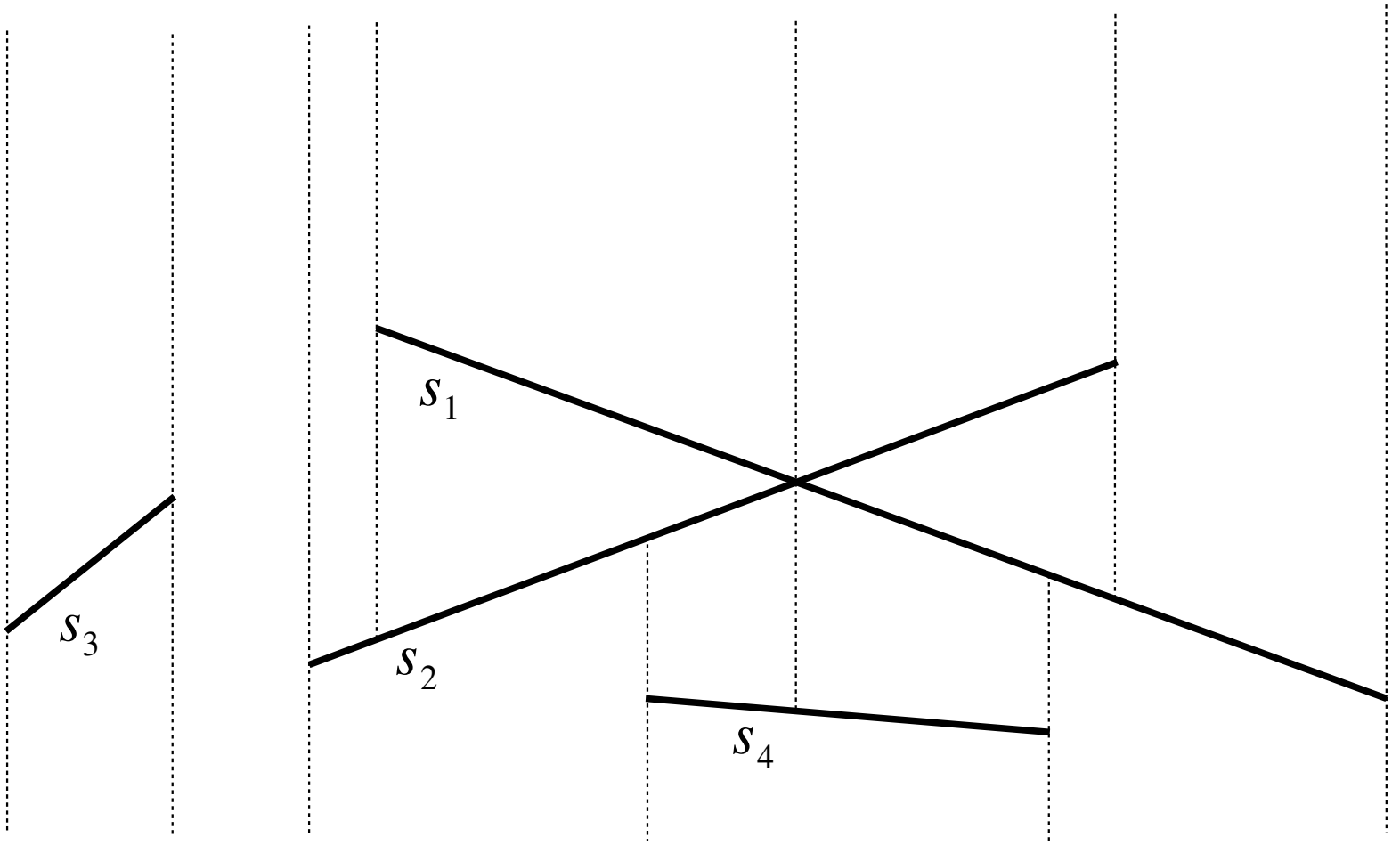
Dann wird s_2 hinzugenommen.



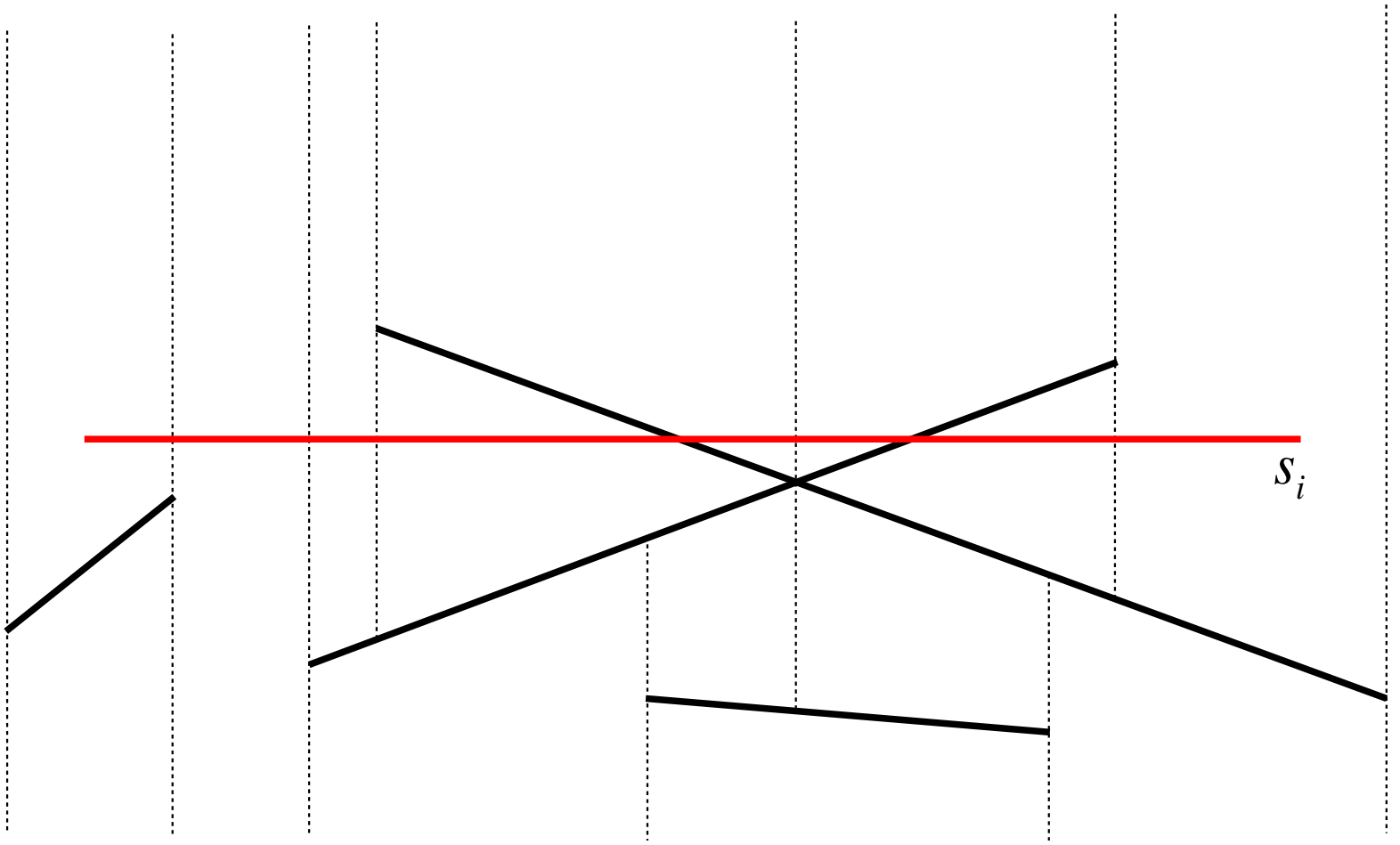
Dann wird s_3 hinzugenommen.



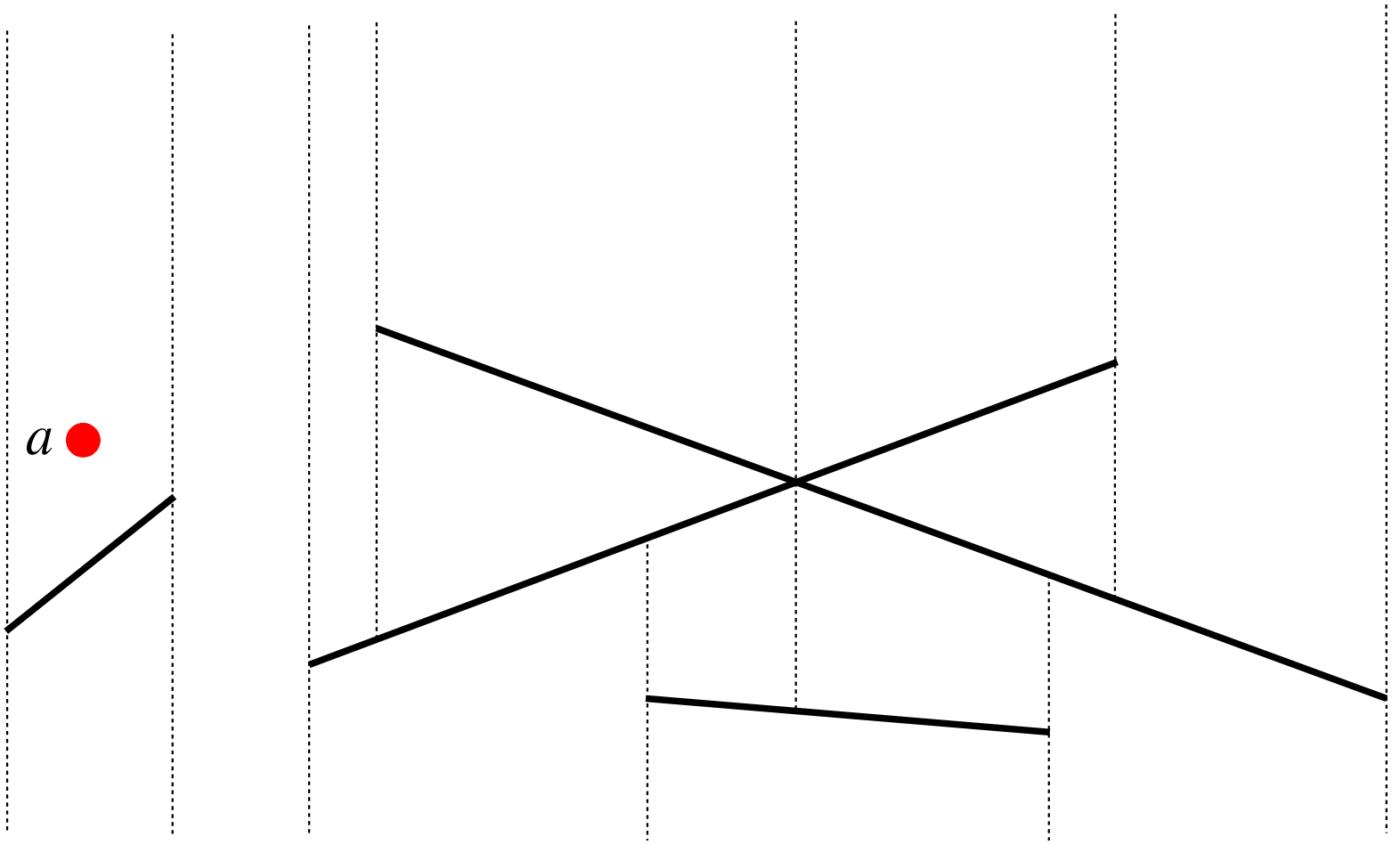
Dann wird s_4 hinzugenommen.



