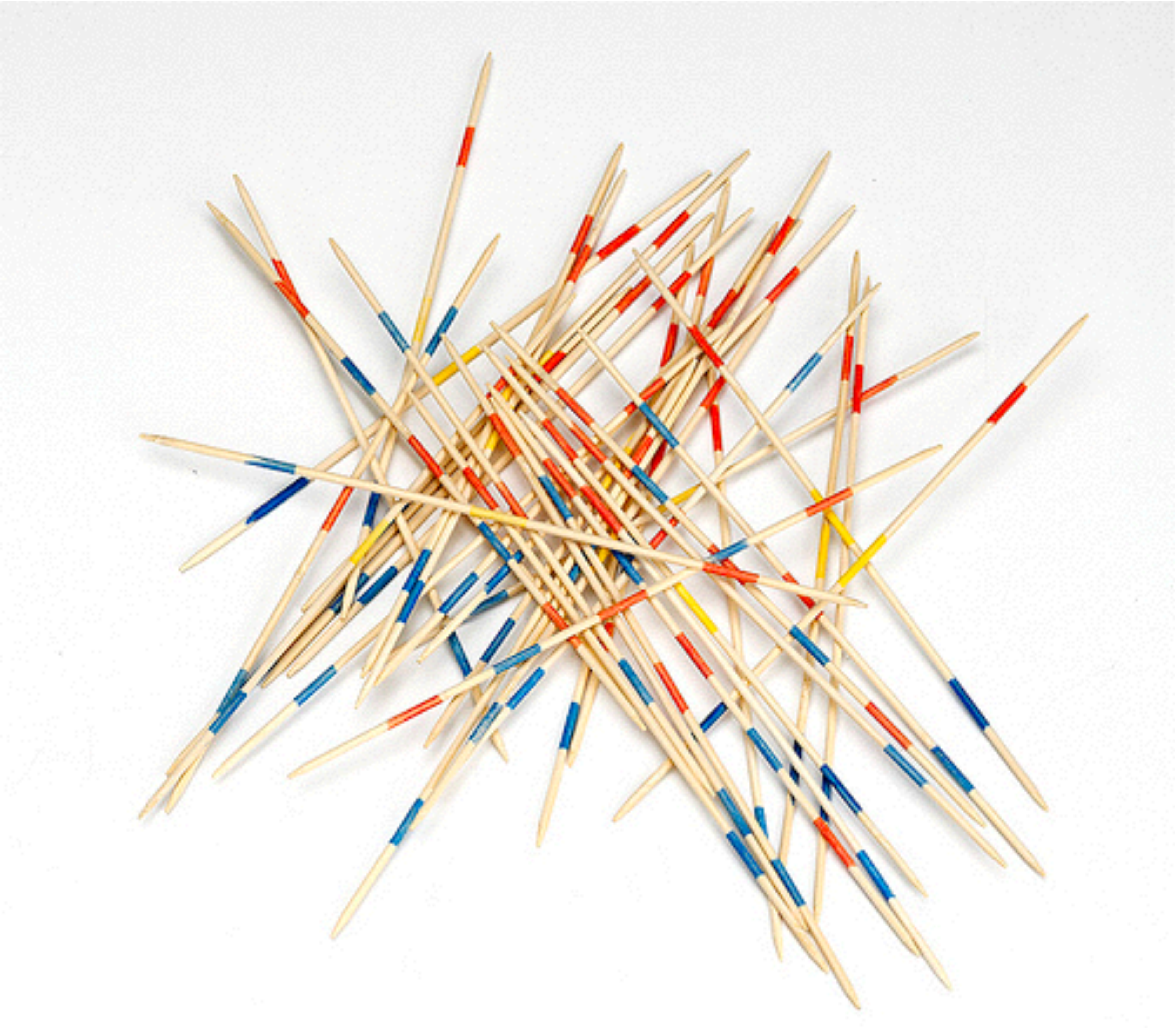
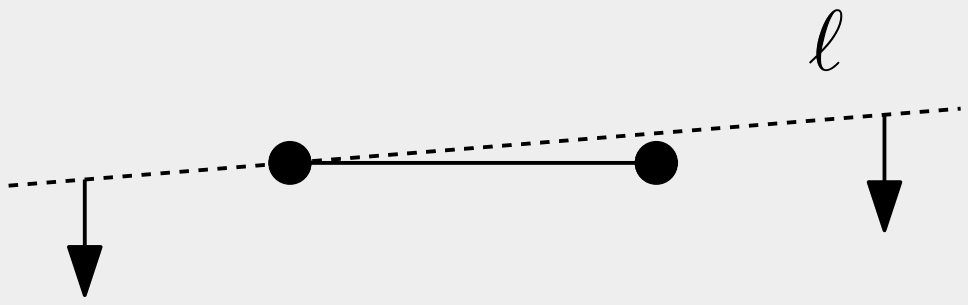


