**Bài toán giao của các đoạn thẳng**

**(Line Segment Intersection)**

# Bài toán thực tế

Bản đồ chứa đựng nhiều thông tin quý giá cho khách du lịch. Nhưng đôi khi để tìm được đúng thông tin lại gặp nhiều khó khăn (VD khi biết khu vực của 1 địa điểm nhưng lại khó tìm thấy chính xác vị trí địa điểm đó trên bản đồ). Điều này xảy ra bởi vì bản đồ chứa rất nhiều loại thông tin như đường phố, sông, các địa danh, tuyến xe bus, đường sắt,…

Do đó để giúp dễ đọc bản đồ hơn, các hệ thống GIS sẽ chia thông tin lưu theo các layer. Mỗi layer sẽ chứa 1 loại thông tin khác nhau và có cách lưu trữ, biểu diễn dữ liệu khác nhau (phụ thuộc loại thông tin mà nó lưu trữ). Ví dụ: layer về tuyến đường sẽ dùng các đoạn thẳng, đường cong; layer về địa danh thì dùng các điểm; layer về thảm thực vật thì dùng cấu trúc phân chia vùng trên mặt phẳng. Một kịch bản sử dụng là: khách du lịch cần tìm đường đi đến các công viên trong 1 quận, họ sẽ xem layer địa danh, sau khi xác định được 1 vài công viên cần đến, họ sẽ kích hoạt thêm layer tuyến đường để biết chi tiết về các con đường có thể đi đến đích. ***Như vậy bài toán đặt ra là cần overlay (phủ) giữa các layer để hợp thành 1 bản đồ chung.***

# Định nghĩa bài toán

* Xét bài toán dạng đơn giản nhất của map overlay là mỗi bản đồ gồm 1 tập các đoạn thẳng.
* Để đơn giản, ta hợp nhất 2 tập đoạn thẳng lại thành 1 tập. ***Bài toán được định nghĩa như sau***: Cho tập các đoạn thẳng trong mặt phẳng. Cần tìm tất cả giao điểm của mọi cặp đoạn thẳng bất kì. Nếu 1 đoạn thẳng có đầu mút nằm trên đoạn thẳng khác thì cũng tính là giao điểm.

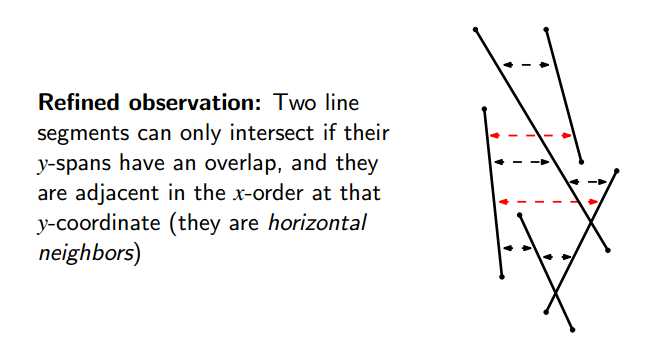
# Một số nhận xét về bài toán

Thuật toán đơn giản có thể giải bài toán như sau: với mọi cặp đoạn thẳng kiểm tra giao điểm giữa chúng. Độ phức tạp là O(n2). Trong nhiều trường hợp, số giao điểm nhỏ hơn nhiều so với O(n2), vậy ta cần phát triển 1 thuật toán ***output sensitive*** – thuật toán mà độ phức tạp phụ thuộc vào kích cỡ của output (trong bài toán này là số giao điểm).

Thuật toán đề xuất phải tránh được việc kiểm tra giao điểm của mọi cặp đoạn thẳng. Điều này đạt được nhờ ý tưởng các đoạn thẳng nằm gần (xa) nhau thì (không) giao nhau và sử dụng quan sát tinh tế sau. Nếu chiếu các đoạn thẳng lên trục Oy thì hai đoạn thẳng không giao nhau khi và chỉ khi hai hình chiếu của chúng không overlap (có điểm chung) nhau. Như vậy ta chỉ cần kiểm tra các cặp đoạn thẳng có hình chiếu trên Oy overlap nhau.

Khi đó phải tồn tại 1 đường thẳng song song Ox mà cùng giao với 2 đoạn thẳng đang xét. Cặp đoạn thẳng này chắc chắn là gần nhau theo phương Oy. Nhưng nếu chỉ hạn chế vậy thì chưa đủ! Khi các đoạn thẳng đều overlap nhau theo Oy nhưng chúng lại xa nhau theo phương Ox, khi đó số giao điểm là rất ít nhưng thuật toán vẫn phải xét rất nhiều cặp đoạn thẳng.

Làm thế nào để chỉ xét các đoạn thẳng gần nhau theo phương Ox? Ta dùng 1 đường thẳng  song song Ox quét từ trên xuống, chỉ quan tâm các cặp đoạn thẳng mà hiện cùng giao với , và chỉ kiểm tra các cặp đoạn thẳng mà có giao điểm với gần nhau.