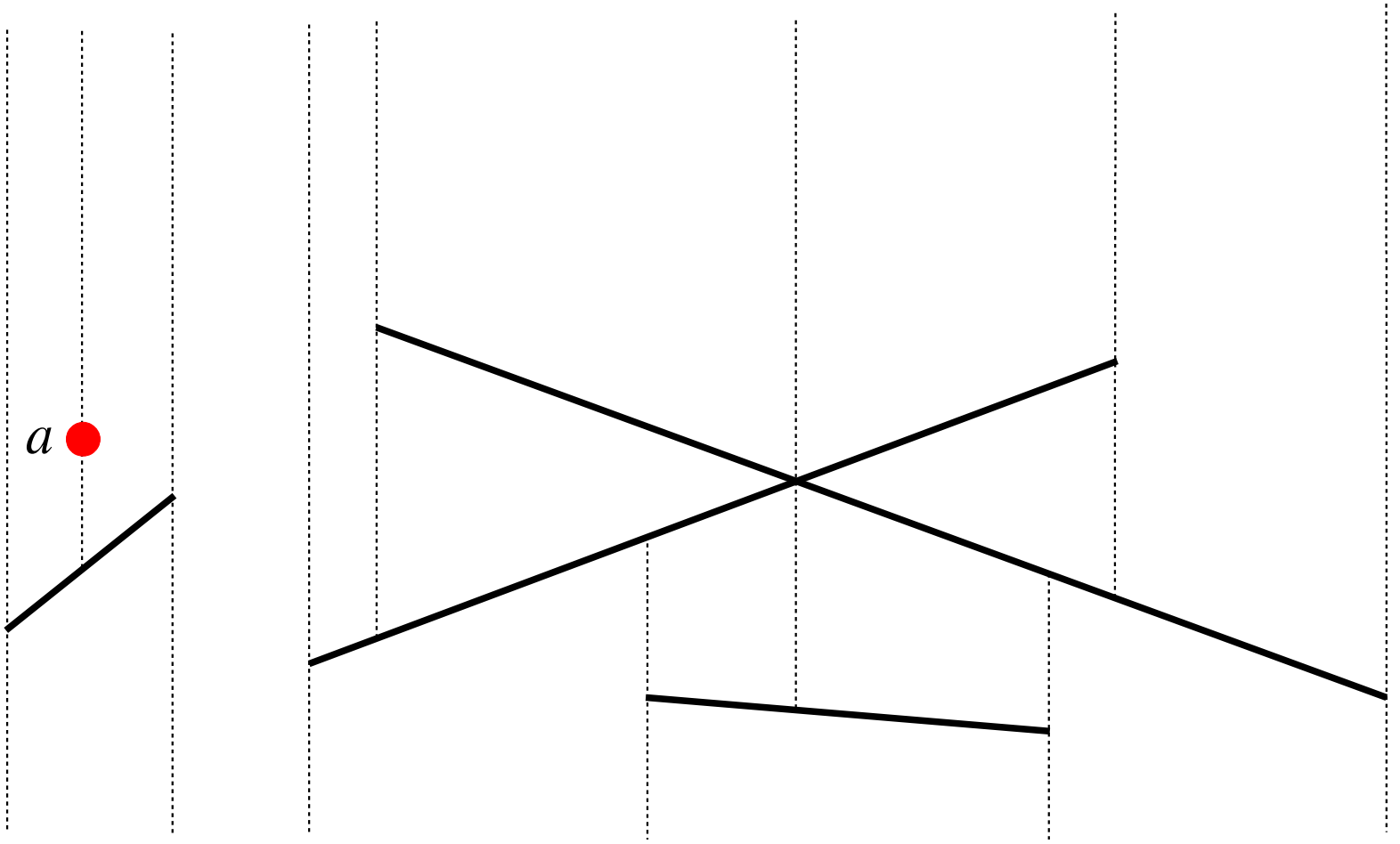
Allgemein: Vorgehensweise bei der Hinzunahme von s_i



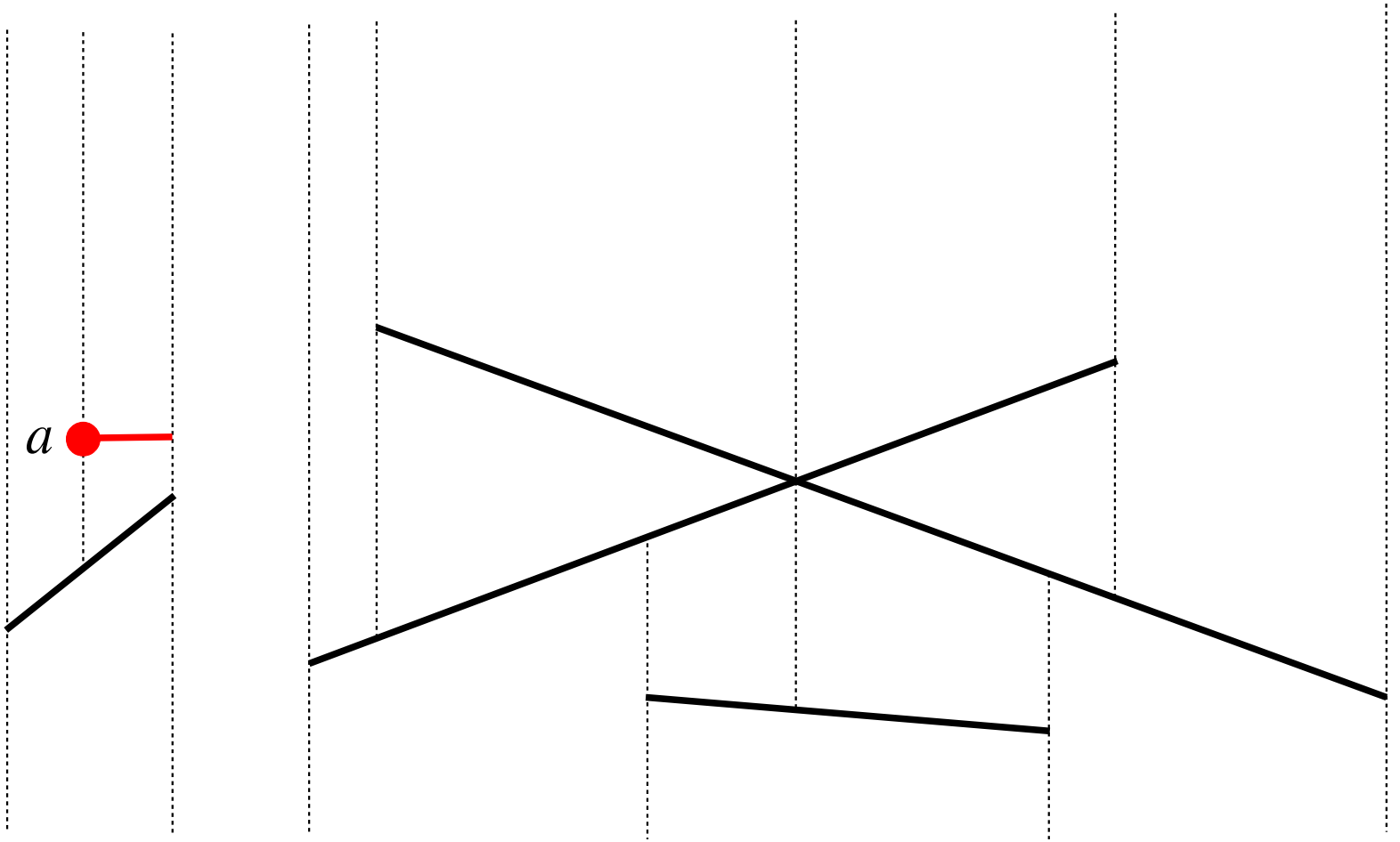
Wir gehen davon aus, dass wir wissen, in welchem Trapez der linke Endpunkt a von s_i liegt.



