Intersección de segmentos de línea

Dr. Eduardo A. Rodríguez Tello

Cinvestav-Tamaulipas

22 de febrero del 2013

1. Intersección de segmentos de línea
   - Intersecciones geométricas
   - Intersección de segmentos de línea
   - Cálculo del punto de intersección

2. Algoritmo de barrido del plano (plane sweep)
   - Introducción
   - Línea de barrido
   - Eventos
   - Actualización de eventos
   - Detección de intersecciones
   - Estructuras de datos
   - Algoritmo de barrido del plano
   - Ejemplo
   - Análisis de complejidad

Cinvestav

- El material de la clase de hoy está basado en el capítulo 2 del siguiente libro:
  Mark de Berg, Otfried Cheong, Marc van Kreveld and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition (April 16, 2008), ISBN-10: 3540779736.

Cinvestav

Intersecciones geométricas

- Uno de los problemas más básicos en geometría computacional es el del cálculo de intersecciones
- El cálculo de intersecciones en espacios 2 y 3 dimensionales es esencial para diversas áreas de aplicación
- En **modelado de sólidos** las personas suelen crear formas complejas mediante la aplicación de diversas operaciones booleanas (intersección, unión, y diferencia) a simples formas primitivas. El proceso es llamado *geometría constructiva de sólidos* (CSG). Con el fin de realizar estas operaciones, el paso más básico es determinar los puntos en los que los límites de dos objetos se intersectan

Cinvestav

Intersecciones geométricas

- En **robótica** y **planificación del movimiento** es importante saber cuando dos objetos se intersectan para detectar y evitar colisiones
- En los **sistemas de información geográfica** es útil la superposición de dos subdivisiones (e.g., una red de carreteras y los límites de un estado para determinar las responsabilidades de mantenimiento de carreteras). Dado que estas redes se forman a partir de colecciones de segmentos de línea, esto genera el problema de la determinación de las intersecciones de segmentos de línea

Cinvestav

Intersecciones geométricas

- En **gráficas computacionales**, *ray shooting* es un importante método para renderizar escenas. Computacionalmente la parte más intensiva de este método es determinar la intersección de los rayos con otros objetos.
- La mayoría de los problemas de intersección complejos se descomponen sucesivamente en subproblemas más simples.
- El día de hoy vamos a discutir el algoritmo más básico, a partir del cual están compuestos otros algoritmos más complejos para resolver el problema del cálculo de intersecciones

Cinvestav

Cinvestav

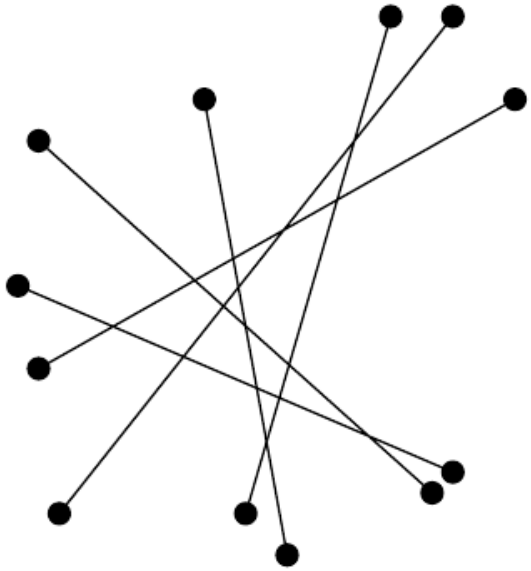
Intersección de segmentos de línea

- El problema que vamos a considerar es, dados n segmentos de línea en el plano, reportar todos los puntos donde un par de segmentos de línea se intersectan
- Asumimos que cada segmento de línea está representado con las coordenadas de sus dos puntos extremos
- Observemos que n segmentos de línea pueden tener un número total de puntos de intersección que va desde 0 hasta $\binom{n}{2} = O(n^2)$

Cinvestav

Intersección de segmentos de línea

- Intersección de segmentos de línea



Cinvestav

Intersección de segmentos de línea

- Es posible diseñar un algoritmo $O(n^2)$, argumentando que es asintóticamente óptimo en el peor caso
- Pero no sería útil en la práctica
- Esto es debido a que en muchas instancias del problema el número de intersecciones puede ser relativamente pequeño (mucho menor que n^2)
- Por lo tanto parece razonable buscar un algoritmo sensible a la salida

Cinvestav

Intersección de segmentos de línea

- Sea I el número de intersecciones
- Asumiremos que los segmentos de línea están en posición general, de esta forma no nos preocuparemos por el problema de si hay tres o más líneas intersectándose en un solo punto
- Sin embargo, generalizar el algoritmo para manejar este tipo de casos especiales es un ejercicio interesante (ver capítulo 2 del libro de Berg et al.)

Cinvestav

Intersección de segmentos de línea

- El mejor tiempo de ejecución en el peor caso que debemos esperar es $O(n \log n + I)$
- Claramente requerimos un tiempo $O(I)$ para reportar los puntos de intersección
- Lo que es menos obvio es que se necesita un tiempo $O(n \log n)$ para el resto del algoritmo
- Este resultado se deriva del hecho de que el *problema de unicidad de elementos* requiere un tiempo $\Omega(n \log n)$ para resolverse usando el modelo de árboles algebraicos de decisión
- Dada una lista de n números reales, el problema de unicidad de elementos consiste en determinar si todos esos números son distintos

Cinvestav

Intersección de segmentos de línea

- Vamos a reducir el problema de unicidad de elementos al de reporte de la intersección en tiempo $O(n)$
- Esto significa que si podemos resolver el reporte de la intersección en tiempo $o(n \log n)$, entonces se podría resolver el problema de unicidad de elementos en tiempo $o(n \log n)$, lo que estaría en contradicción con la cota inferior conocida $\Omega(n \log n)$
- Aquí está la reducción. Dada una lista de n números, $(x_1, x_2, \ldots, x_n)$, en tiempo $O(n)$ podemos construir un conjunto de n segmentos de línea horizontales, todos teniendo la misma coordenada y

Cinvestav

Intersección de segmentos de línea

- Por ejemplo, mapear x_i a un segmento de línea horizontal $\overline{p_i q_i}$ donde $p_i = (0, x_i)$ y $q_i = (1, x_i)$
- Observemos que si los números son distintos, entonces no hay intersecciones entre los segmentos
- Por otra parte, si dos o más números son iguales, entonces los segmentos correspondientes serán iguales y, por lo tanto existe al menos una intersección

Cinvestav

Intersección de segmentos de línea

- Tengamos en cuenta que esta cota inferior asume el uso del modelo de árboles algebraicos de decisión, en el cual todas las decisiones están hechas por comparaciones basadas en operaciones algebraicas exactas $(+, -, *, /)$ aplicadas a entradas numéricas
- A pesar de que esto abarca la mayor parte de lo que se consideran como cálculos geométricos "normales", hay otros modelos alternativos de cálculo
- Por ejemplo, utilizando operaciones de *módulo*, *floor*, o *ceil*, es posible implementar el *hashing*

Cinvestav

Intersección de segmentos de línea

- Usando *hashing* es posible resolver el problema de unicidad de elementos en tiempo $O(n)$
- Lamentablemente, incluso en este modelo computacional más poderoso, nadie conoce cómo determinar si dos segmentos de línea cualquiera se intersectan en un tiempo más rápido que $O(n \log n + I)$ en el peor caso

Cinvestav

Intersección de segmentos de línea

- Más adelante en el curso discutiremos un algoritmo óptimo aleatorizado que resuelve el problema de intersección de segmentos de línea en tiempo $O(n \log n + I)$
- Por ahora sólo presentaremos un algoritmo (no óptimo) para resolver este problema que corre en tiempo $O(n \log n + I \log n)$
- Pero antes veamos cómo calcular el punto en el cuál dos segmentos de línea se intersectan

Cinvestav

1. Intersección de segmentos de línea
   - Intersecciones geométricas
   - Intersección de segmentos de línea
   - Cálculo del punto de intersección

Cinvestav

Cálculo del punto de intersección

- Como muchas primitivas geométricas, calcular el punto en el cuál dos segmentos de línea se intersectan pude ser reducido a resolver un sistema lineal de ecuaciones
- Sean $\overline{ab}$ y $\overline{cd}$ dos segmentos de línea, definidos por sus puntos extremos
- Notemos primero que es posible determinar dónde estos segmentos de línea se intersectan, simplemente aplicando una combinación apropiada de pruebas de orientación (Ejercicio)

Cinvestav

Cálculo del punto de intersección

- Otra forma de determinar el punto en el cuál los segmentos se intersectan es usando una técnica empleada en gráficas computacionales la cual consiste en representar los segmentos como ecuaciones paramétricas
- Recordemos que cualquier punto en un segmento de línea $\overline{ab}$ puede ser escrito como una combinación convexa empleando un parámetro real s:

$$p(s) = (1 - s)a + sb \qquad \text{para } 0 \leq s \leq 1 \tag{1}$$

- De forma similar para $\overline{cd}$ podemos introducir un parámetro t:

$$q(t) = (1 - t)c + td \qquad \text{para } 0 \leq t \leq 1 \tag{2}$$

Cálculo del punto de intersección

- Una intersección occurre si y sólo si podemos encontrar valores para s y t en los rangos deseados tales que $p(s) = q(t)$
- Por lo tanto obtenemos las dos ecuaciones siguientes:

$$\begin{aligned} (1-s)a_x + sb_x &= (1-t)c_x + td_x \\ (1-s)a_y + sb_y &= (1-t)c_y + td_y \end{aligned} \tag{3}$$

- Las coordenadas de los puntos son conocidas, por lo tanto es un simple ejercicio de álgebra lineal resolver este sistema para s y t

Cinvestav

Cálculo del punto de intersección

- El cálculo de s y t implica una división
- Si el divisor es cero, significa que los segmentos de línea son paralelos o colineales
- Estos casos especiales deben ser tratados con cuidado
- Si el divisor es distinto de cero, entonces obtendremos valores para s y t como número racionales

Cinvestav

Cálculo del punto de intersección

- Podemos aproximarlos a números de punto flotante, o si deseamos realizar cálculos exactos es posible simular álgebra de números racionales usando enteros de alta precision
- Una vez que los valores de s y t han sido calculados todo lo que se requiere es comprobar que ambos están en el $[0, 1]$

Cinvestav

2 Algoritmo de barrido del plano (plane sweep)

- Introducción
- Línea de barrido
- Eventos
- Actualización de eventos
- Detección de intersecciones
- Estructuras de datos
- Algoritmo de barrido del plano
- Ejemplo
- Análisis de complejidad

Cinvestav

Introducción

- Dado un conjunto de segmentos de línea $S = \{s_1, s_2, \ldots, s_n\}$ cuyas intersecciones se desean conocer
- El método llamado de **barrido del plano** (*plane sweep*) está compuesto por los siguientes elementos esenciales, los cuales detallaremos a continuación:
  - Línea de barrido
  - Eventos
  - Actualización de eventos

Cinvestav

2 Algoritmo de barrido del plano (plane sweep)

- Introducción
- Línea de barrido
- Eventos
- Actualización de eventos
- Detección de intersecciones
- Estructuras de datos
- Algoritmo de barrido del plano
- Ejemplo
- Análisis de complejidad

Cinvestav

Línea de barrido

- Se simulará el barrido de una línea vertical l, llamada la **línea de barrido** de izquierda a derecha (el libro utiliza una línea horizontal, pero no hay una diferencia substancial)
- Vamos a mantener los segmentos de línea que se intersectan con la línea de barrido ordenados (digamos, de arriba a abajo).
- Tenga en cuenta que no podemos almacenar de manera eficiente las coordenadas reales en las cuales los segmentos de línea intersectan la línea de barrido, ya que esto requeriría demasiadas actualizaciones.

Cinvestav

Línea de barrido

- En vez de eso, simplemente se almacena un puntero que nos proporciona el acceso a la ecuación del segmento de línea
- Dada la coordenada x actual de la línea de barrido, podemos calcular la coordenada y del punto de intersección como se requiere

Cinvestav

Cinvestav

Eventos

- Aunque podríamos pensar en la línea de barrido en movimiento continuo, sólo tenemos que actualizar las estructuras de datos en los puntos donde se producen algunos cambios significativos en el contenido de la línea de barrido
- Estos puntos son llamados **puntos de evento**
- La interpretación de los puntos de evento varia de acuerdo con el problema al cual se aplique el método de barrido del plano

Cinvestav

Eventos

- En nuestro caso los puntos de evento corresponden a:
  - **Eventos de extremos:** donde la línea de barrido se encuentra con el punto extremo de un segmento de línea
  - **Eventos de intersección:** donde la línea barrido se encuentra con el punto de intersección de dos segmentos de línea
- Tenga en cuenta que los eventos de extremos se pueden preordenar antes de que el barrido se ejecute
- En cambio, los eventos de intersección se descubrirán conforme el barrido se ejecuta

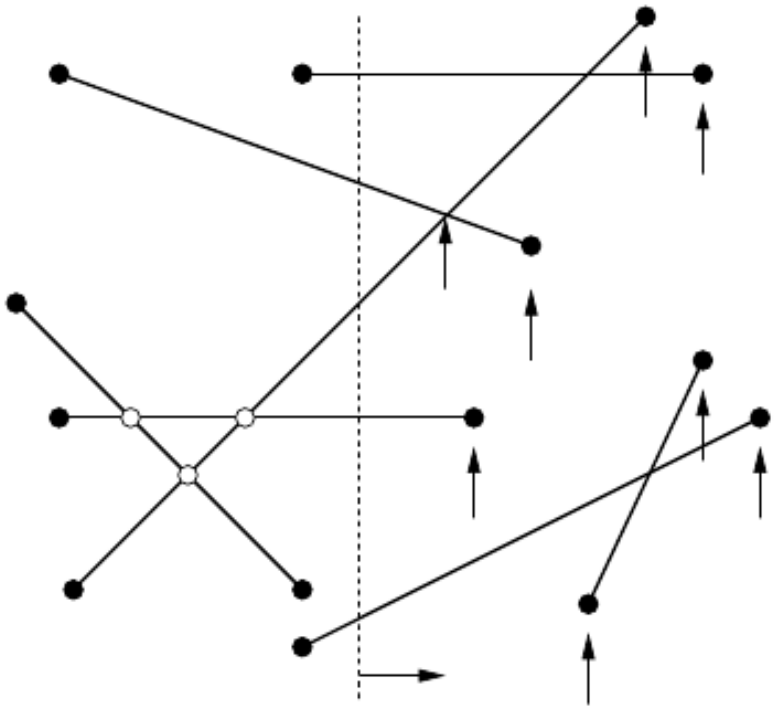
Cinvestav

Eventos

- Por ejemplo, en la siguiente figura algunos de los puntos de intersección situados a la derecha de la línea de barrido no han sido aún descubiertos como eventos
- Sin embargo, mostraremos más adelante que cada punto de intersección será descubierto como un evento antes de que la línea de barrido llegue a él