Intersección de Segmentos de Recta (2)

Geometría Computacional , MAT-125

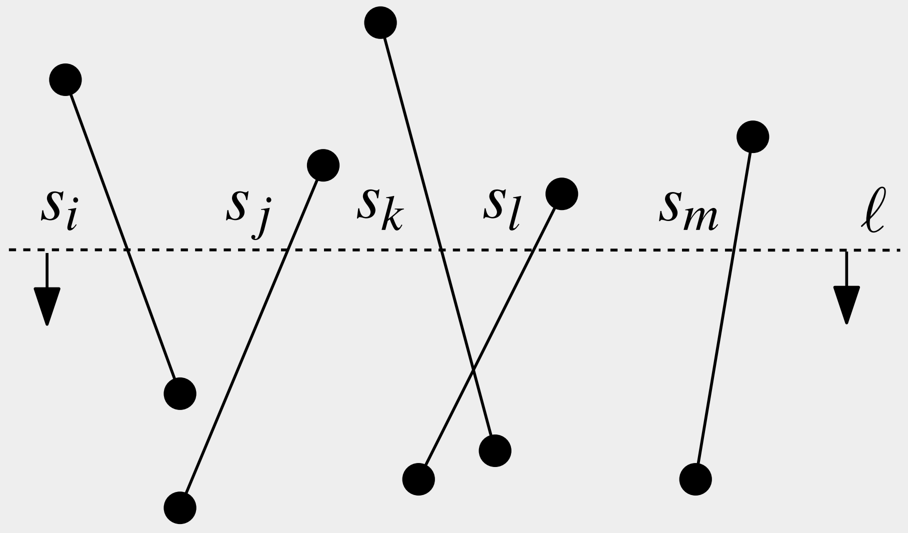
¿Qué estructuras de datos necesitamos para implementar este algoritmo?

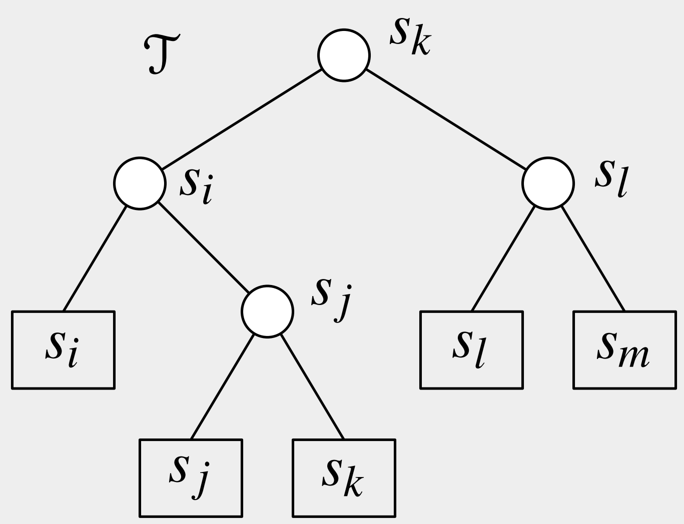
- **cola de eventos Q.**
- **Operaciones**:
  - Eliminar el próximo evento (el más alto abajo de la línea de barrido) en Q y regresar el punto evento.
  - Si dos puntos evento tienen la misma coordenada y, regresar aquel con la coordenada x más pequeña.
  - En una línea horizontal el punto más a la izquierda será el evento superior.