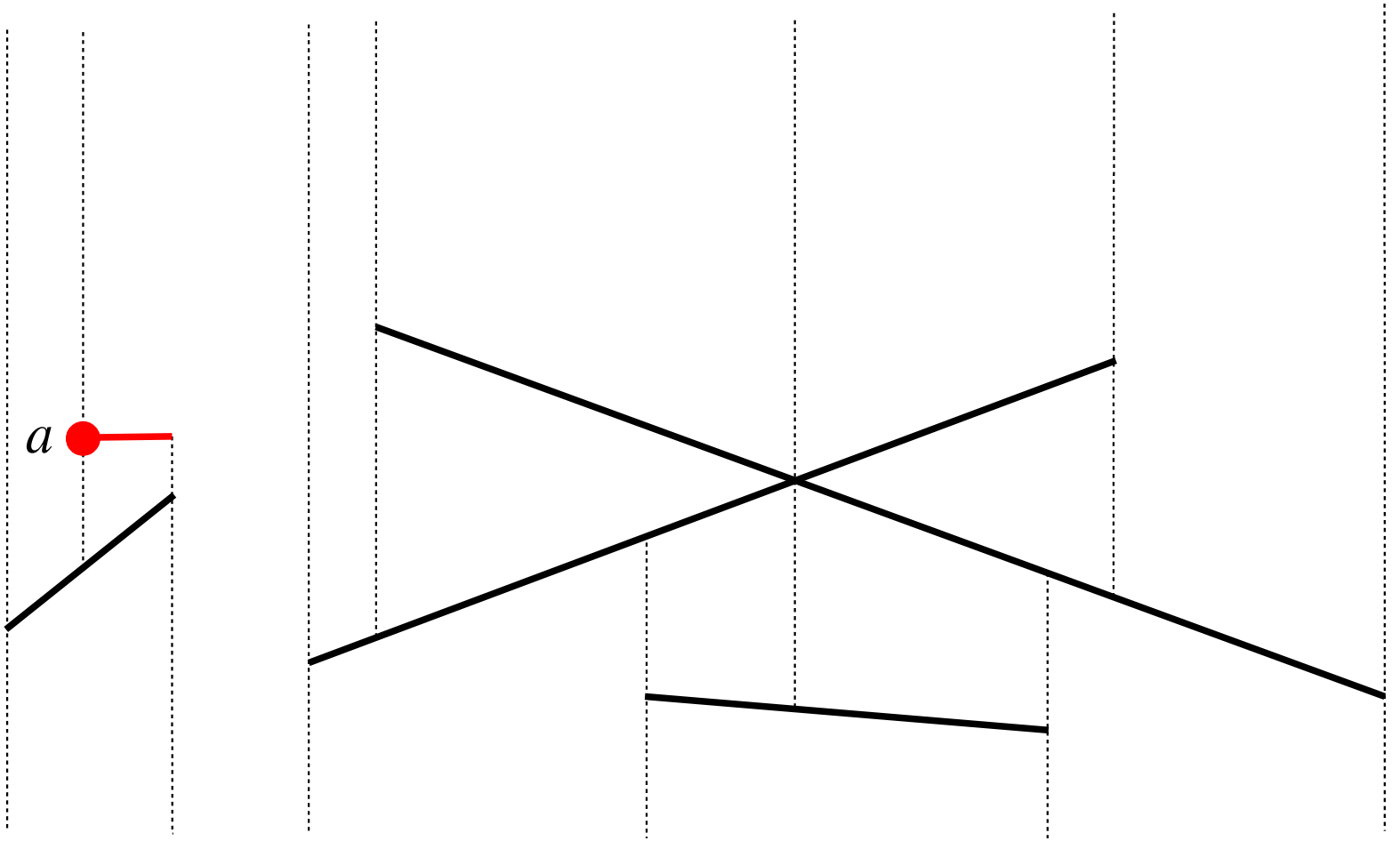
Wir schicken von a nach oben und unten je einen Strahl.
Diese stoppen, wenn sie eine Strecke treffen.



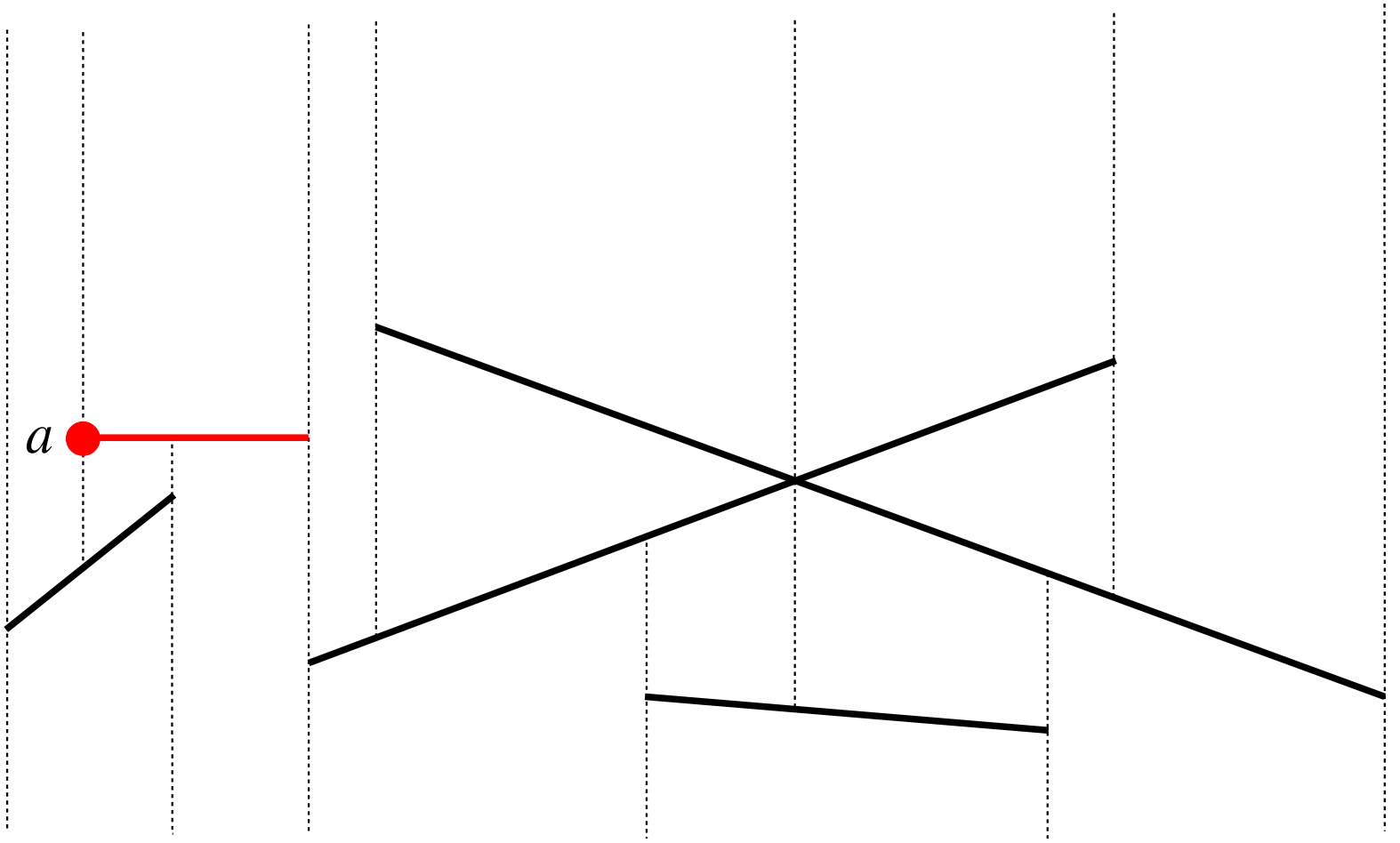
Wir folgen nun von a aus der Strecke s_i bis wir den Rand eines Trapezes erreichen.



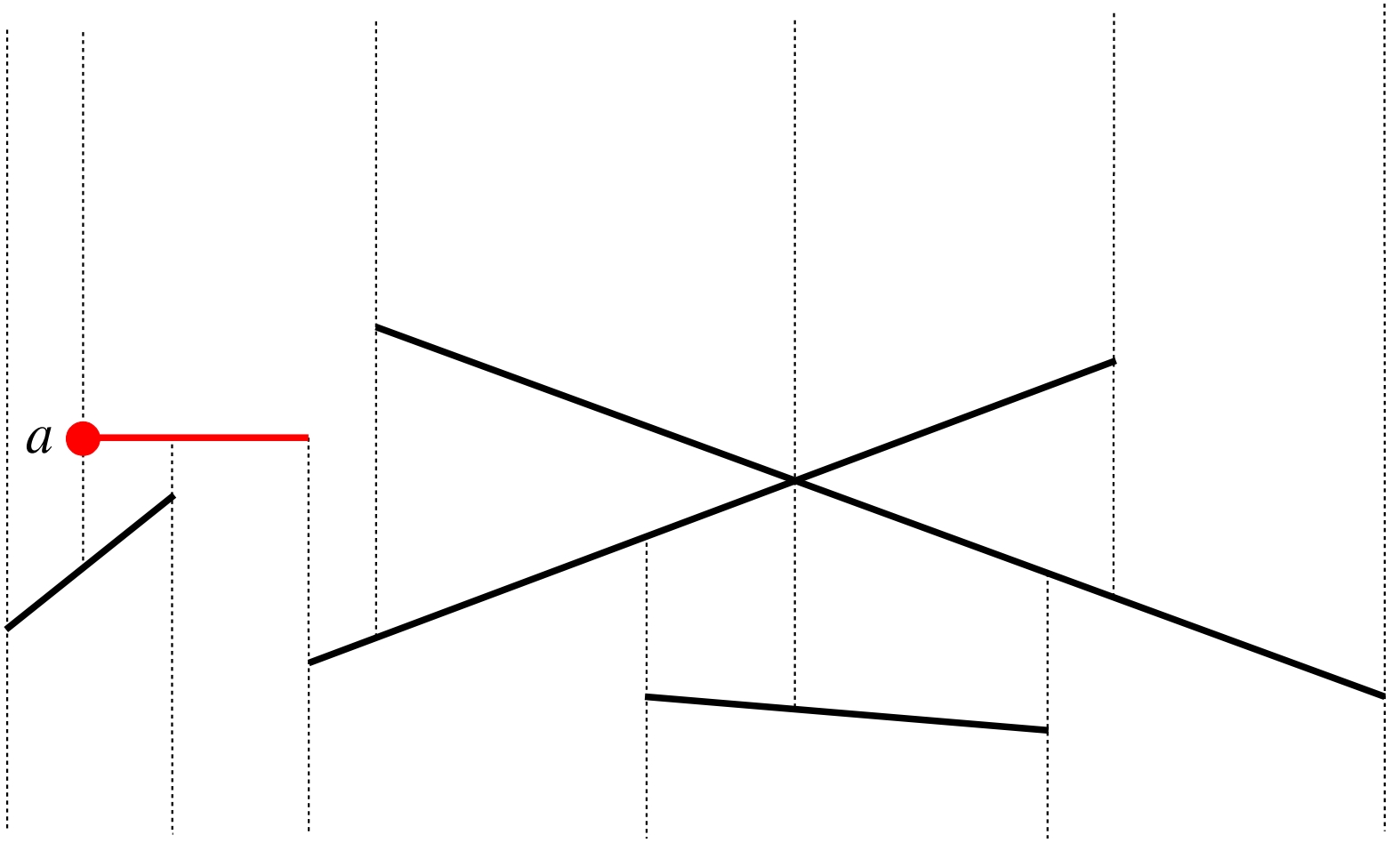
Wenn wir auf einen senkrechten Strahl treffen, wird dieser entsprechend gekürzt. Anschließend geht es weiter.