Cinvestav

Eventos



2. Algoritmo de barrido del plano (plane sweep)
   - Introducción
   - Línea de barrido
   - Eventos
   - Actualización de eventos
   - Detección de intersecciones
   - Estructuras de datos
   - Algoritmo de barrido del plano
   - Ejemplo
   - Análisis de complejidad

Cinvestav

Actualización de eventos

- Cuando un evento es encontrado las estructuras de datos asociados con ese evento deben ser actualizadas
- Es importante tener cuidado en especificar exactamente las invariantes que se van a mantener
- Por ejemplo, cuando nos encontramos con un punto de intersección, debemos intercambiar el orden de los segmentos de línea de intersección a lo largo de la línea de barrido

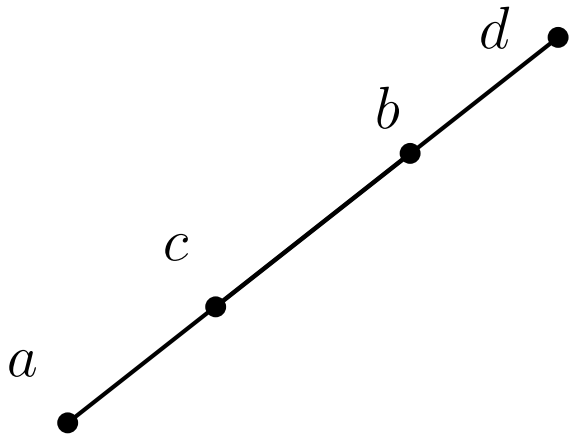
Cinvestav

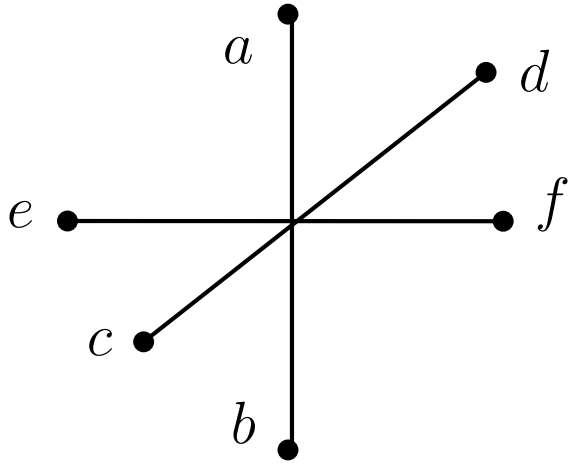
Actualización de eventos

- Hay un gran número de casos especiales que complican el funcionamiento del algoritmo
- Sin embargo, haremos una serie de supuestos para simplificar este problema:
  1. Ningún segmento de línea es vertical
  2. Si dos segmentos se intersectan, entonces lo harán en un solo punto (i.e., no son colineales)
  3. No existen tres segmentos de línea que se intersecten en un punto común

Cinvestav

Actualización de eventos





2. Algoritmo de barrido del plano (plane sweep)
   - Introducción
   - Línea de barrido
   - Eventos
   - Actualización de eventos
   - Detección de intersecciones
   - Estructuras de datos
   - Algoritmo de barrido del plano
   - Ejemplo
   - Análisis de complejidad

Cinvestav

Detección de intersecciones

- Recordemos que se mencionó que los *eventos de extremos* son todos conocidos de antemano
- Sin embargo, ¿cómo detectar los eventos de intersección?
- Para el algoritmo es importante que cada evento pueda ser detectado antes de que ocurra

Cinvestav

Detección de intersecciones

- La estrategia es la siguiente, cuando dos segmentos de línea son adyacentes a lo largo de la línea de barrido, se comprobará si tienen una intersección que ocurren a la derecha de la línea de barrido
- Si es así, añadimos este nuevo evento (suponiendo que no haya sido ya agregado)
- Una pregunta natural es si ésto es suficiente
- En particular, si dos segmentos de línea se intersectan, existe necesariamente un posicionamiento previo de la línea de barrido de tal manera que sean adyacentes
- Afortunadamente, este es el caso, pero requiere una prueba

Cinvestav

Detección de intersecciones

Lema

Dados dos segmentos s_i y s_j, que se intersectan en un único punto p (y suponiendo que ningún otro segmento de línea pasa por este punto) hay un posicionamiento de la línea de barrido previo a este evento, de manera que s_i y s_j son adyacentes a lo largo de la línea de barrido (y por lo tanto, se probará su intersección)

Cinvestav

Detección de intersecciones

Prueba

- A partir de nuestro supuesto de la posición general, no existen tres líneas que se intersecten en un punto común
- Por lo tanto, si consideramos un posicionamiento de la línea de barrido que es infinitesimalmente a la izquierda del punto de intersección, las líneas s_i y s_j serán adyacentes a lo largo de esta línea de barrido.

Cinvestav

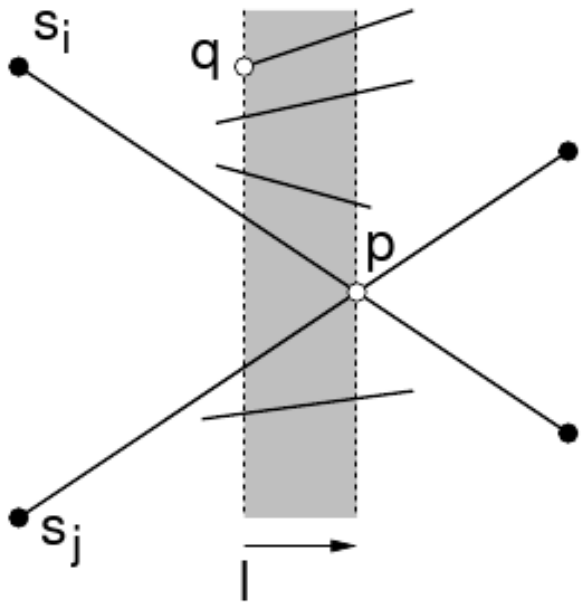
Detección de intersecciones

Prueba ..

- Considere el punto de evento q con la mayor coordenada x que es estrictamente menor que p_x
- El orden de las líneas a lo largo de la línea de barrido después de procesar q será idéntico al orden de las líneas a lo largo de la línea de barrido justo antes de p
- Por lo tanto, s_i y s_j serán adyacentes en este punto

Cinvestav

Detección de intersecciones



2. Algoritmo de barrido del plano (plane sweep)
   - Introducción
   - Línea de barrido
   - Eventos
   - Actualización de eventos
   - Detección de intersecciones
   - Estructuras de datos
   - Algoritmo de barrido del plano
   - Ejemplo
   - Análisis de complejidad

Cinvestav

Estructuras de datos

- En la implementación del algoritmo de barrido del plano se requiere el uso de dos estructuras de datos:
  1. Cola de eventos
  2. Estatus de la línea de barrido
- A continuación describiremos cada una de ellas

Cinvestav

Estructuras de datos

- La **cola de eventos** almacena un conjunto de eventos futuros, ordenados de manera creciente con respecto a la coordenada x
- Cada evento contiene la información auxiliar que permite identificar su tipo (punto extremo del segmento o de intersección), y qué segmento(s) está involucrado
- Las operaciones que esta estructura de datos debe soportar son:
  - La inserción de un evento (si no está ya presente en la cola)
  - Extraer el mínimo evento

Cinvestav

Estructuras de datos

- Pareciera que una estructura de datos de tipo *heap* sería ideal para ello, ya que soporta las inserciones y la extracción del mínimo en tiempo $O(\log M)$, donde M es el número de entradas en la cola
- Sin embargo, un *heap* no permite verificar la existencia de eventos duplicados
- Existen dos maneras de manejar esto, la primera es utilizar una estructura de datos más sofisticada, como un árbol binario balanceado o una lista por saltos (*skip list*)

Cinvestav

Estructuras de datos

- Esto añade un pequeño factor constante a la complejidad, pero permite verificar duplicados fácilmente
- La segunda es utilizar el *heap*, aceptar el hecho de que puede haber eventos duplicados y agregar algún mecanismo que permita detectar e ignorar los eventos duplicados
- El libro de Berg et al. recomienda usar un árbol binario balanceado

Cinvestav

Estructuras de datos

- Al utilizar esta estructura de datos si los eventos tienen la misma coordenada x, entonces podemos ordenar los puntos lexicográficamente con respecto a (x, y)
- Esto resulta en que los eventos se procesen de abajo hacia arriba a lo largo de la línea de barrido
- Y tiene el mismo efecto que si giraramos (hipotéticamente) la línea de barrido infinitesimalmente a la izquierda

Cinvestav

Estructuras de datos

- **Estatus de la línea de barrido**, esta estructura de datos se emplea para almacenar el estado actual de la línea de barrido
- Se mantiene un diccionario ordenado (por ejemplo, un árbol binario balanceado) cuyas entradas son apuntadores a los segmentos de línea, almacenados en orden creciente de la coordenada y a lo largo de la línea de barrido actual
- Normalmente, cuando se almacenan elementos en un árbol, los valores llave son constantes
- Dado que la línea de barrido varía, es necesario almacenar llaves "variables"

Cinvestav

Estructuras de datos

- Para ello, vamos a suponer que cada segmento de línea calcula una ecuación de la recta $y = mx + b$ como parte de su representación
- El valor llave en cada nodo del árbol es un apuntador a un segmento de línea
- Para calcular la coordenada y de algún segmento en la posición actual de la línea de barrido, simplemente se toma la coordenada x actual de la línea de barrido
- Este valor de x se utiliza en la ecuación de la recta para este segmento de línea obteniéndose la coordenada y
- La coordenada y resultante es entonces utilizada en las comparaciones

Cinvestav

Estructuras de datos

- Las operaciones que se requiere implementar en esta estructura de datos son:
  1. La eliminación de un segmento de línea
  2. La inserción de un segmento de línea
  3. El intercambio de la posición de dos segmentos de línea
  4. Determinar el predecesor y sucesor inmediato de cualquier elemento
- Mediante un árbol binario balanceado o una lista por saltos (*skip list*), estas operaciones se puede realizar en tiempo $O(\log n)$ cada una

Cinvestav

2 Algoritmo de barrido del plano (plane sweep)

- Introducción
- Línea de barrido
- Eventos
- Actualización de eventos
- Detección de intersecciones
- Estructuras de datos
- Algoritmo de barrido del plano
- Ejemplo
- Análisis de complejidad

Cinvestav

Algoritmo de barrido del plano

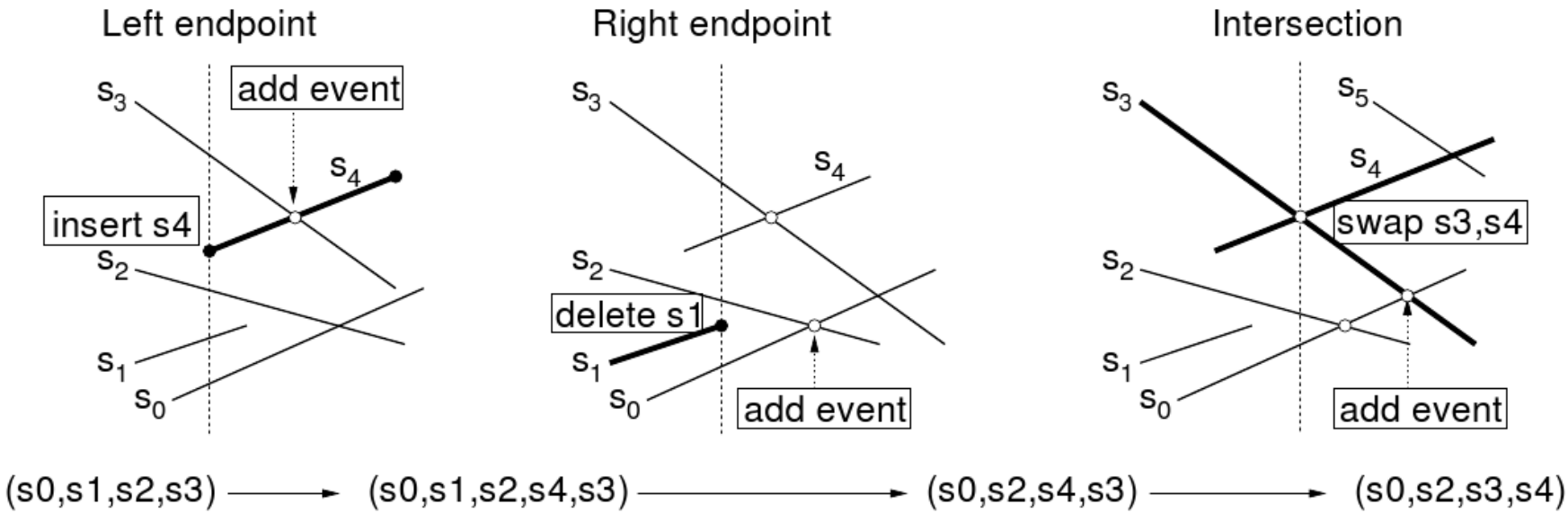
- El algoritmo completo de barrido del plano será presentado a continuación
- Nuestro algoritmo considera la creación de nuevos eventos, pero no toma en cuenta el problema de eventos duplicados
- En el libro de Berg et al. se muestra una implementación que toma en cuenta este detalle