- Insertar un evento.
- Verificar si un segmento está dentro de Q.
- Definir un orden $\prec$ en los puntos evento.
- Si p y q son puntos evento, $p \prec q$ si y solo si $p_y > q_y$ o si $p_y = q_y$, $p_x < q_x$.
- Guardar los puntos evento en un árbol binario balanceado, ordenado de acuerdo a $\prec$
- Con cada punto evento p en Q se deben almacenar también los segmentos que empiecen en p.
- Ambas operaciones toman O(log m) donde m es el número de eventos en Q.
- No se utiliza un montículo porque hay que verificar si un evento ya está presente en Q.

- Se debe mantener un estado del algoritmo,: una secuencia de segmentos ordenados que intersequen la línea de barrido.
- La estructura del estado T, se usa para acceder a los vecinos de un segmento dado s, de tal manera que se pueda probar si intersecta con s.
- La estructura debe ser dinámica ya que los segmentos empiezan o terminan de intersectar a la línea de barrido (se añaden y eliminan).
- Como hay un orden bien definido en los segmentos dentro de la estructura de estado, se puede usar un árbol binario de búsqueda balanceado.
- Los segmentos que intersecan la línea de barrido se encuentran en el mismo orden en las hojas del árbol binario de búsqueda.





- El orden de izquierda a derecha sobre la línea de barrido corresponde al orden de izquierda a derecha de las hojas de T.
- Los nodos internos mantienen la información necesaria para guiar la búsqueda hacia abajo.
- En cada nodo interno, almacenamos el segmento más a la derecha en el subárbol izquierdo.

- Supongamos que buscamos en T al segmento inmediatamente a la izquierda de un punto p sobre la línea de barrido.
- En cada nodo interno v, probamos si p se encuentra a la izquierda o a la derecha del segmento almacenado en v.
- Dependiendo de estas prueba bajamos hacia el subárbol izquierdo o al derecho hasta llegar a una hoja.
- El segmento buscado estará almacenado en esta hoja o en la inmediata izquierda.
- Cada actualización y búsqueda de vecino toma O(log n).
- Las únicas estructuras que necesitamos entonces son:
    - La cola de eventos Q.
    - El estado de la línea de barrido T.

Algorithm FINDINTERSECTIONS(S)

Input. A set S of line segments in the plane.

Output. The set of intersection points among the segments in S, with for each intersection point the segments that contain it.

1. Initialize an empty event queue $\mathcal{Q}$. Next, insert the segment endpoints into $\mathcal{Q}$; when an upper endpoint is inserted, the corresponding segment should be stored with it.
2. Initialize an empty status structure $\mathcal{T}$.
3. **while** $\mathcal{Q}$ is not empty
4.   **do** Determine the next event point p in $\mathcal{Q}$ and delete it.
5.   HANDLEEVENTPOINT(p)

HANDLEEVENTPOINT(p)

Let $U(p)$ be the set of segments whose upper endpoint is p; these segments are stored with the event point p. (For horizontal segments, the upper endpoint is by definition the left endpoint.)

Find all segments stored in $\mathcal{T}$ that contain p; they are adjacent in $\mathcal{T}$. Let $L(p)$ denote the subset of segments found whose lower endpoint is p, and let $C(p)$ denote the subset of segments found that contain p in their interior.

if $L(p) \cup U(p) \cup C(p)$ contains more than one segment

  then Report p as an intersection, together with $L(p)$, $U(p)$, and $C(p)$.

Delete the segments in $L(p) \cup C(p)$ from $\mathcal{T}$.

Insert the segments in $U(p) \cup C(p)$ into $\mathcal{T}$. The order of the segments in $\mathcal{T}$ should correspond to the order in which they are intersected by a sweep line just below p. If there is a horizontal segment, it comes last among all segments containing p.

($*$ Deleting and re-inserting the segments of $C(p)$ reverses their order. $*$)

if $U(p) \cup C(p) = \emptyset$

  then Let s_l and s_r be the left and right neighbors of p in $\mathcal{T}$.

    FINDNEWEVENT(s_l, s_r, p)

  else Let s' be the leftmost segment of $U(p) \cup C(p)$ in $\mathcal{T}$.

    Let s_l be the left neighbor of s' in $\mathcal{T}$.