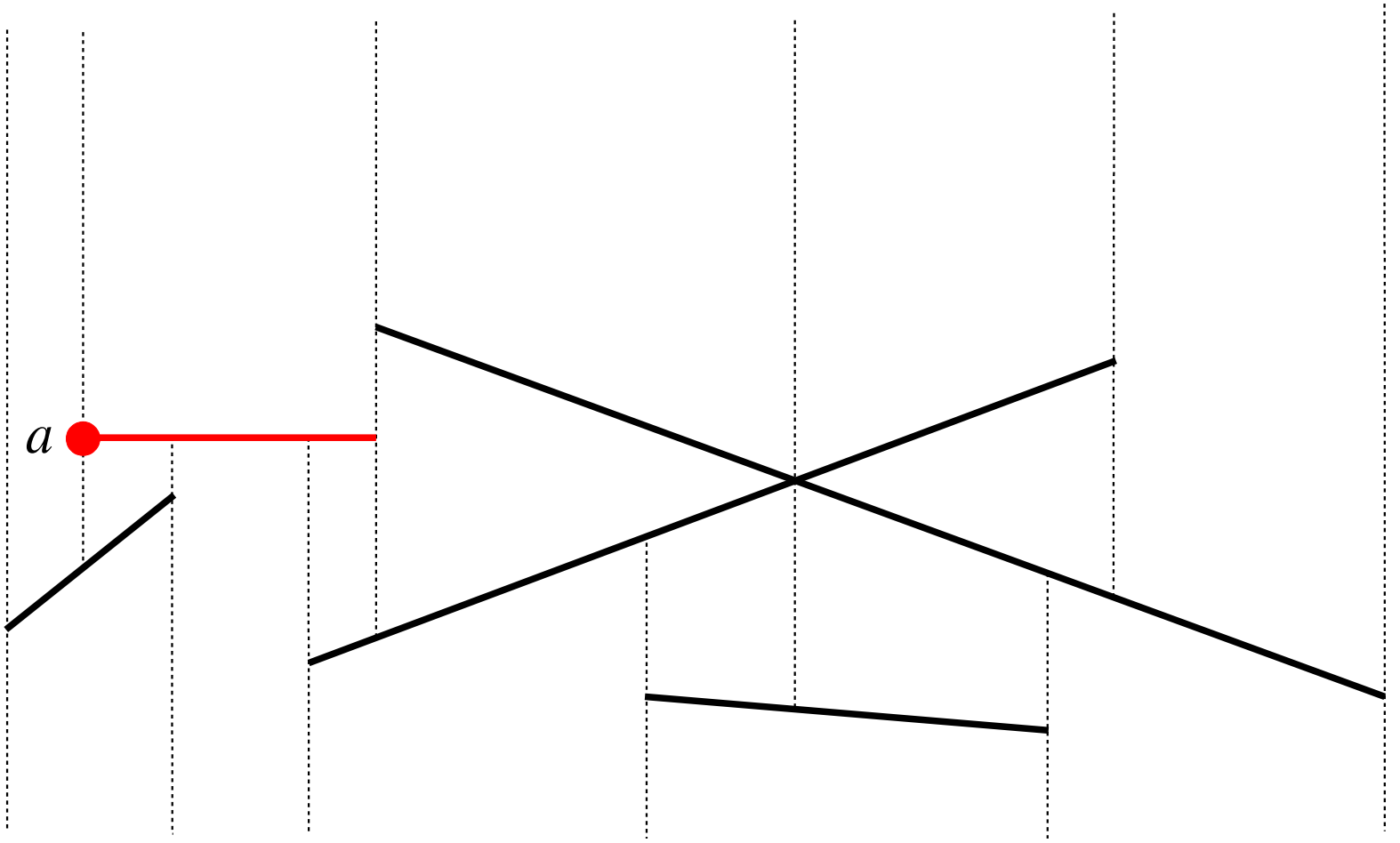
Wenn wir auf einen senkrechten Strahl treffen, wird dieser entsprechend gekürzt. Anschließend geht es weiter.



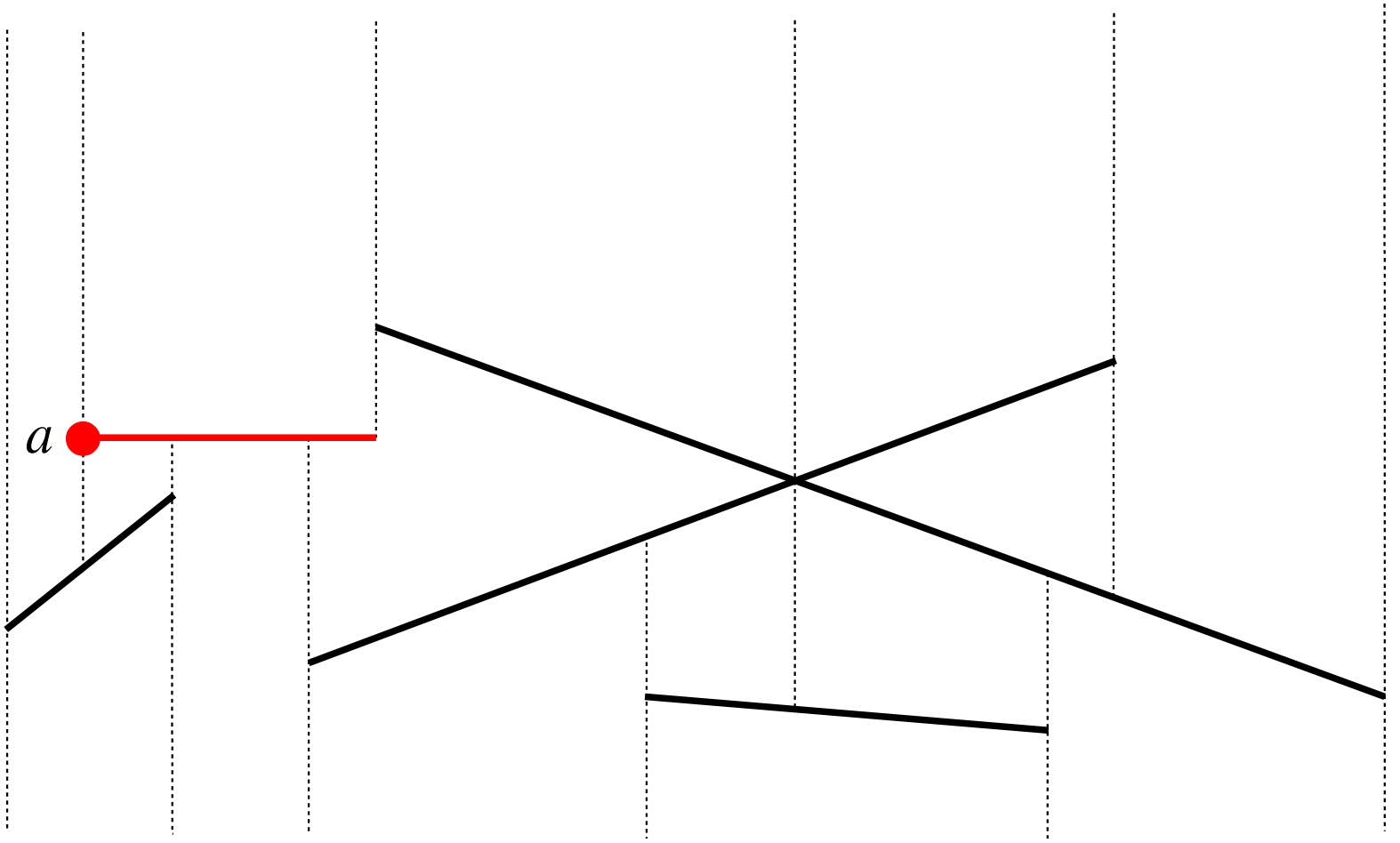
Wenn wir auf einen senkrechten Strahl treffen, wird dieser entsprechend gekürzt. Anschließend geht es weiter.