Cinvestav

Algoritmo de barrido del plano

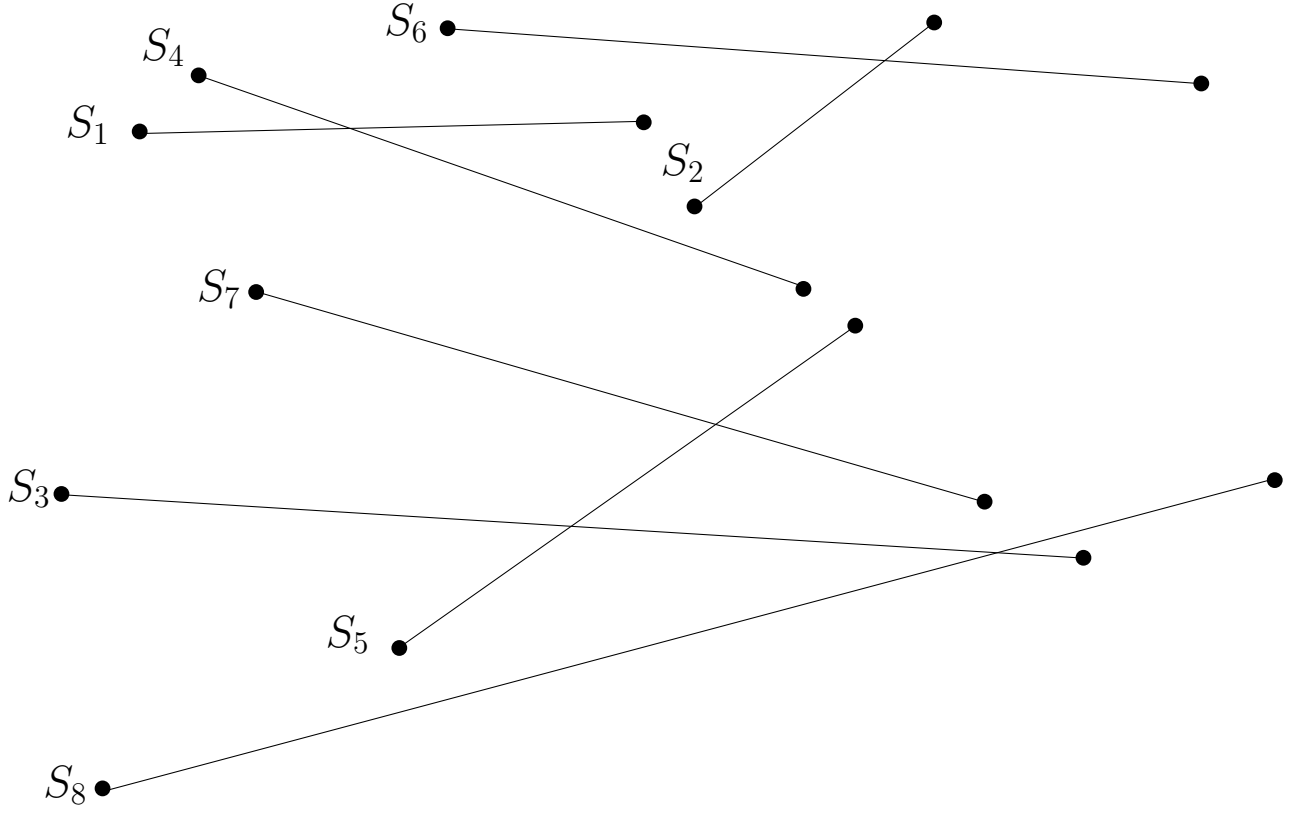
1. Insertar todos los puntos extremos de los segmentos de línea de S en la *cola de eventos* $\mathcal{Q}$
2. Inicializar a vacio el *estatus de la línea de barrido* $\mathcal{T}$
3. **while** $\mathcal{Q} \neq \emptyset$ extraer el próximo evento de la cola. Existen tres casos dependiendo del tipo de evento:
   - **Punto extremo izquierdo:** Insertar este segmento de línea en $\mathcal{T}$, basado en las coordenadas y tanto de este punto extremo como de los segmentos actualmente a lo largo de la línea de barrido. Verificar las intersecciones con el segmento inmediato superior e inmediato inferior
   - **Punto extremo derecho:** Borrar este segmento de línea de $\mathcal{T}$. Verificar las intersecciones con el segmento inmediato anterior e inmediato posterior
   - **Punto de intersección:** Intercambiar el orden de los dos segmentos a lo largo de la línea de barrido. Para el nuevo segmento superior, verificar de nuevo la intersección con el segmento inmediato anterior. Para el nuevo segmento inferior, verificar de nuevo la intersección con el segmento inmediato posterior