    FINDNEWEVENT(s_l, s', p)

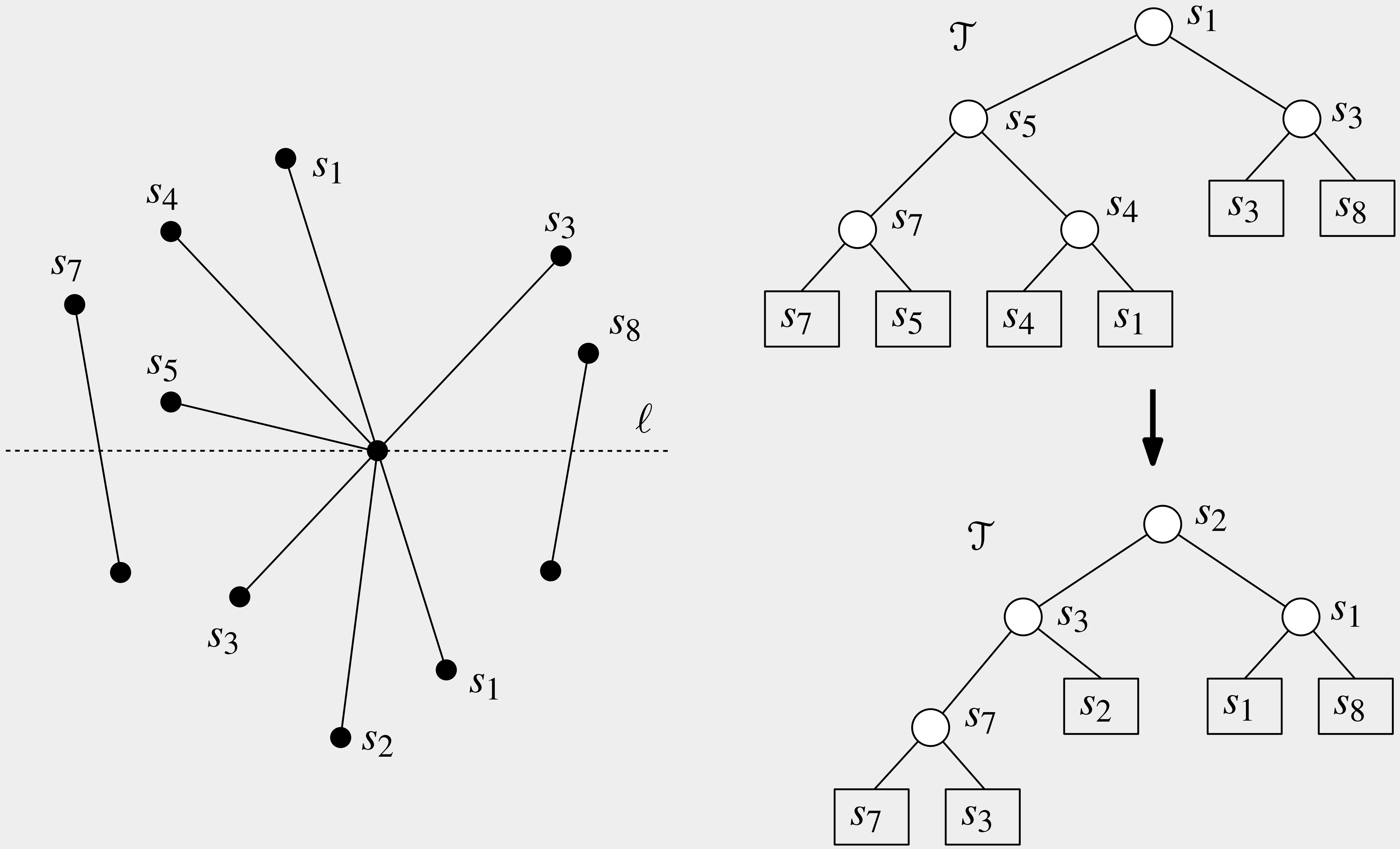
    Let s'' be the rightmost segment of $U(p) \cup C(p)$ in $\mathcal{T}$.

    Let s_r be the right neighbor of s'' in $\mathcal{T}$.

    FINDNEWEVENT(s'', s_r, p)

FINDNEWEVENT(s_l, s_r, p)

1. **if** s_l and s_r intersect below the sweep line, or on it and to the right of the current event point p, and the intersection is not yet present as an event in $\mathcal{Q}$
2. **then** Insert the intersection point as an event into $\mathcal{Q}$.



s_1 s_2 s_3 s_4 s_5 s_7 s_8
ℓ
$\mathcal{T}$
$\mathcal{T}$