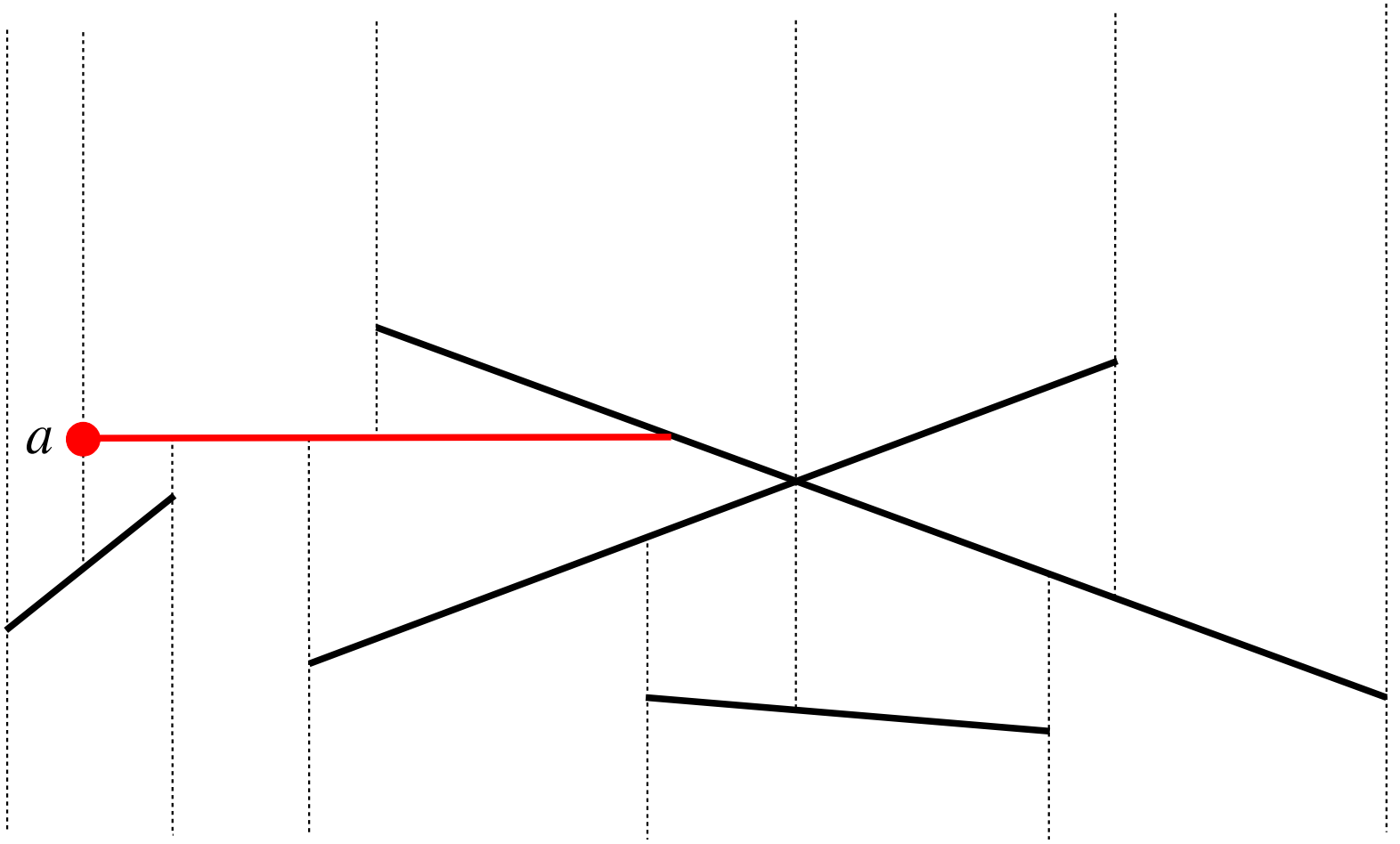
Wenn wir auf einen senkrechten Strahl treffen, wird dieser entsprechend gekürzt. Anschließend geht es weiter.



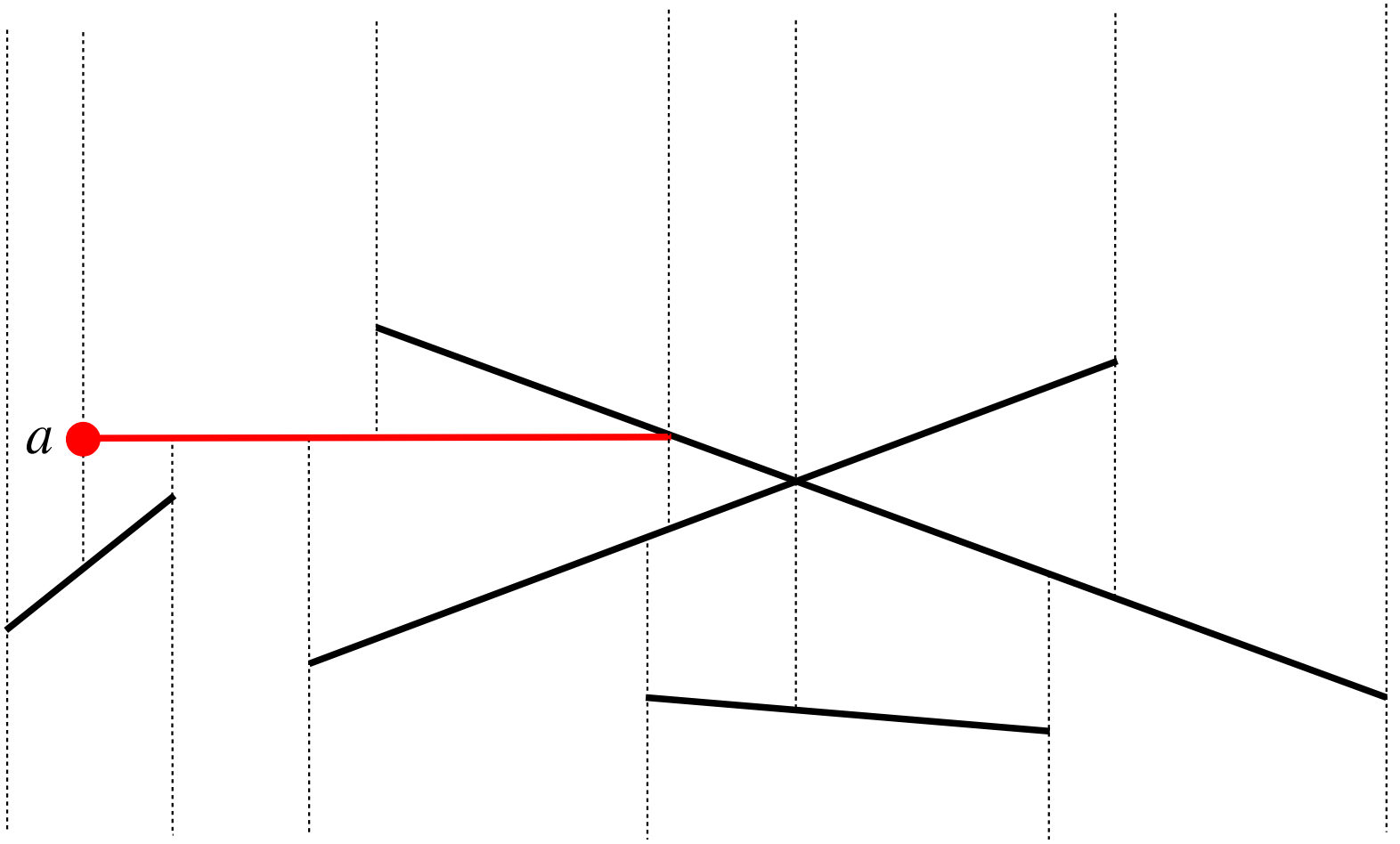
Wenn wir auf einen senkrechten Strahl treffen, wird dieser entsprechend gekürzt. Anschließend geht es weiter.



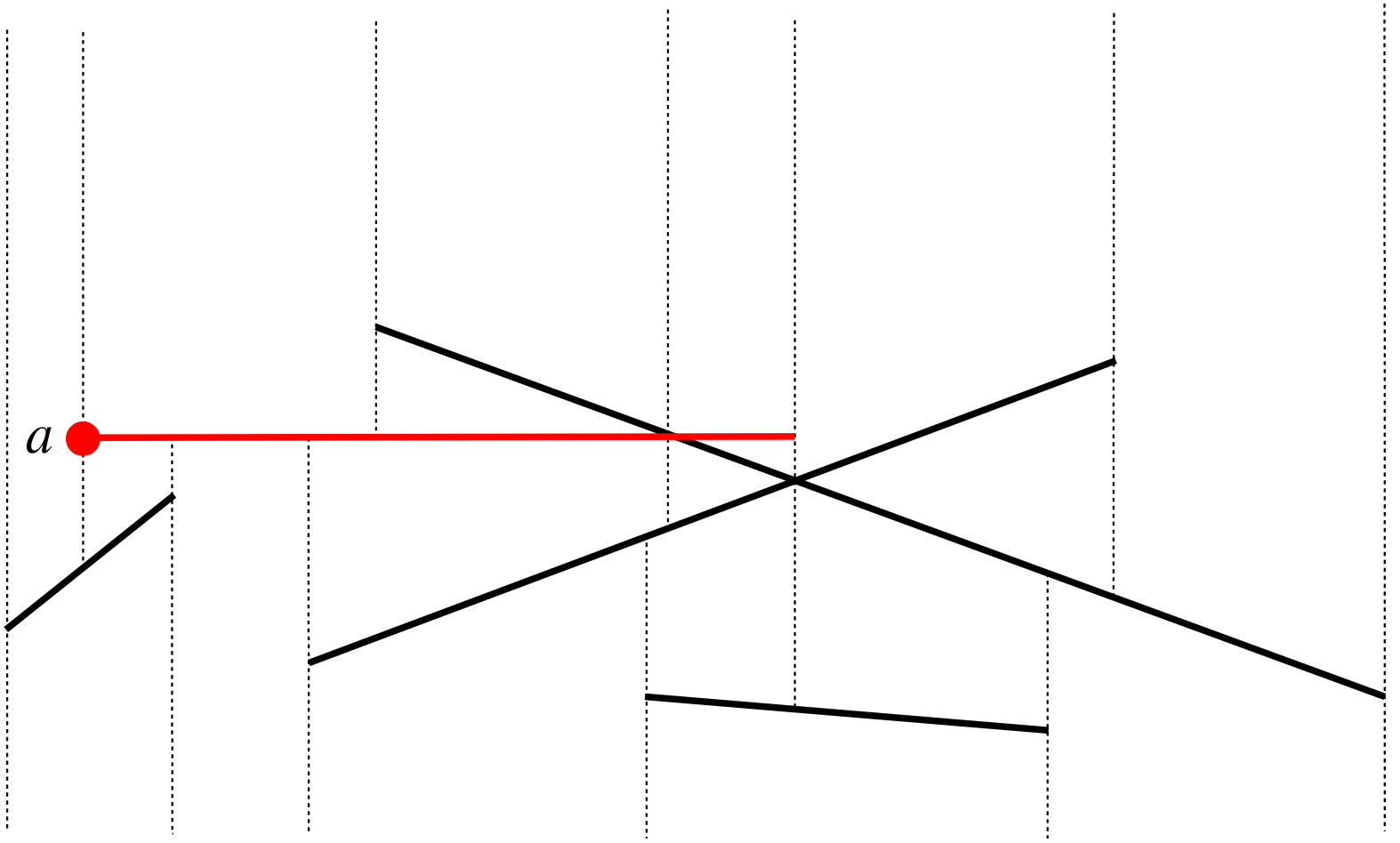
Wenn wir auf eine andere Strecke treffen, schicken wir vom Schnittpunkt nach oben und unten Strahlen.