Algoritmo de barrido del plano



Cinvestav

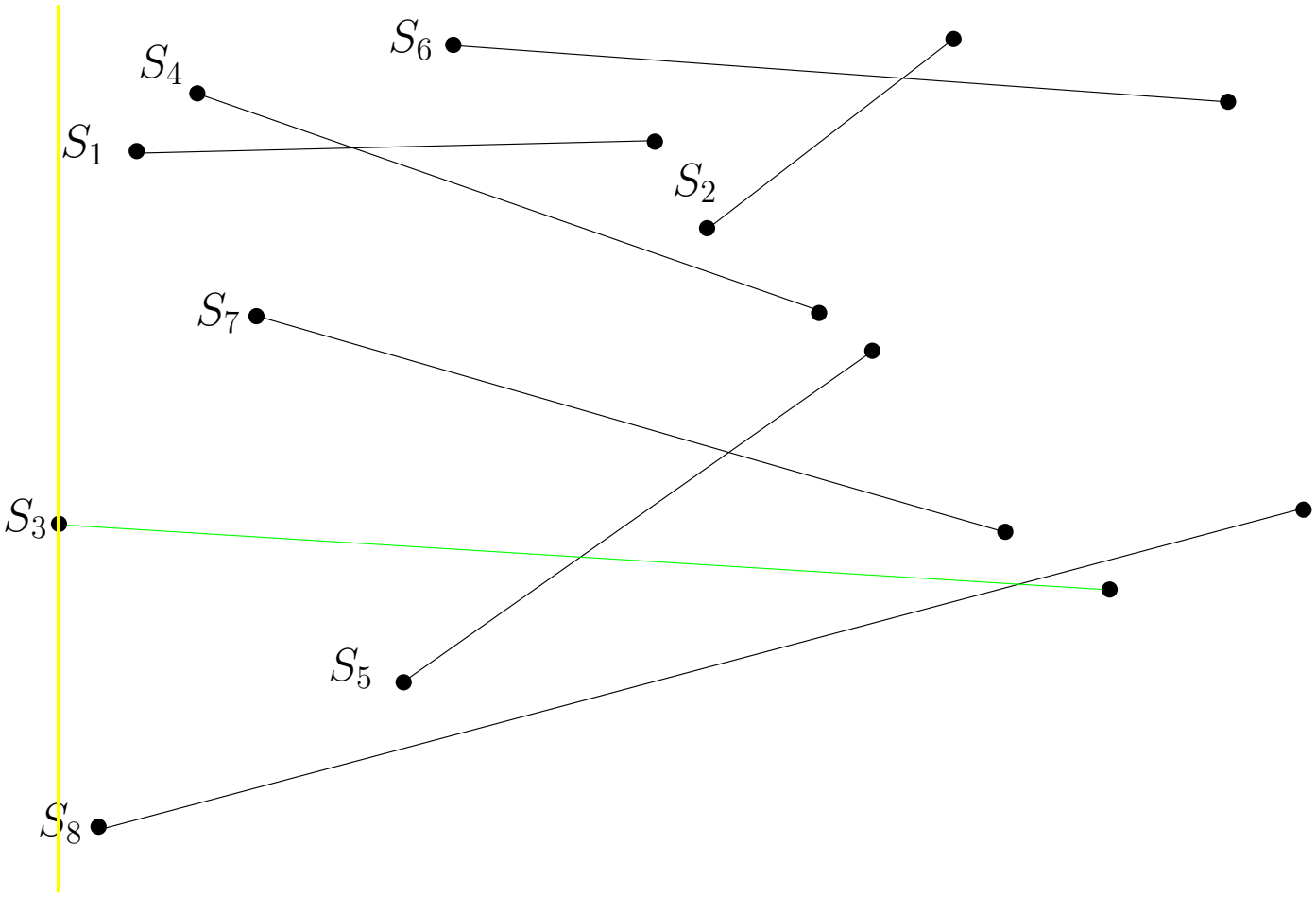
Ejemplo



$\mathcal{T}$

$\emptyset$

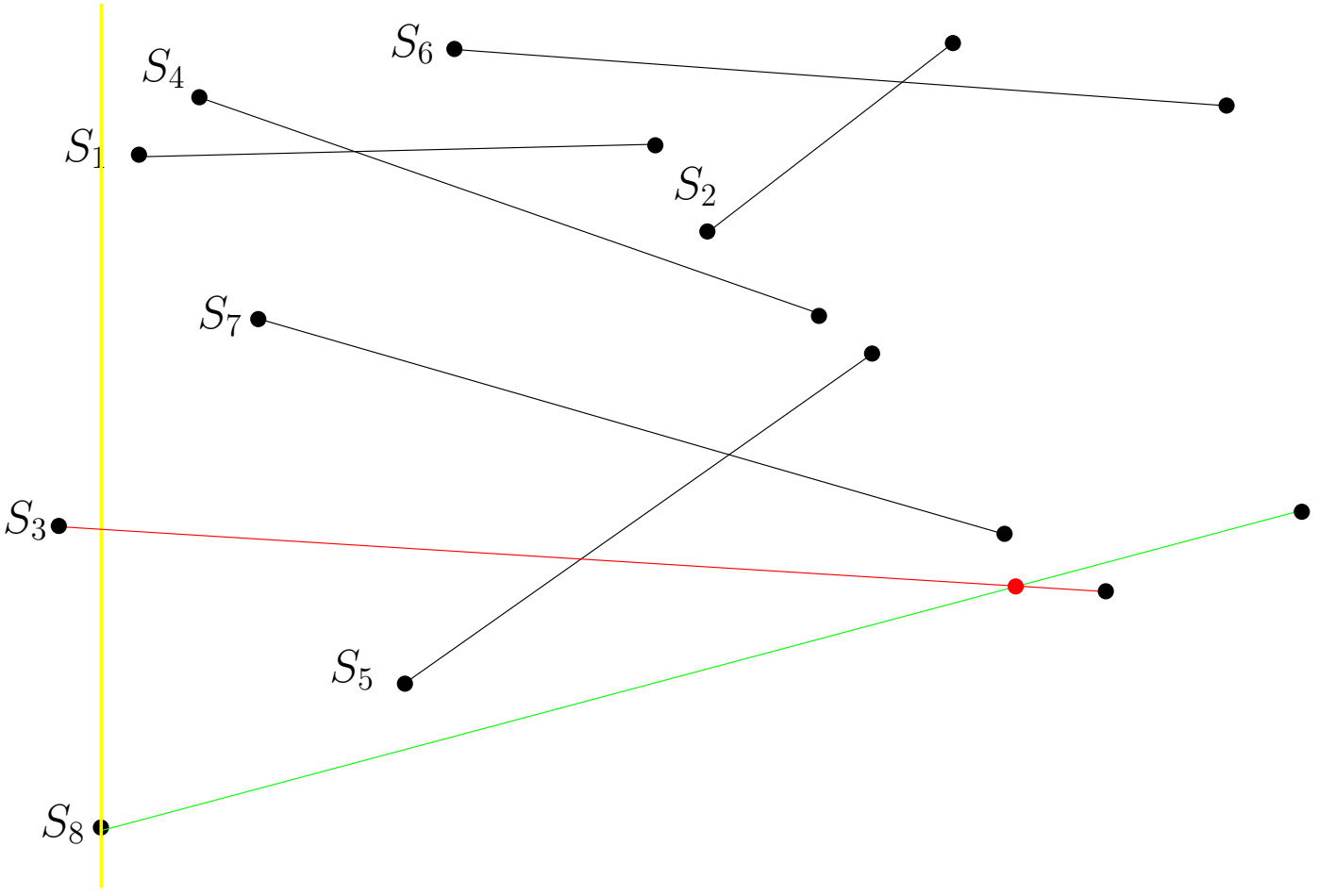
Ejemplo



$\mathcal{T}$

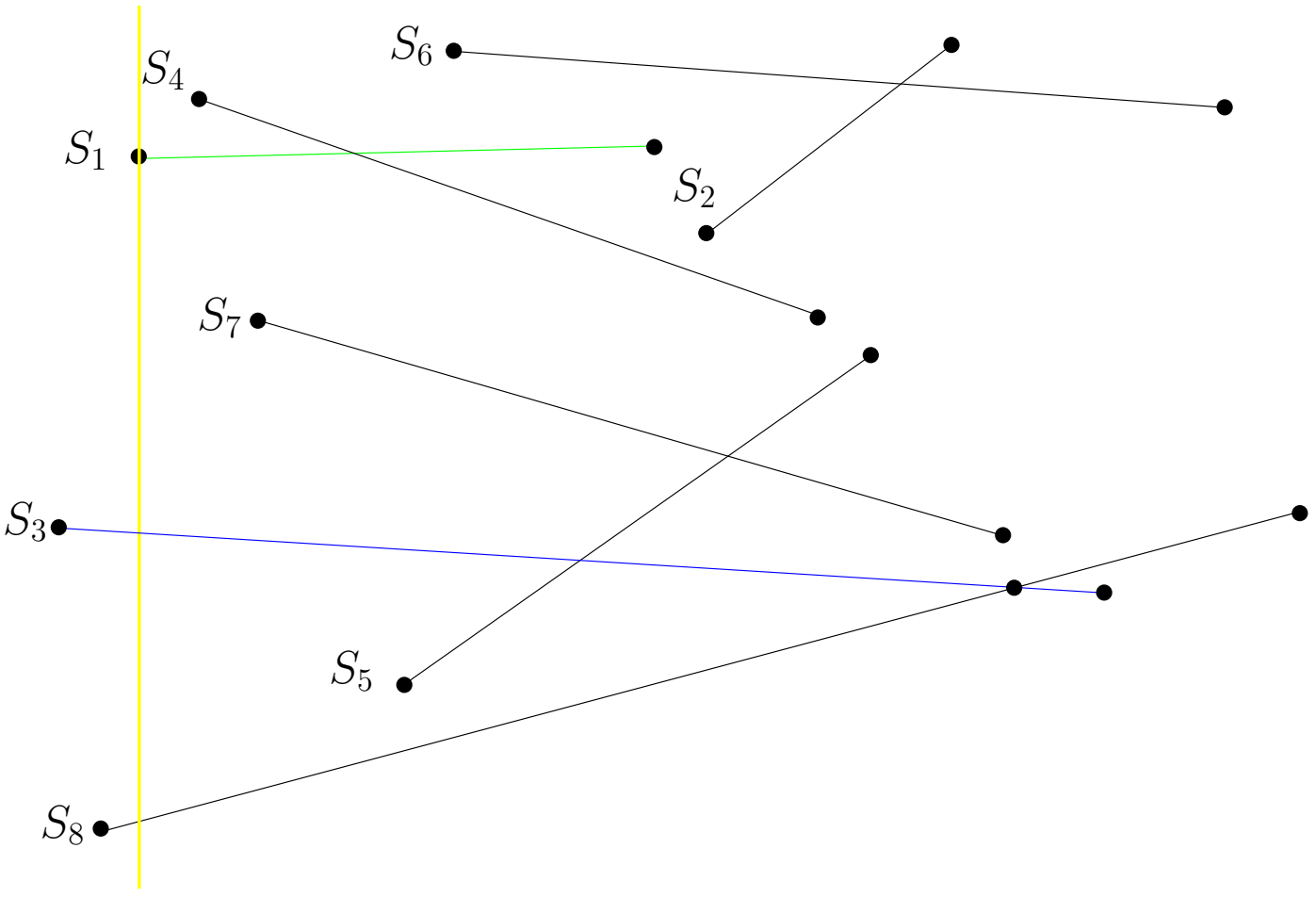
S_3

Ejemplo



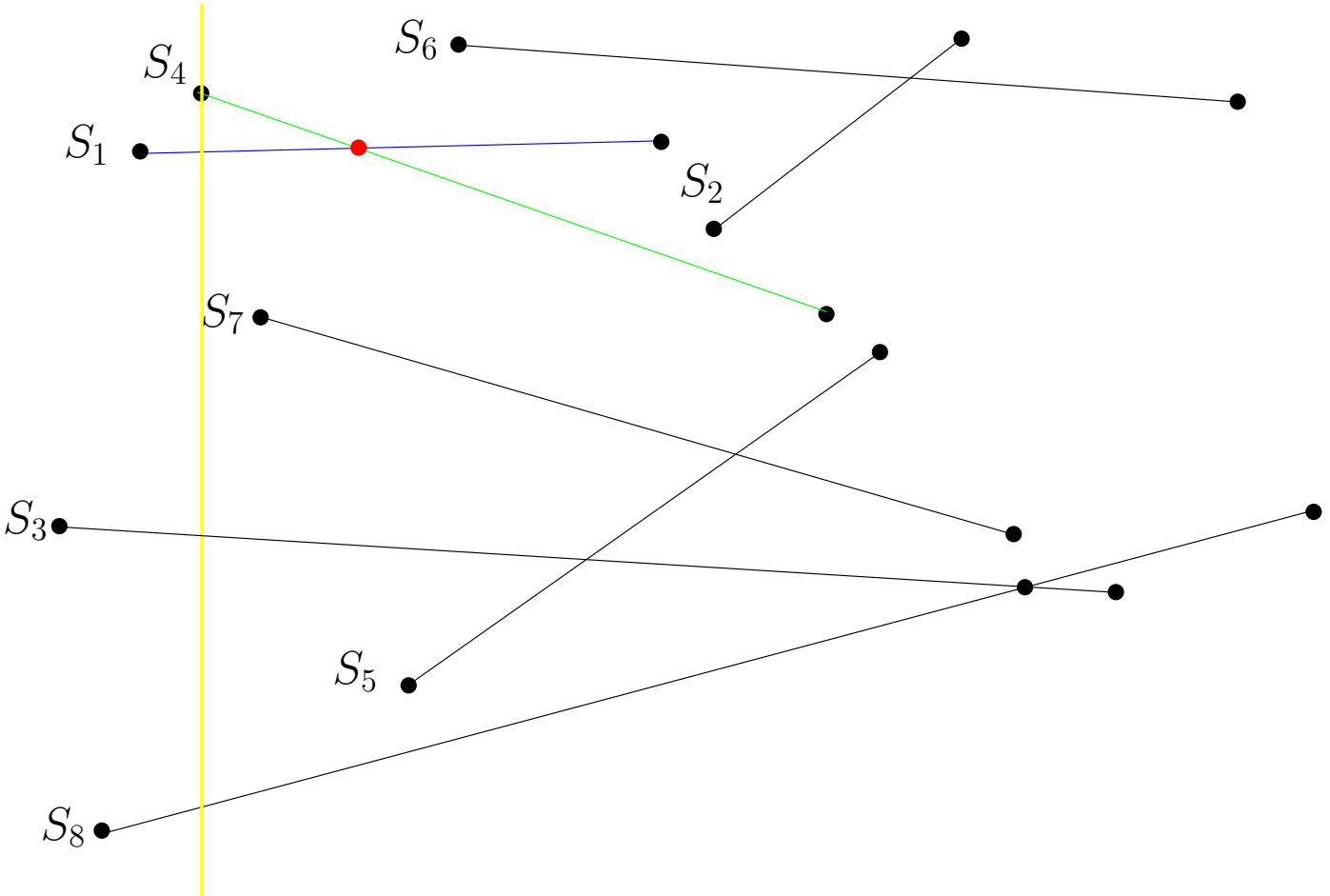
$\mathcal{T}$
S_3
S_8

Ejemplo



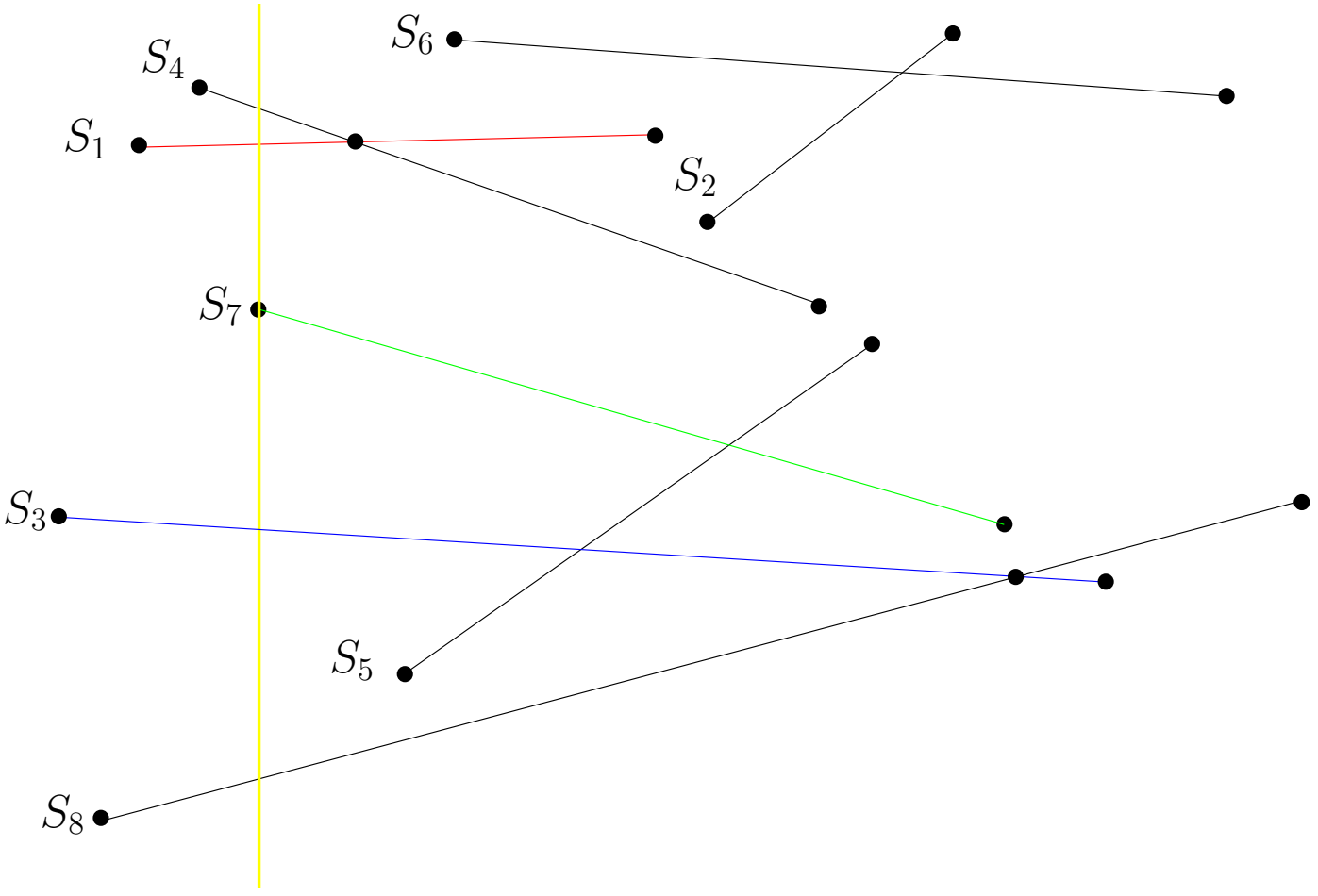
$\mathcal{T}$
S_1
S_3
S_8

Ejemplo