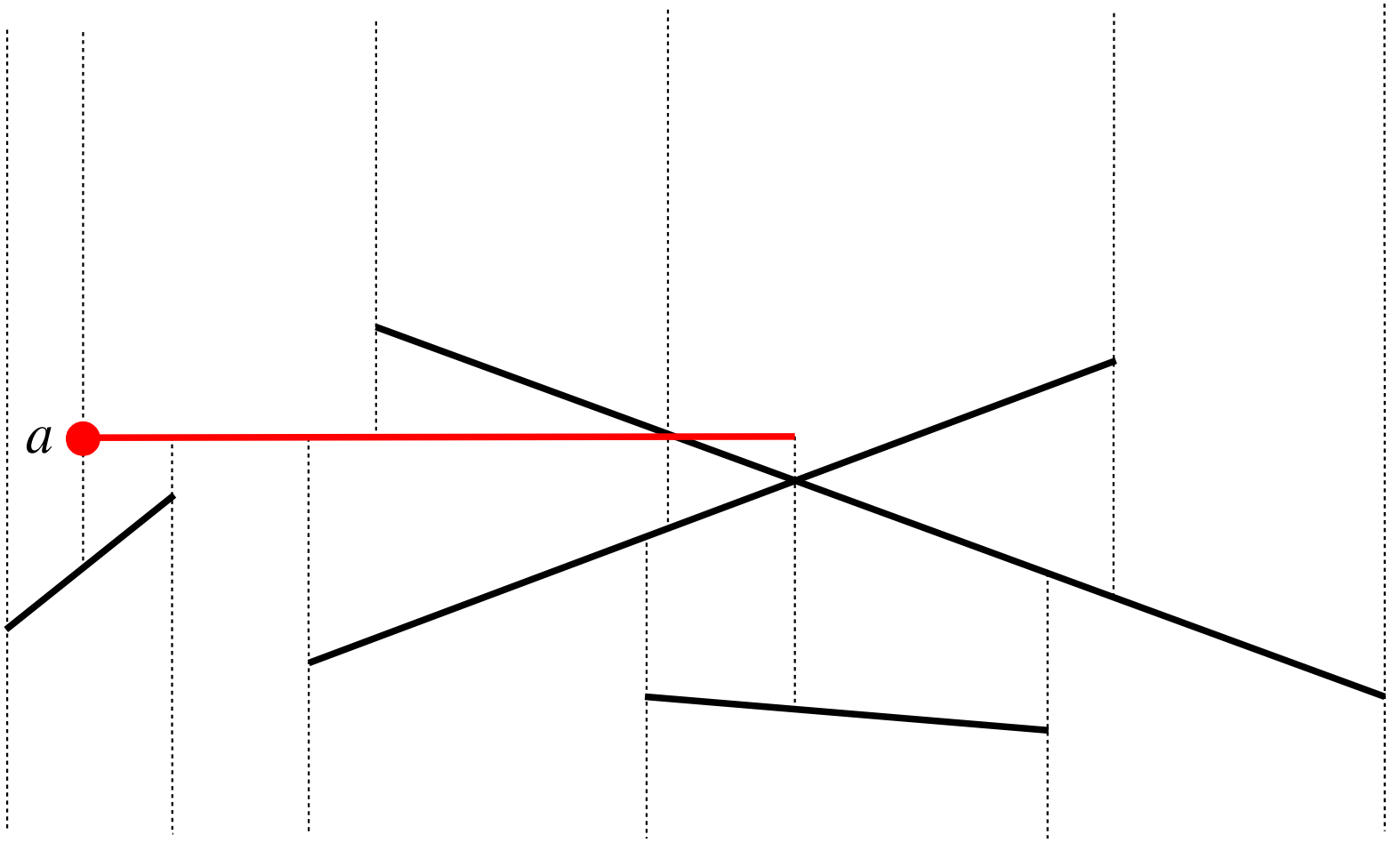
Wenn wir auf eine andere Strecke treffen, schicken wir vom Schnittpunkt nach oben und unten Strahlen.



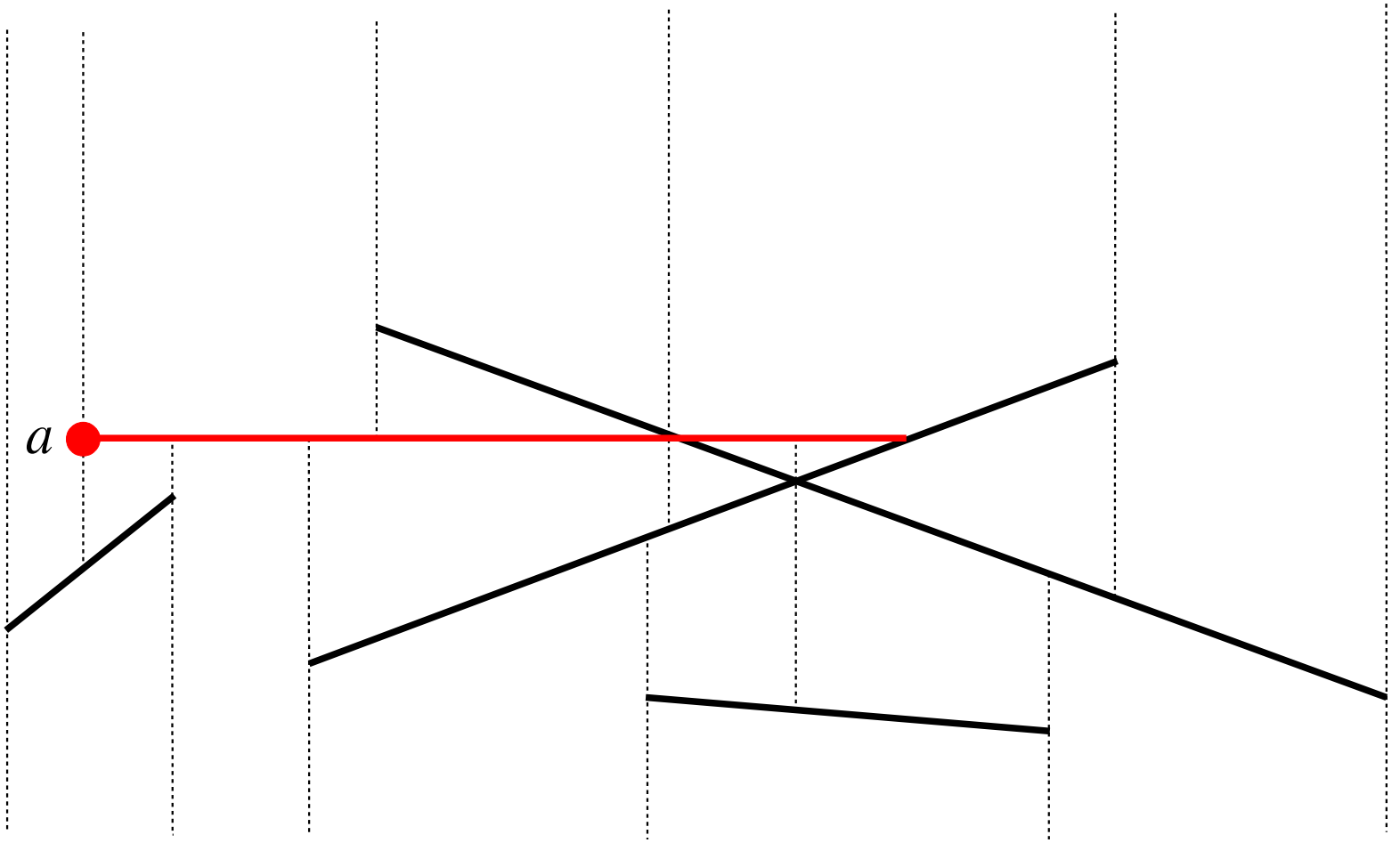
Wenn wir auf einen senkrechten Strahl treffen, wird dieser entsprechend gekürzt. Anschließend geht es weiter.



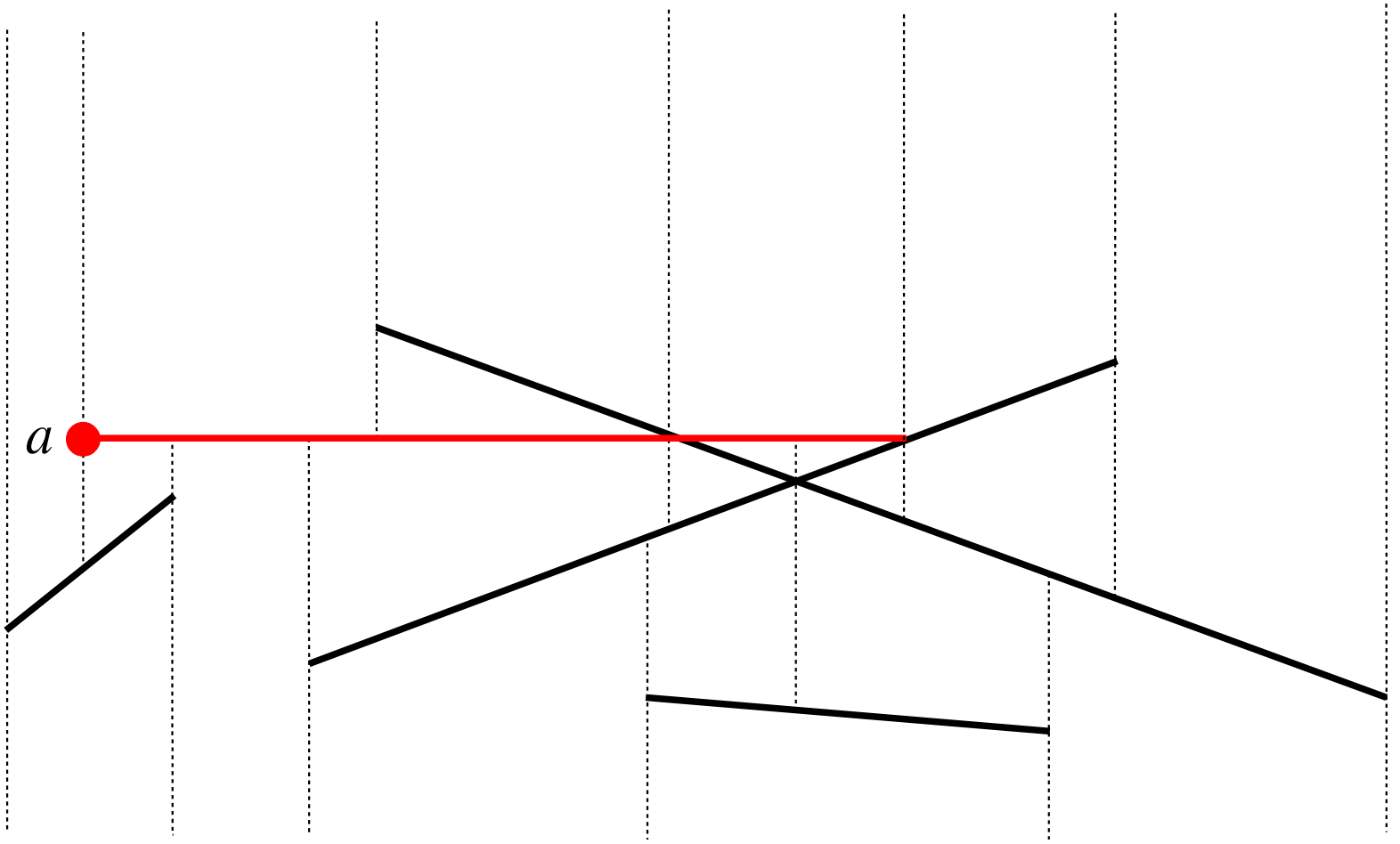
Wenn wir auf einen senkrechten Strahl treffen, wird dieser entsprechend gekürzt. Anschließend geht es weiter.



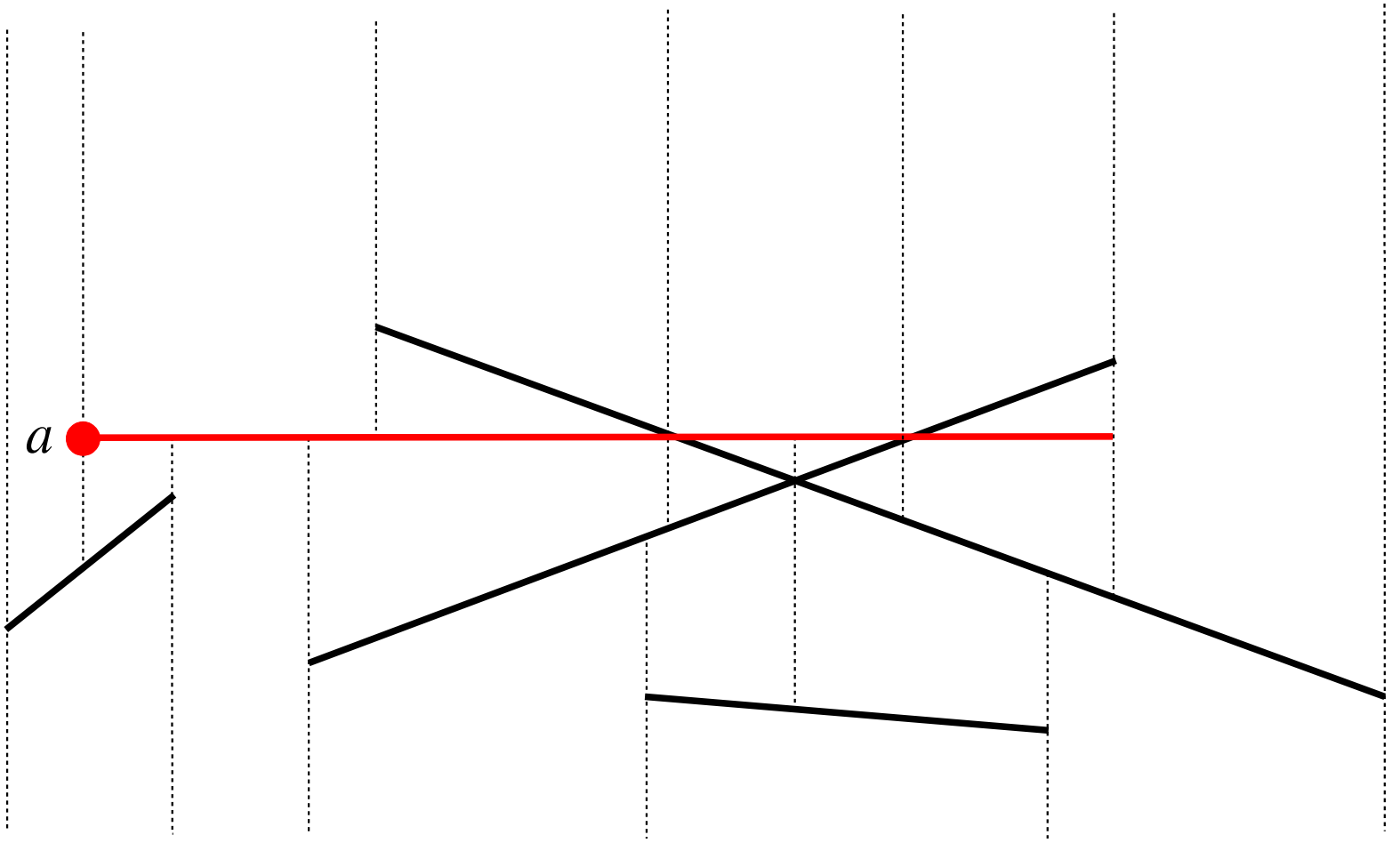
Wenn wir auf eine andere Strecke treffen, schicken wir vom Schnittpunkt nach oben und unten Strahlen.



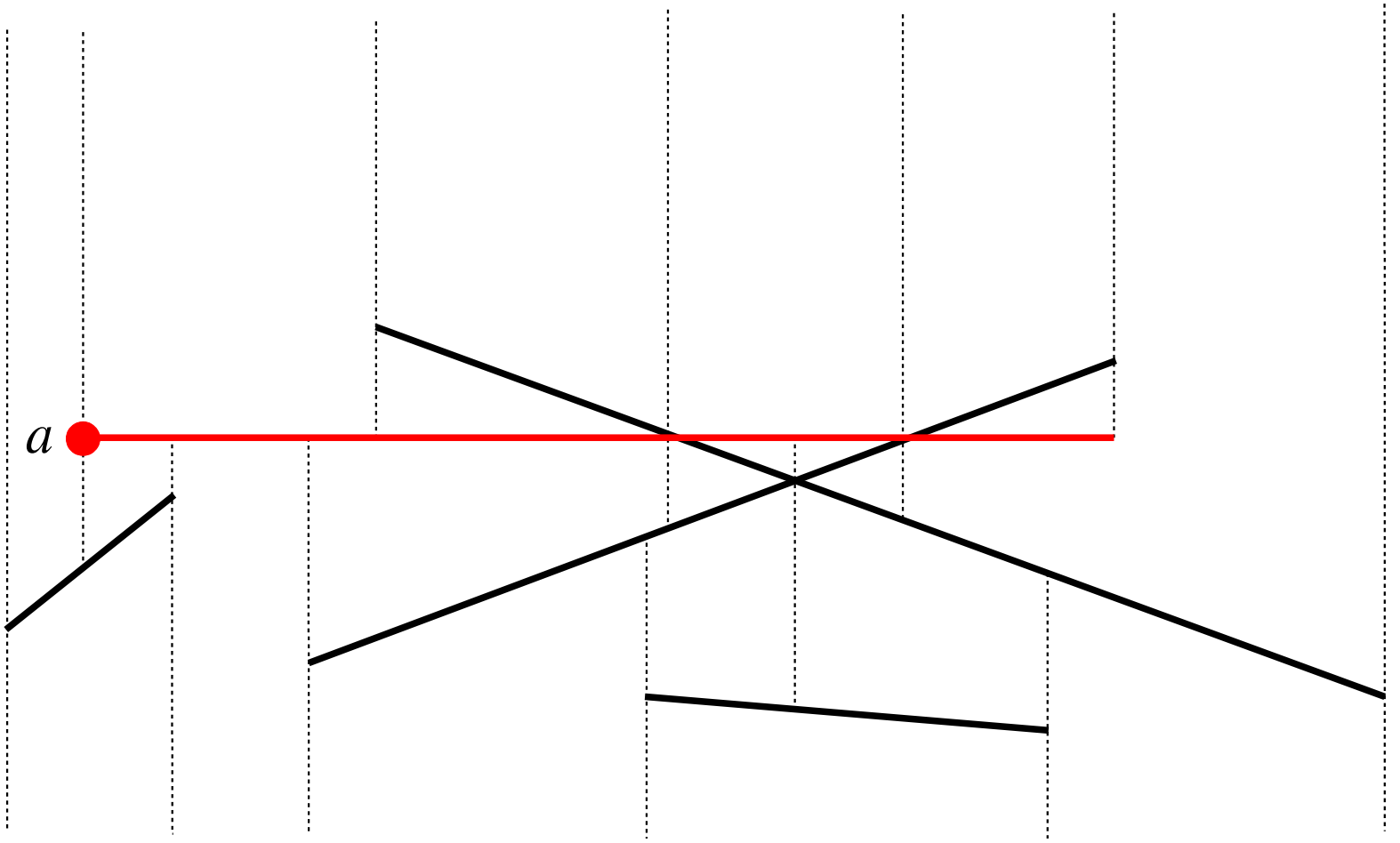
Wenn wir auf eine andere Strecke treffen, schicken wir vom Schnittpunkt nach oben und unten Strahlen.