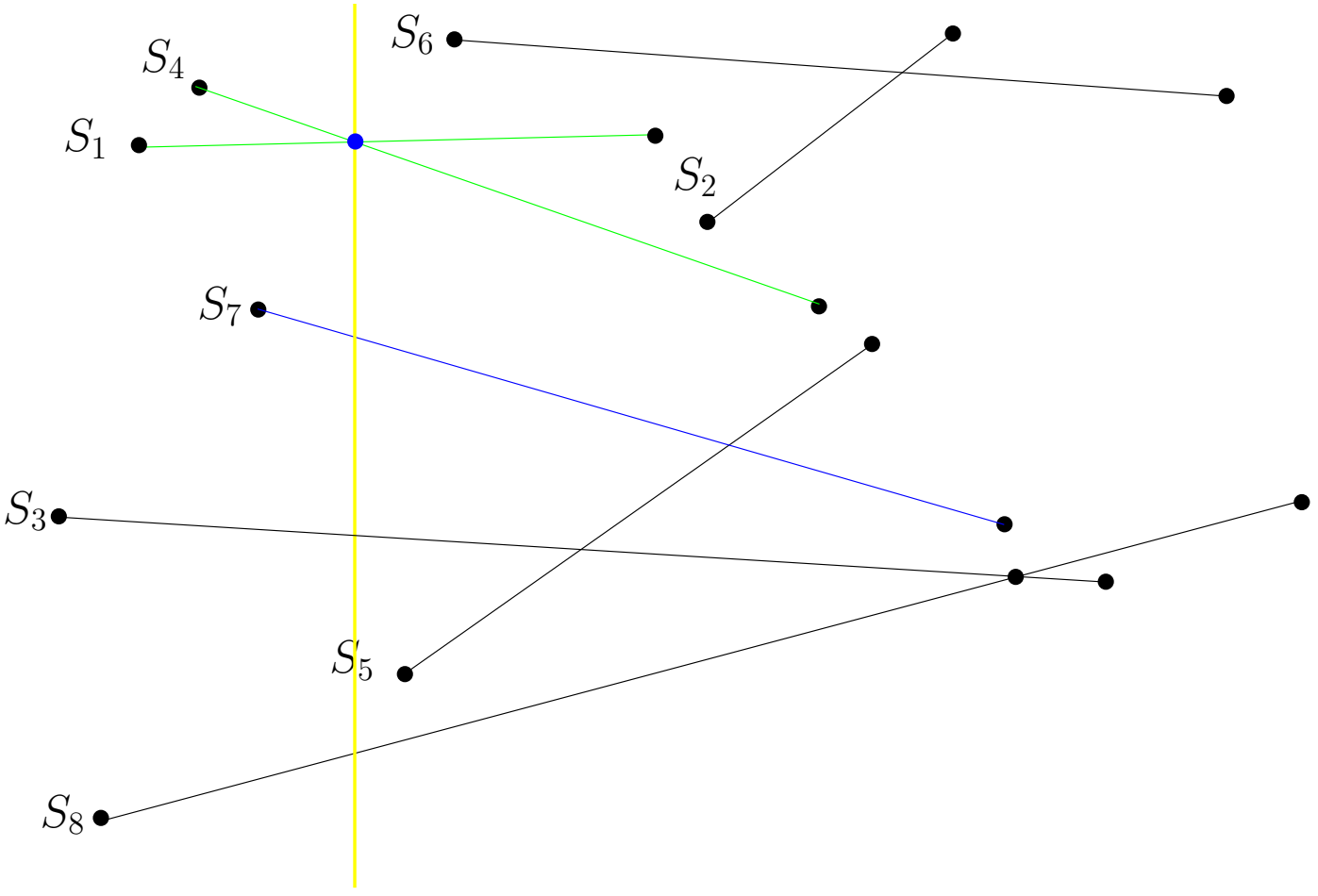
$\mathcal{T}$
S_4
S_1
S_3
S_8

Ejemplo



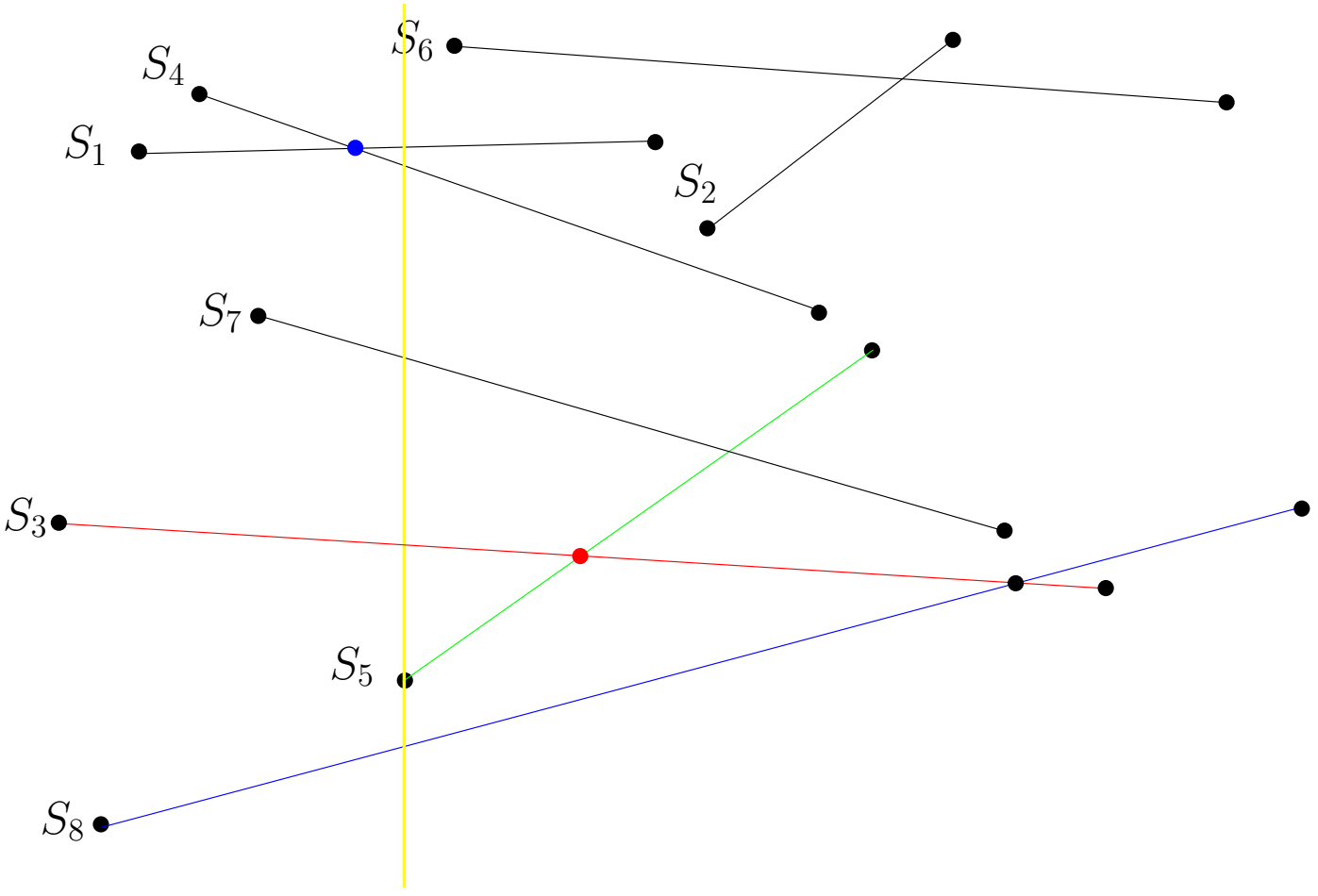
$\mathcal{T}$
S_4
S_1
S_7
S_3
S_8

Ejemplo



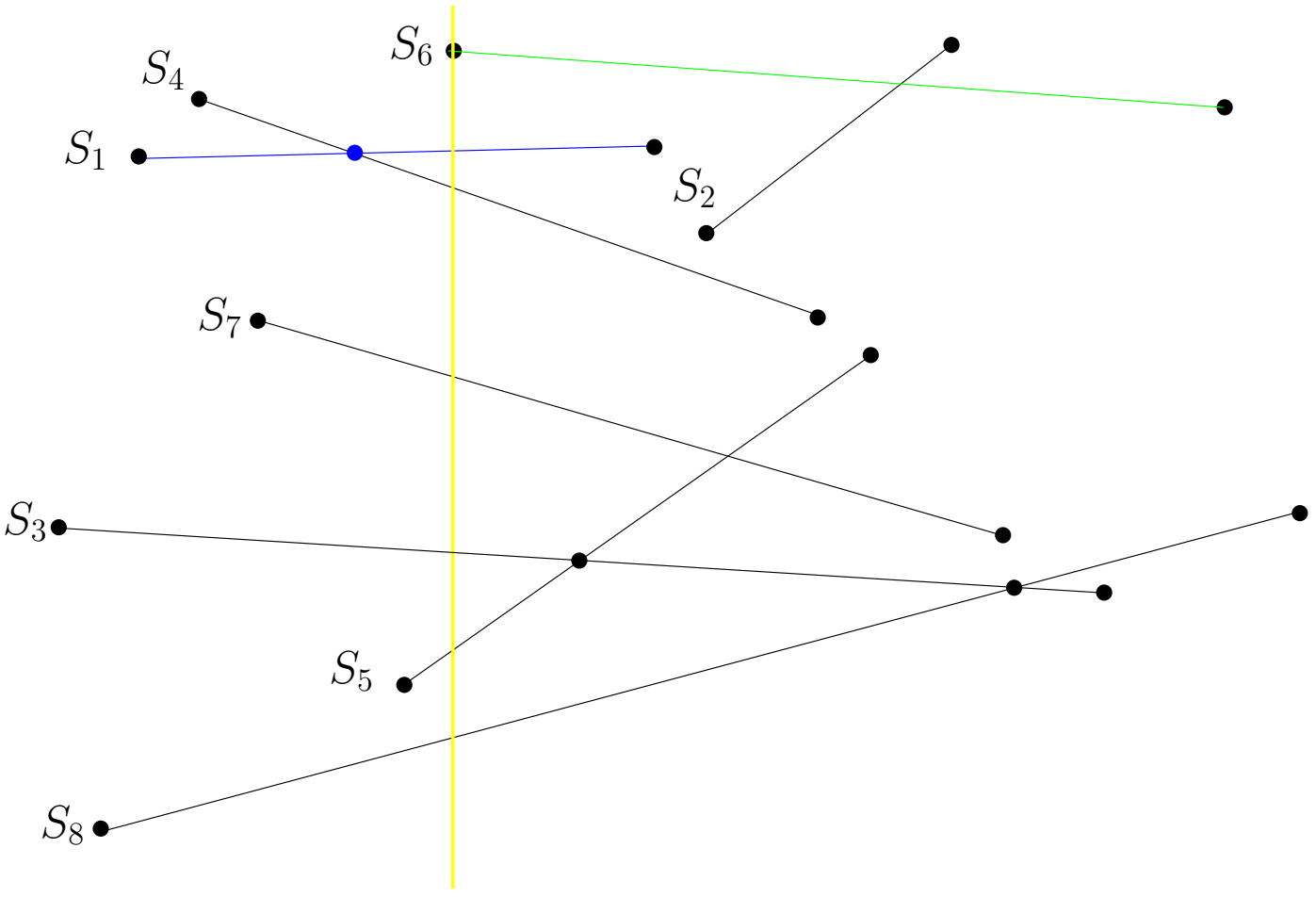
$\mathcal{T}$
S_1
S_4
S_7
S_3
S_8

Ejemplo



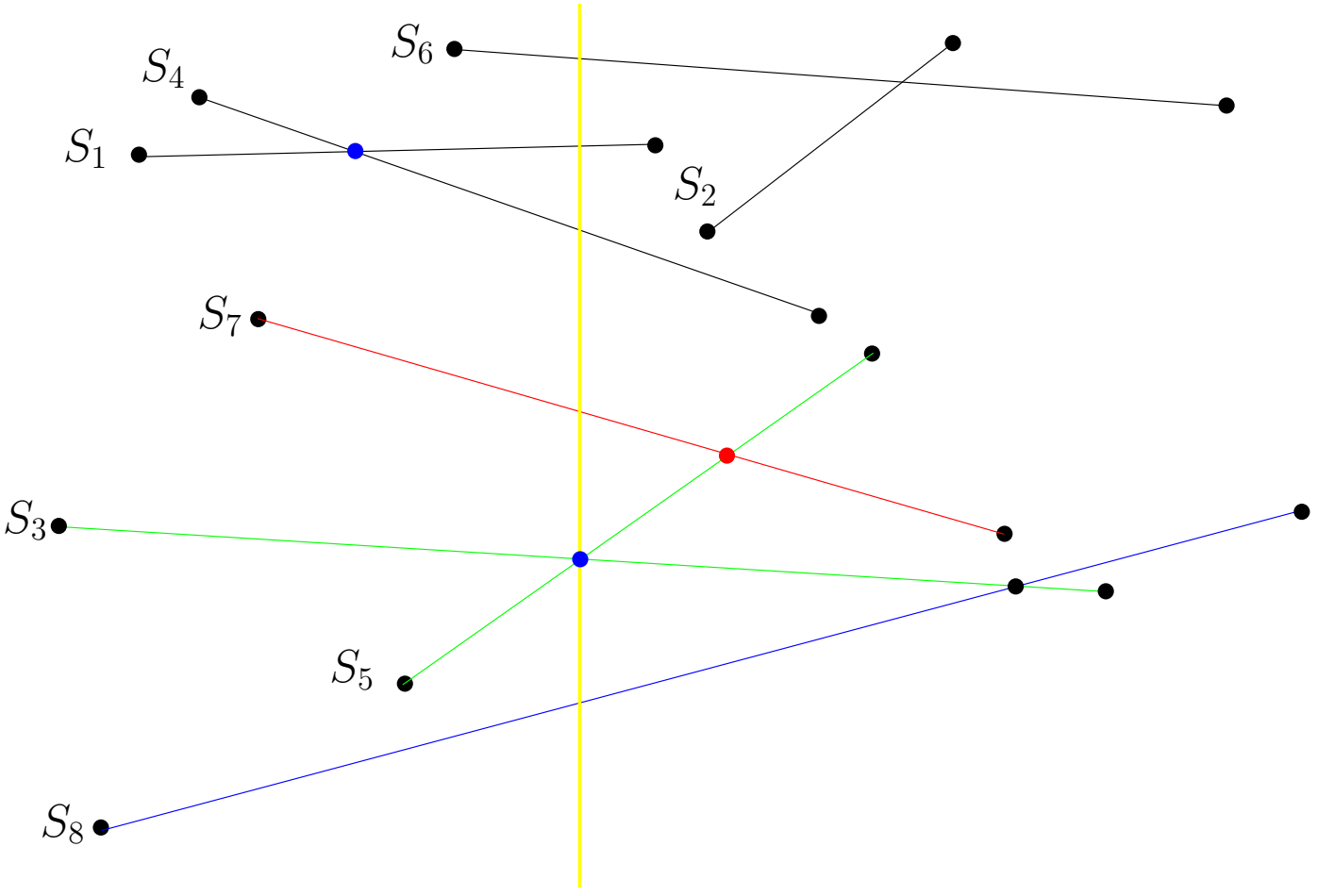
$\mathcal{T}$
S_1
S_4
S_7
S_3
S_5
S_8

Ejemplo



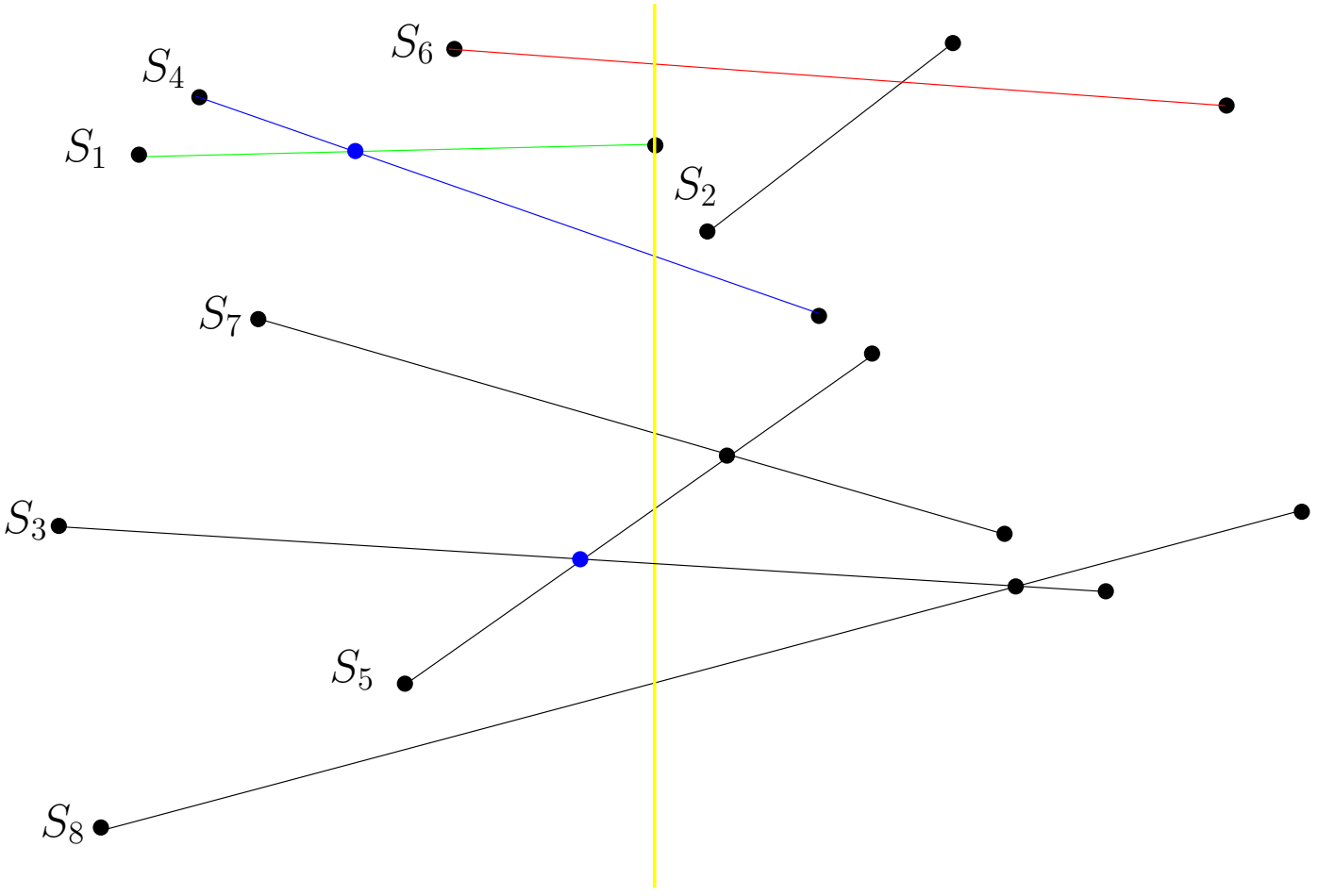
$\mathcal{T}$
S_6
S_1
S_4
S_7
S_3
S_5
S_8