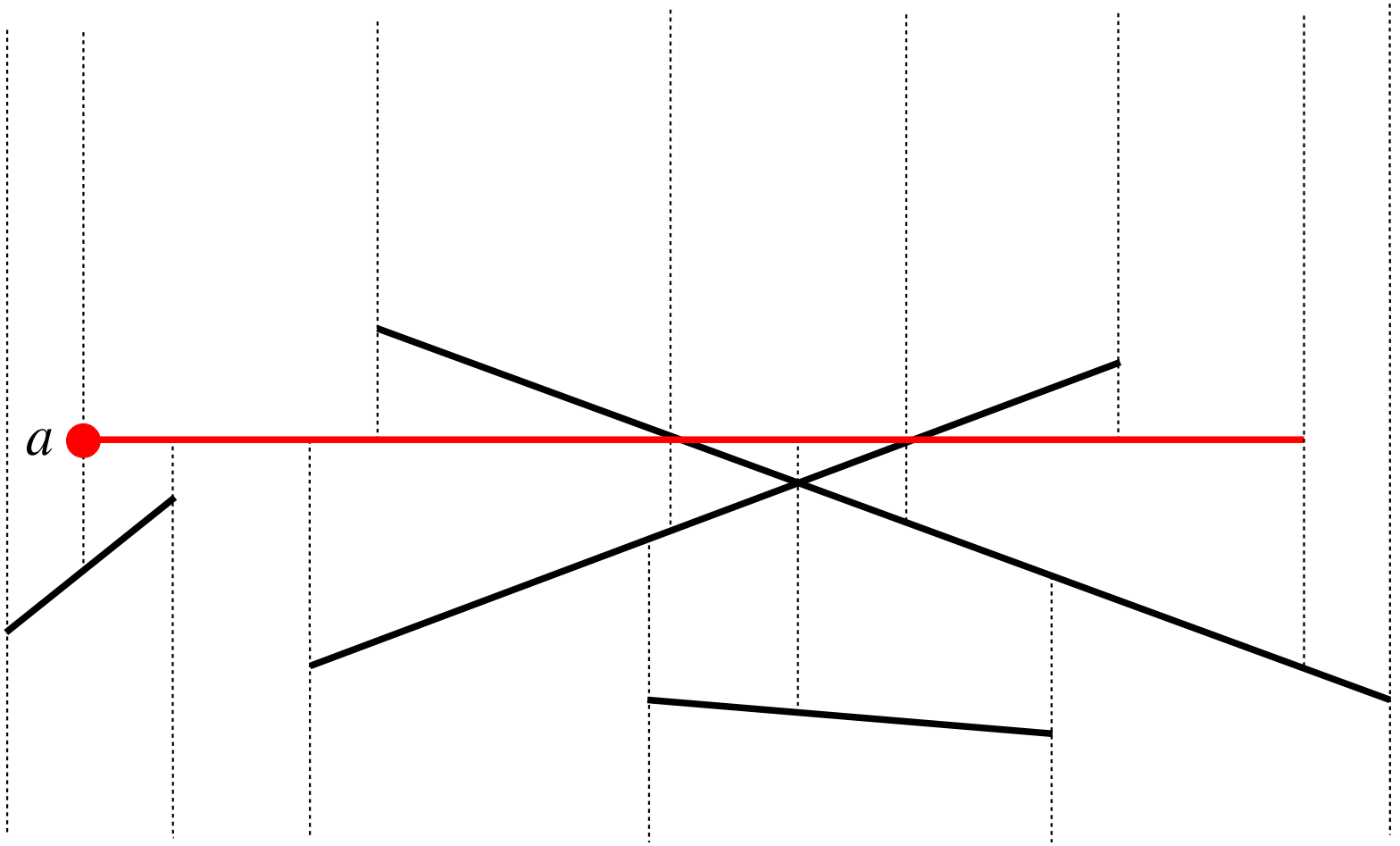
Wenn wir auf einen senkrechten Strahl treffen, wird dieser entsprechend gekürzt. Anschließend geht es weiter.



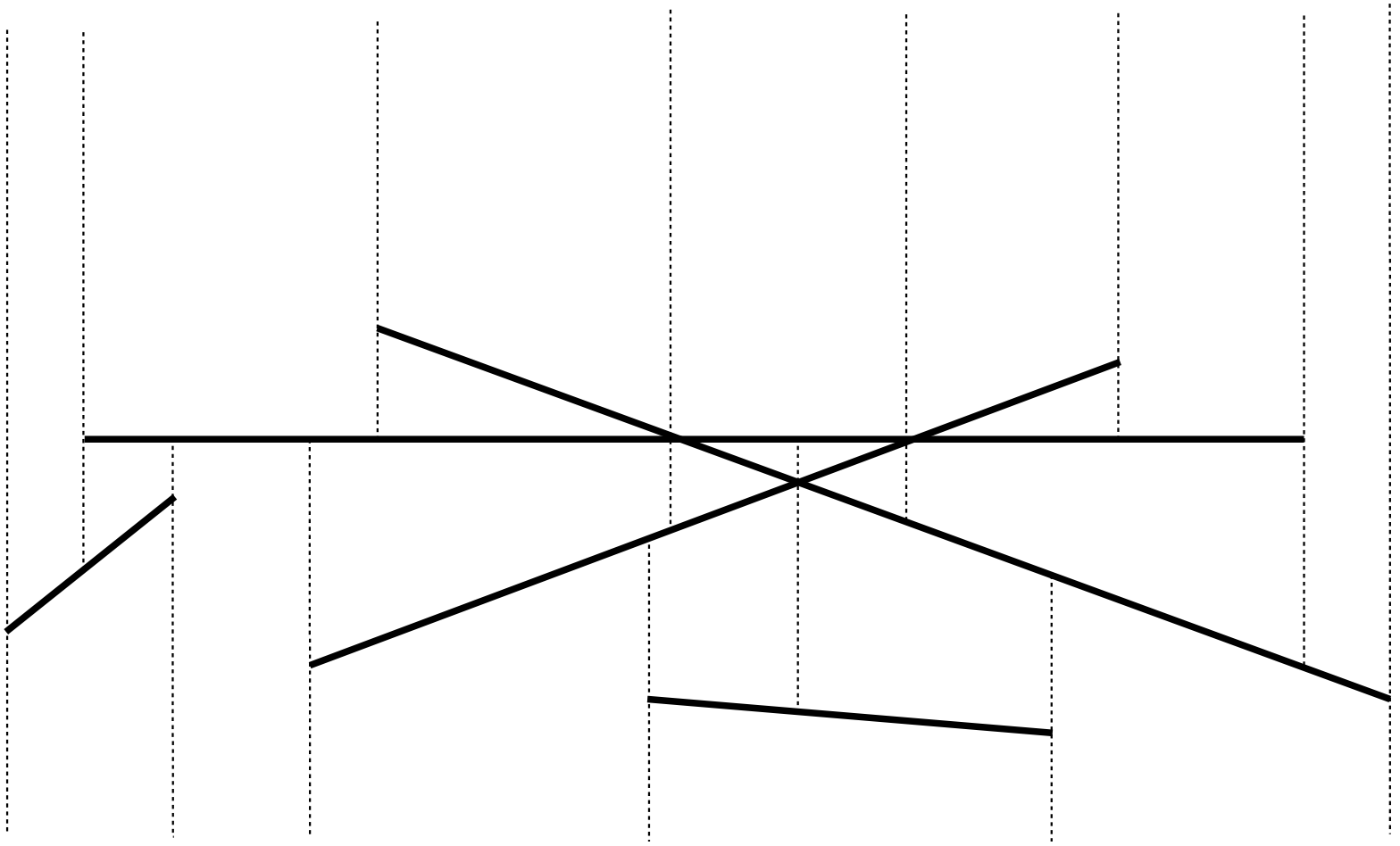
Wenn wir auf einen senkrechten Strahl treffen, wird dieser entsprechend gekürzt. Anschließend geht es weiter.



Wenn wir den rechten Endpunkt von s_i erreichen, schicken wir nach oben und unten einen Strahl. Damit sind wir fertig.



Wenn wir den rechten Endpunkt von s_i erreichen, schicken wir nach oben und unten einen Strahl. Damit sind wir fertig.



Rechenzeit bei zufälliger Reihenfolge der Strecken

Wir brauchen zwei Dinge:

1. Datenstruktur zur Lokalisierung des Anfangspunkts einer Strecke (siehe späteres Kapitel)

2. Datenstruktur zur Verfolgung einer Strecke durch die Trapezzerlegung (Listen mit Querverweisen)

Erwartete Laufzeit: $O(n \log n + k)$