Ejemplo



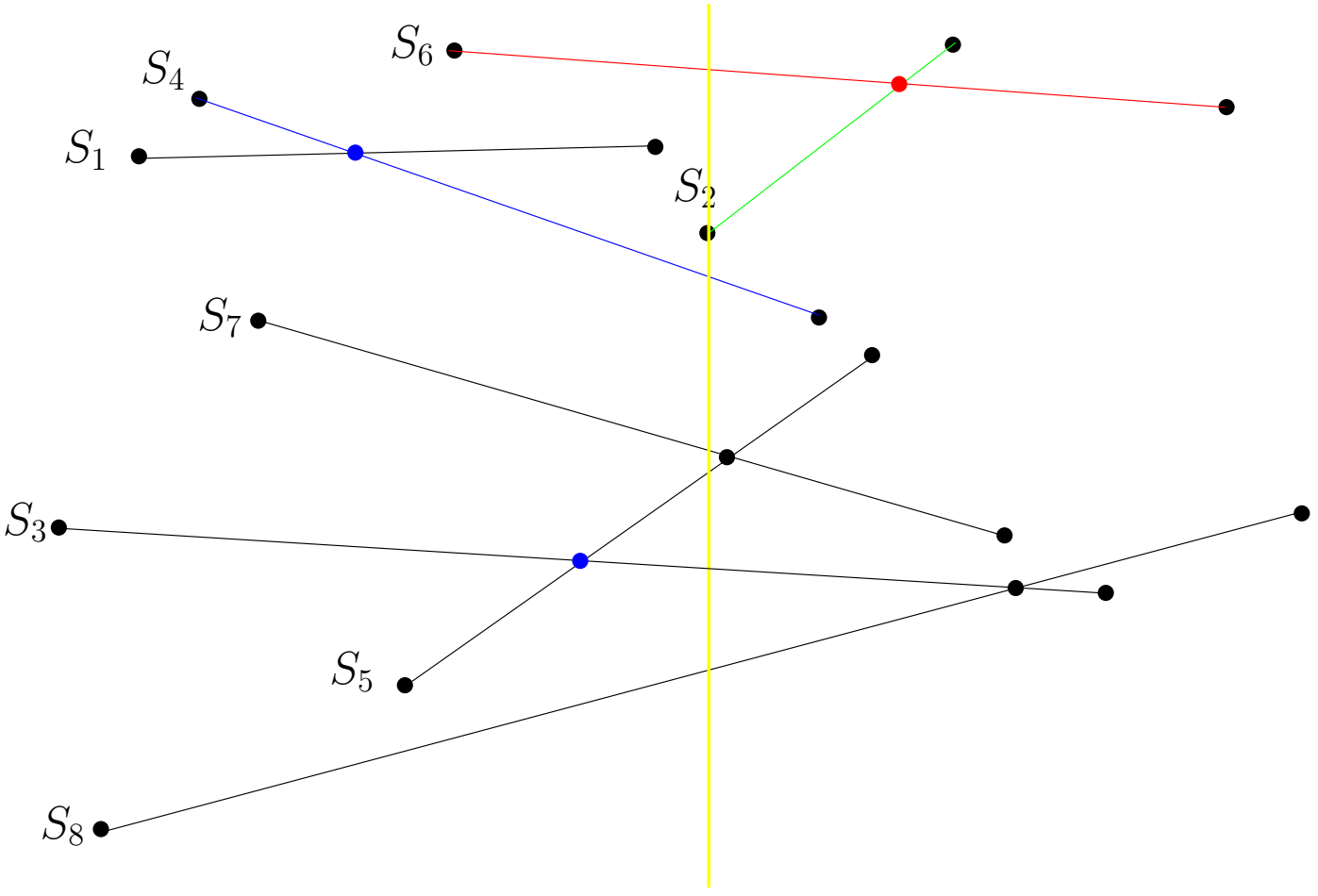
$\mathcal{T}$
S_6
S_1
S_4
S_7
S_5
S_3
S_8

Ejemplo



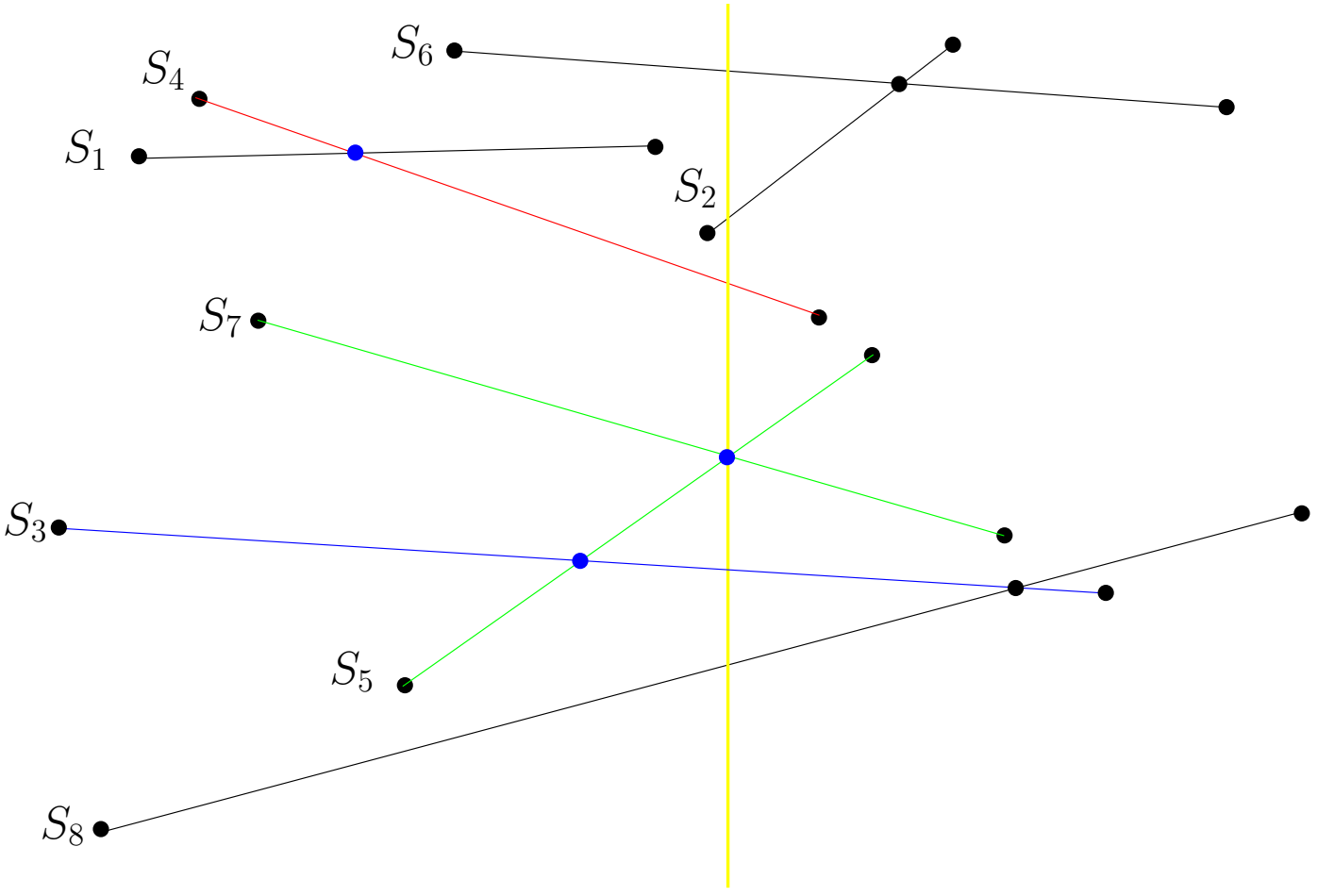
$\mathcal{T}$
S_6
S_4
S_7
S_5
S_3
S_8

Ejemplo



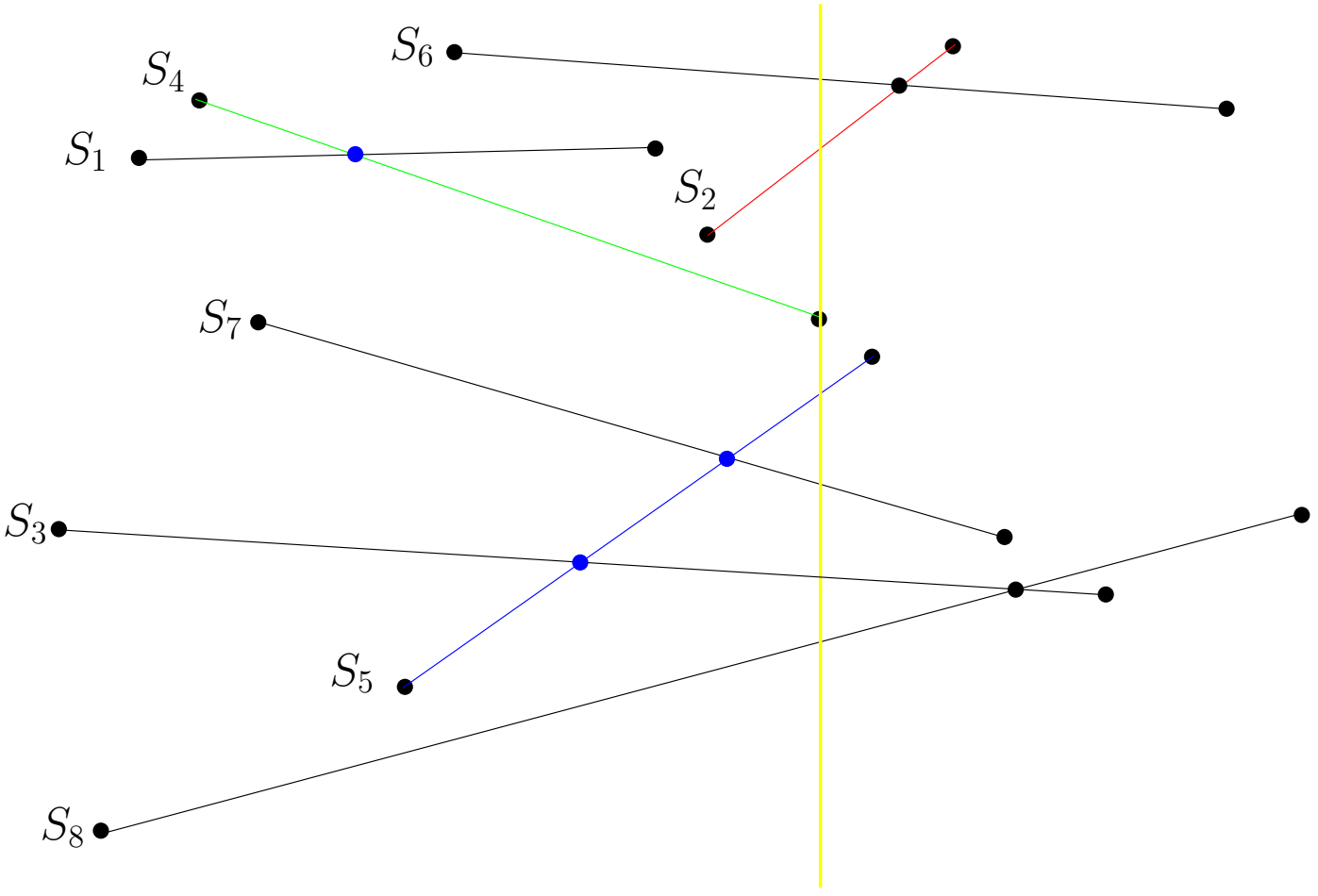
$\mathcal{T}$
S_6
S_2
S_4
S_7
S_5
S_3
S_8

Ejemplo



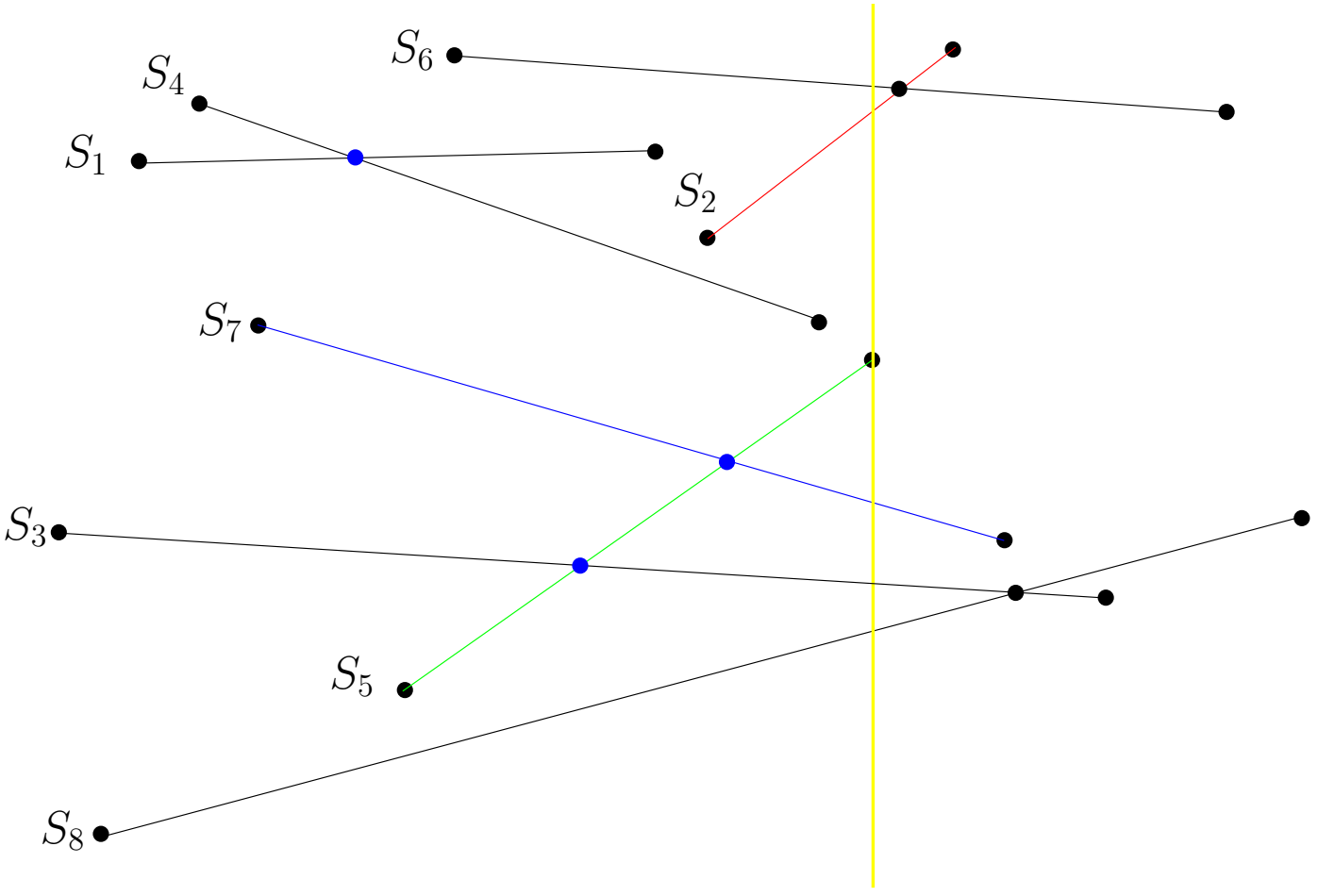
$\mathcal{T}$
S_6
S_2
S_4
S_5
S_7
S_3
S_8

Ejemplo



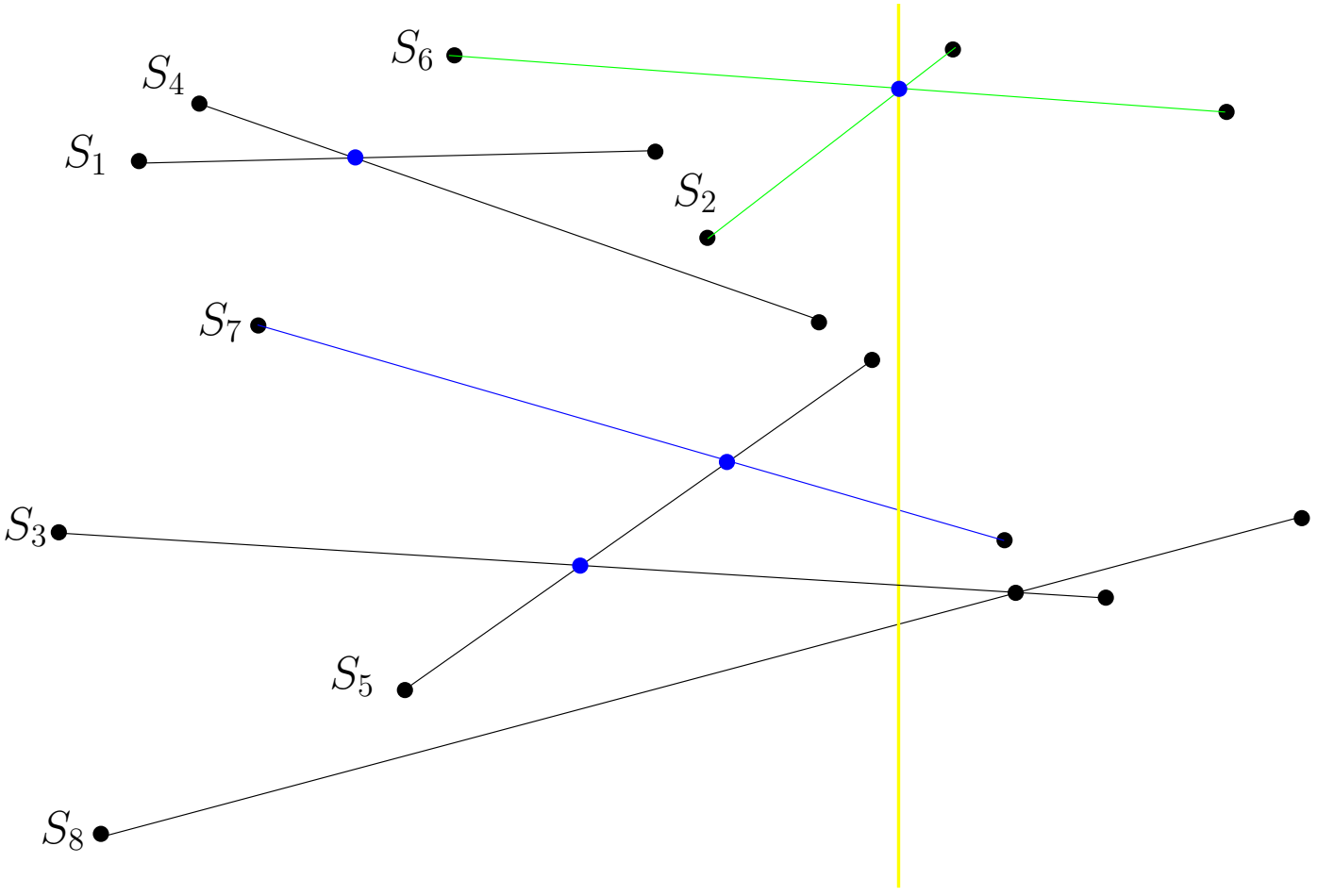
$\mathcal{T}$
S_6
S_2
S_5
S_7
S_3
S_8

Ejemplo



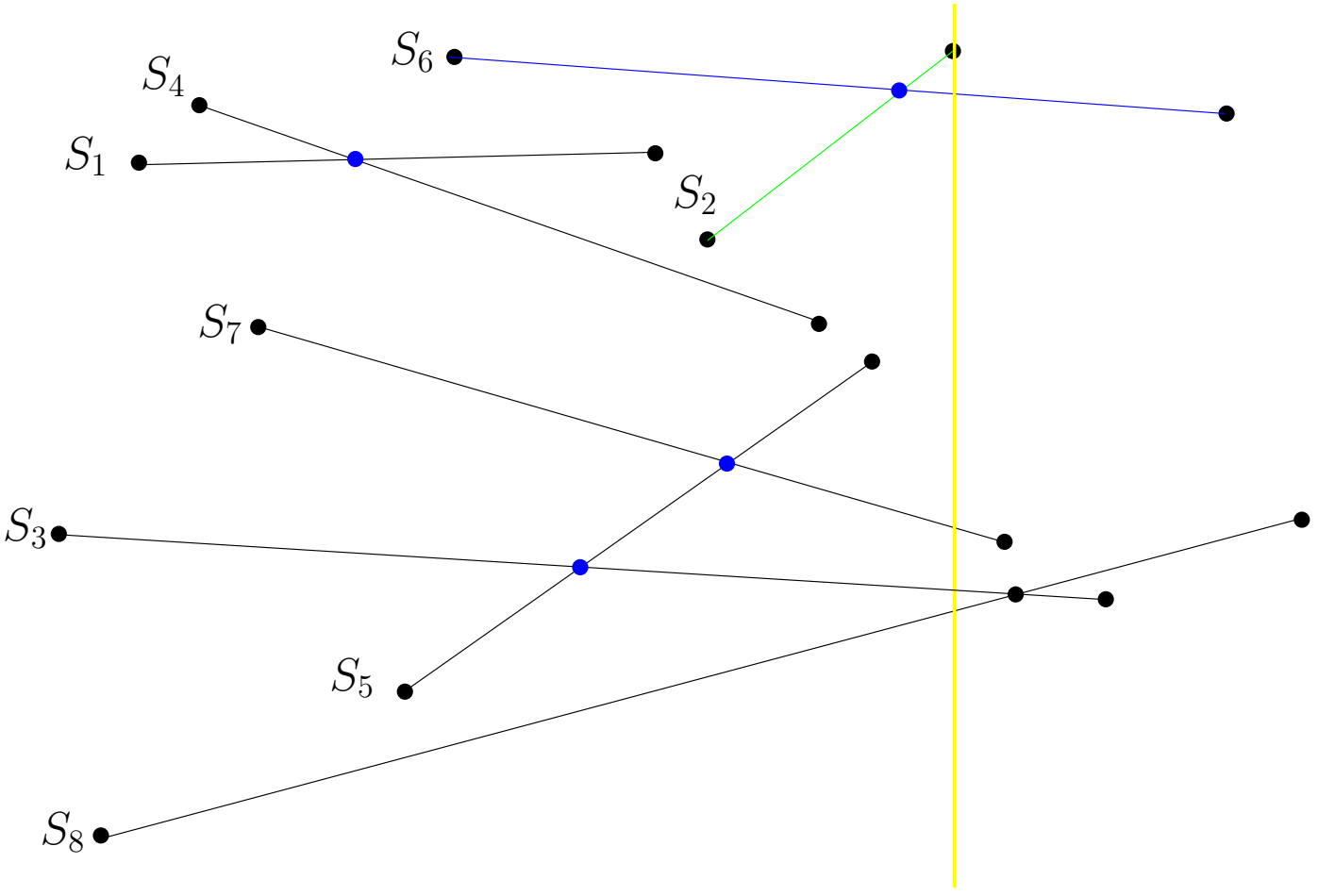
$\mathcal{T}$
S_6
S_2
S_7
S_3
S_8

Ejemplo



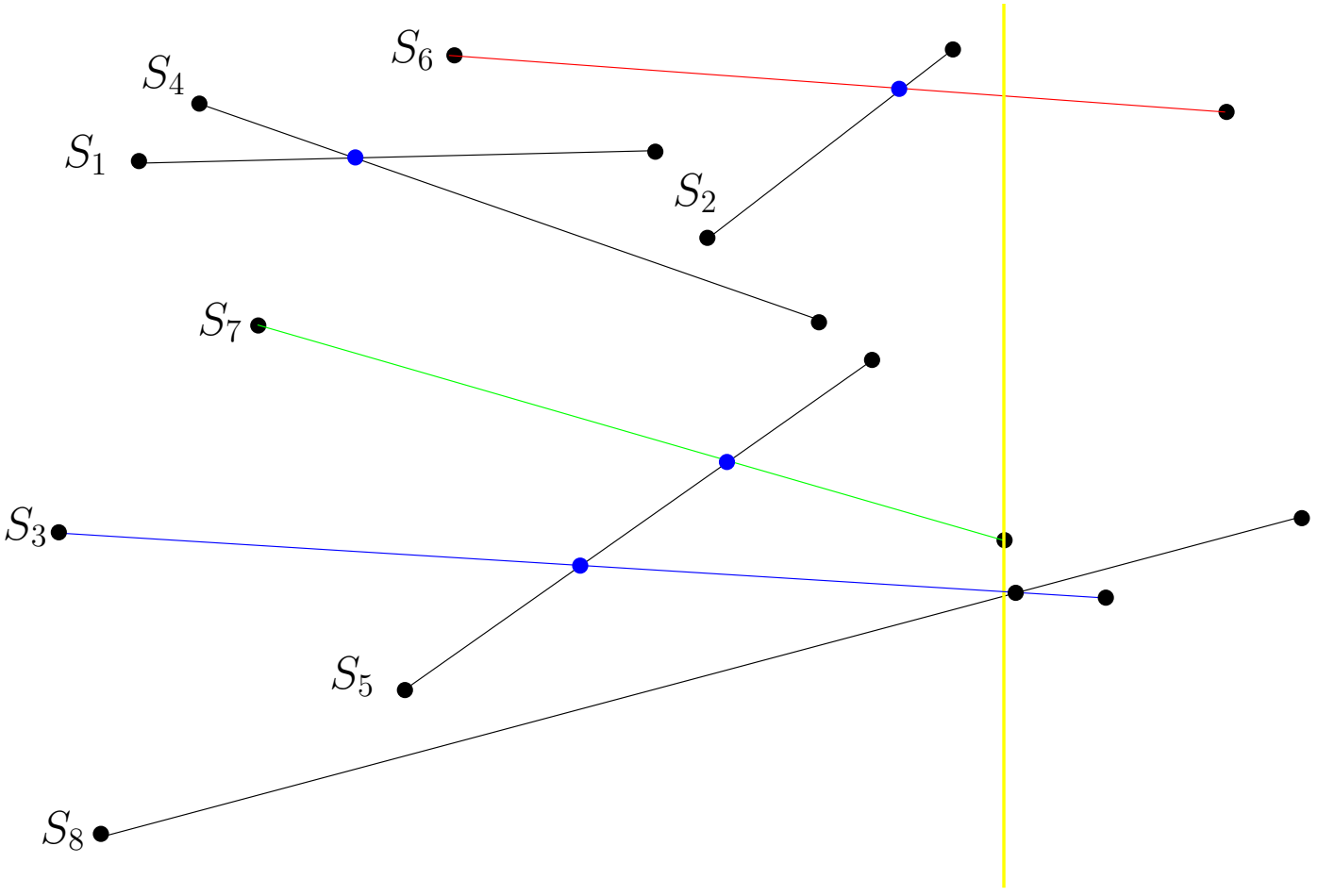
$\mathcal{T}$
S_2
S_6
S_7
S_3
S_8

Ejemplo



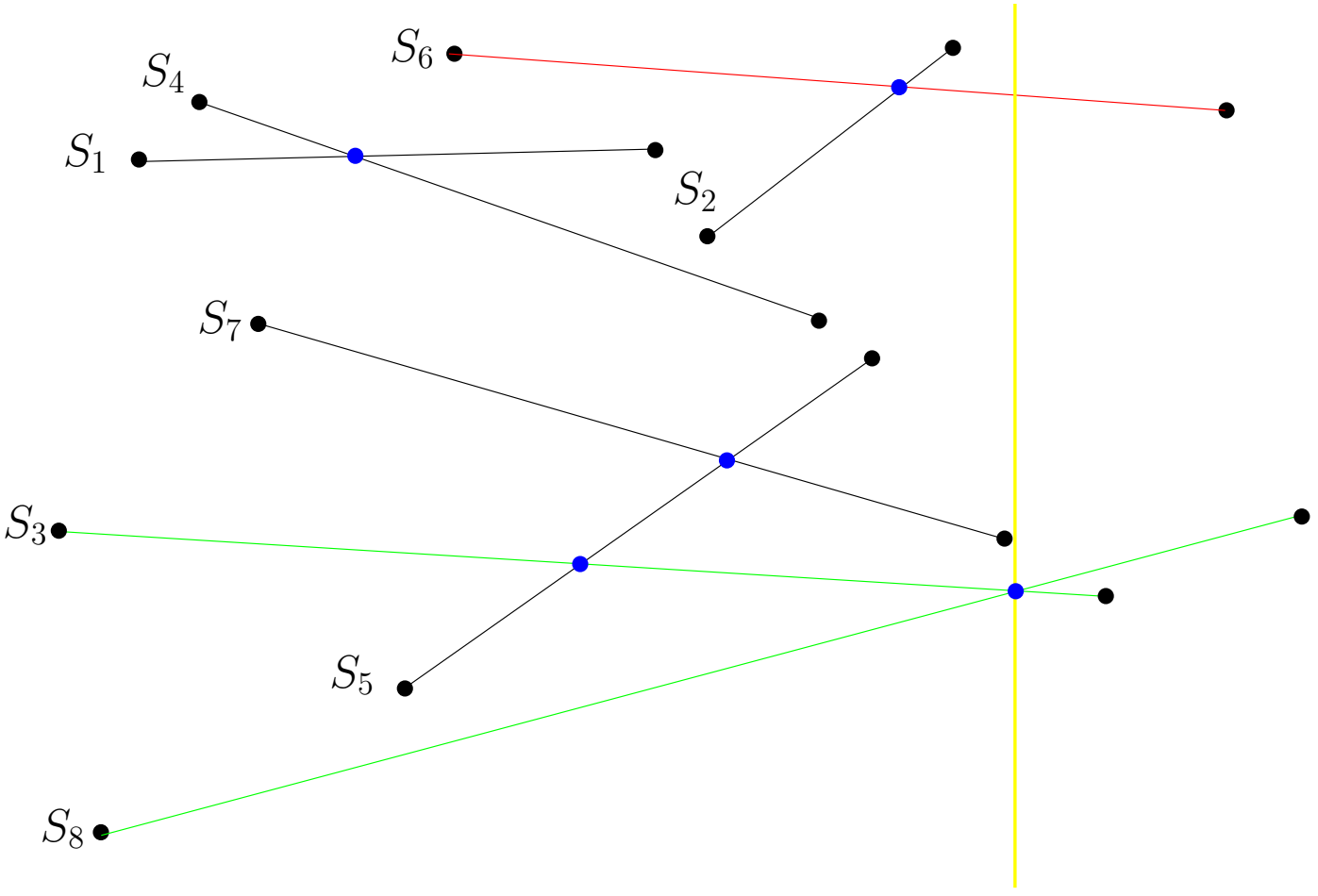
$\mathcal{T}$
S_6
S_7
S_3
S_8

Ejemplo



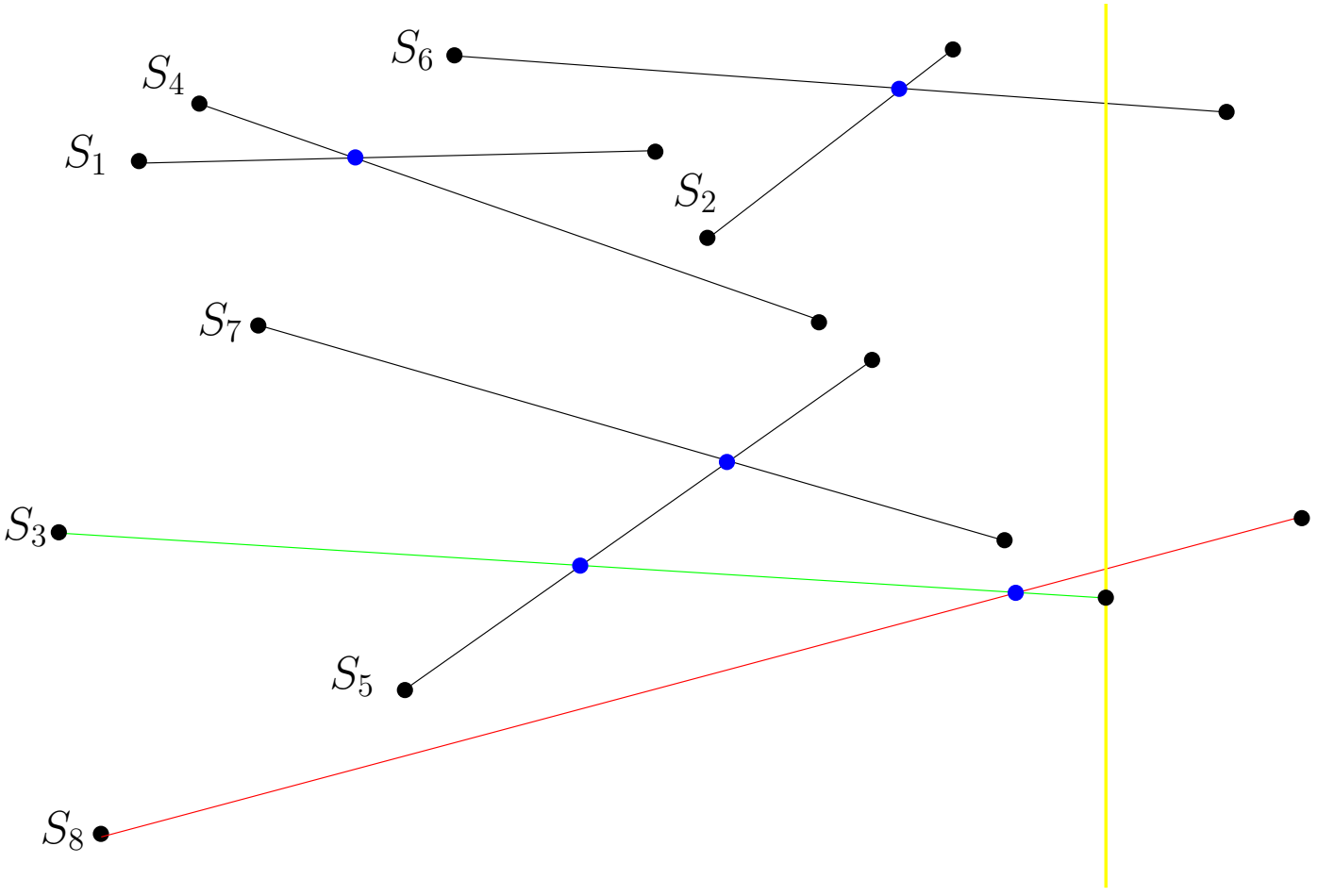
$\mathcal{T}$
S_6
S_3
S_8

Ejemplo



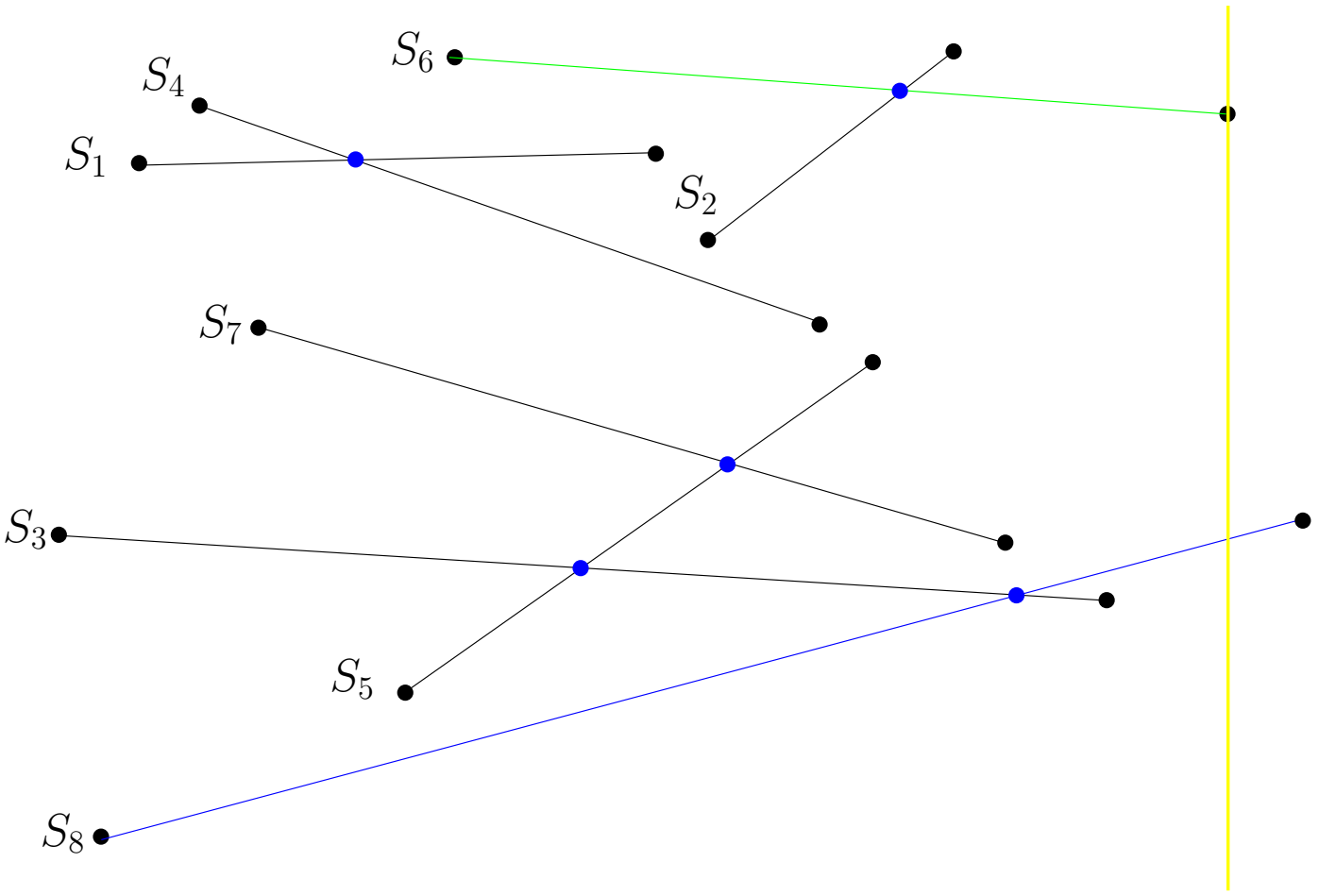
$\mathcal{T}$
S_6
S_8
S_3

Ejemplo



$\mathcal{T}$
S_6
S_8

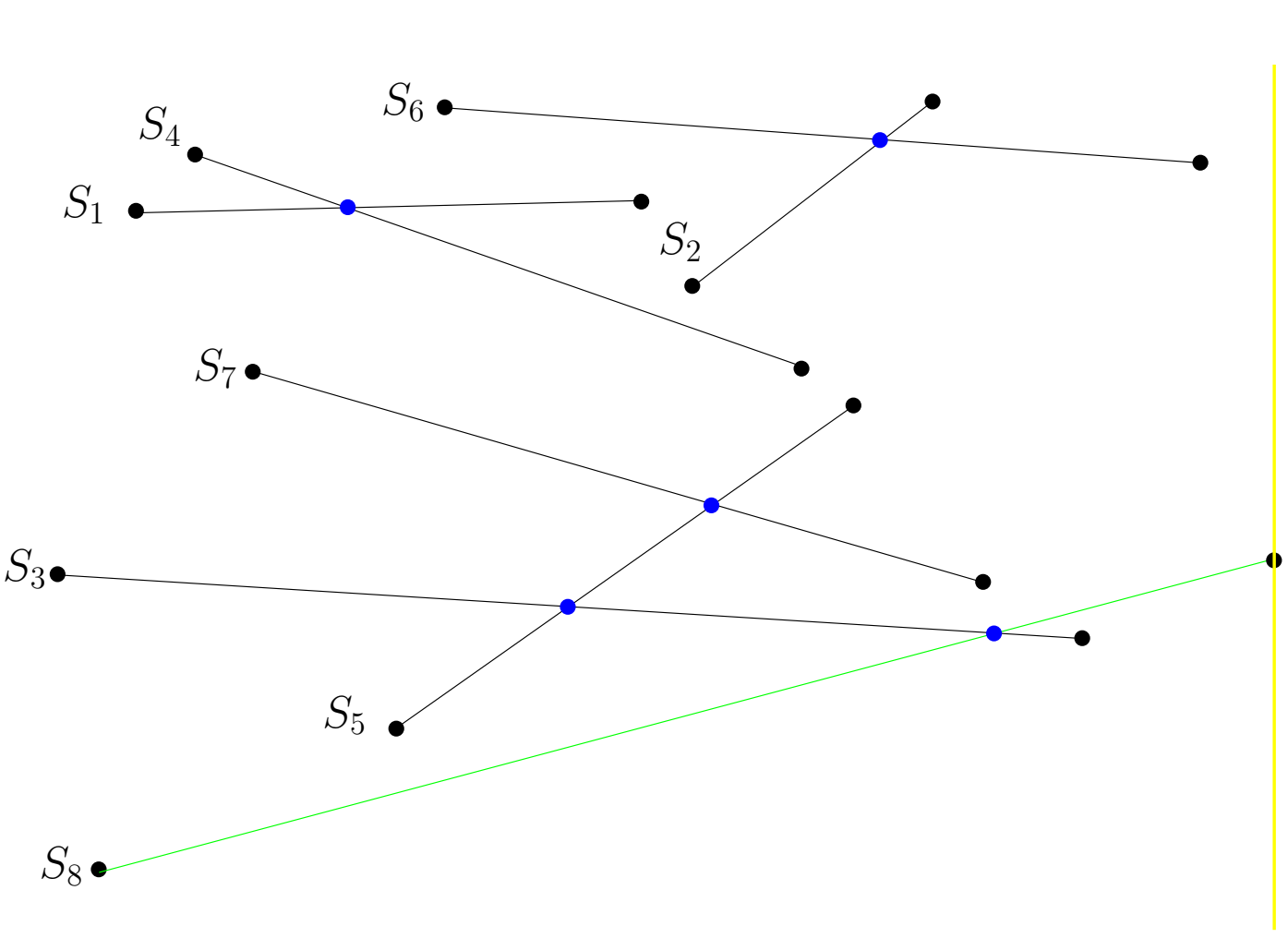
Ejemplo



$\mathcal{T}$

S_8

Ejemplo



2. Algoritmo de barrido del plano (plane sweep)
   - Introducción
   - Línea de barrido
   - Eventos
   - Actualización de eventos
   - Detección de intersecciones
   - Estructuras de datos
   - Algoritmo de barrido del plano
   - Ejemplo
   - Análisis de complejidad

Cinvestav

Análisis de complejidad

- El trabajo realizado por el algoritmo está dominado por el tiempo dedicado a la actualización de las diversas estructuras de datos (ya que por el contrario cada barrido toma solo un tiempo constante)
- Tenemos por lo tanto que contar dos cosas:
  - El número de operaciones aplicadas a cada estructura de datos
  - La cantidad de tiempo necesario para procesar cada operación

Cinvestav

Análisis de complejidad

- Para el *estatus de la línea de barrido*, hay como máximo n elementos intersectando la línea de barrido en cualquier momento
- Por lo tanto el tiempo necesario para realizar cualquier operación es $O(\log n)$ (árboles binarios balanceados)
- Ya que no se permiten eventos duplicados en la cola, el número total de elementos en la cola en cualquier momento es como máximo $2n + I$

Cinvestav

Análisis de complejidad

- Debido a que usamos un árbol binario balanceado para almacenar la *cola de eventos*, cada operación toma un tiempo como máximo logarítmico en el tamaño de la cola, que es $O(\log(2n + I))$
- Dado que $I \leq n^2$, esto es a lo sumo un tiempo $O(\log n^2) = O(2 \log n) = O(\log n)$
- Cada evento implica un número constante de accesos o de operaciones sobre la estructura de *estatus de la línea de barrido*

Cinvestav

Análisis de complejidad

- Puesto que cada operación de ese tipo toma un tiempo $O(\log n)$, se deduce que el tiempo total empleado para procesar todos los eventos de la línea de barrido es:

$$O((2n + I)\log n) = O((n + I)\log n) = O(n \log n + I \log n). \quad (4)$$

- Este es el tiempo total de ejecución del algoritmo de barrido del plano

Cinvestav