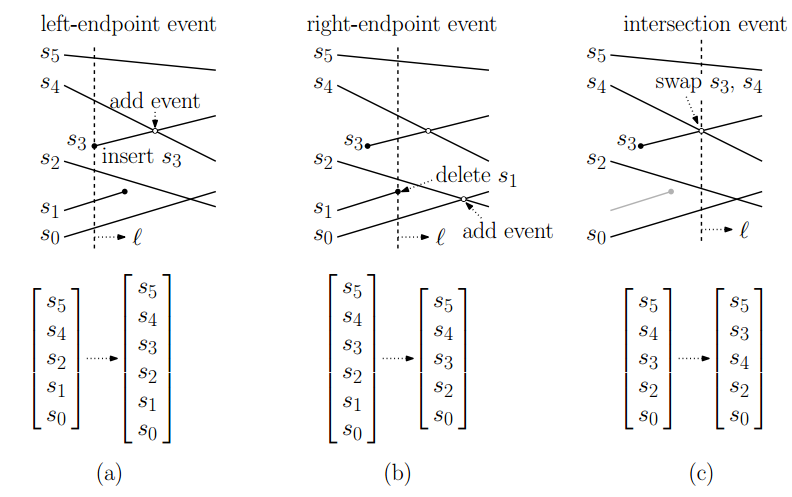


# Thuật toán Sweepline

## Ý tưởng thuật toán

Sử dung 1 đường thẳng sweep line song song Ox quét từ trên xuống. Thuật toán sẽ thực hiện các thao tác khi sweep line gặp một số điểm đặc biệt gọi là ***event point***. Ta cần 1 CTDL T lưu các đoạn thẳng hiện đang giao với sweep line (gần nhau theo phương Oy) và lưu chúng theo thứ tự tọa độ x của giao điểm (để tìm được các đoạn thẳng hàng xóm gần nhau theo Ox). Bất kì khi nào có 2 đoạn thẳng trở thành ‘hàng xóm’ theo Ox thì ta cần kiểm tra giữa chúng có giao điểm hay không?

Tại các điểm event point, danh sách các đoạn thẳng (theo thứ tự tăng dần tọa độ x của giao điểm với sweep line) hiện đang giao sweepline thay đổi. Điều này xảy ra trong 3 trường hợp: khi có 1 đoạn thẳng mới xuất hiện, hoặc 1 đoạn thẳng cũ biến mất, hoặc 2 đoạn thẳng liên tiếp hoán đổi vị trí. Ba sự kiện tương ứng là upper endpoint, lower endpoint, intersection point. Hình sau minh họa sự thay đổi danh sách khi gặp các sự kiện. Hình này lấy từ bài giảng [3], trong đó sử dụng sweep line là đường song song Oy, cũng tương tự với sweep line song song Ox.



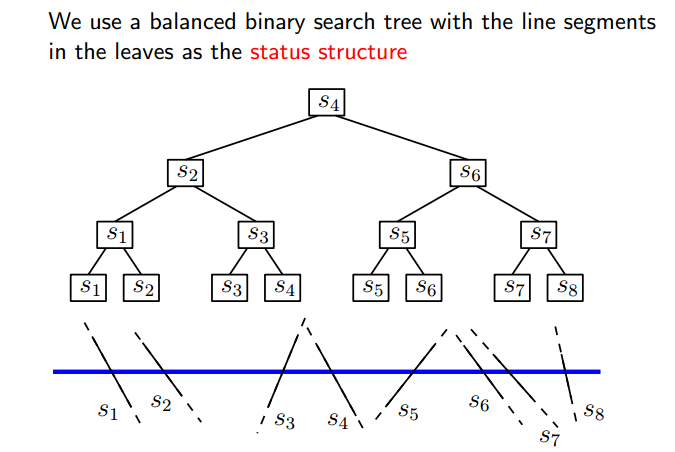
Trong quá trình di chuyển sweep line, mỗi khi có 2 đoạn thẳng trở thành ‘hàng xóm’ thì ta cần phát hiện giữa chúng có giao điểm không? Do đó ta cần 1 CTDL Q để lưu các event point, hỗ trợ thao tác extract\_max theo tọa độ y, insert, delete event point. Do đó ta sẽ dùng priority queue để cài đặt cho Q.

## Cấu trúc dữ liệu

### Cây nhị phân tìm kiếm cân bằng T

Cây BBST T sẽ biểu diễn trạng thái hiện tại.

* Lá sẽ biểu diễn các đoạn thẳng. Các lá từ trái qua phải sẽ có thứ tự giống thứ tự giao điểm của các đoạn thẳng với sweep line theo tọa độ x tăng dần. Ngoài ra có thể có thêm pointer trỏ đến intersection point trong Q để sau này xóa hoặc kiểm tra event đã tồn tại chưa.
* Nút trong sẽ biểu diễn đoạn thẳng bên phải nhất của cây con trái.

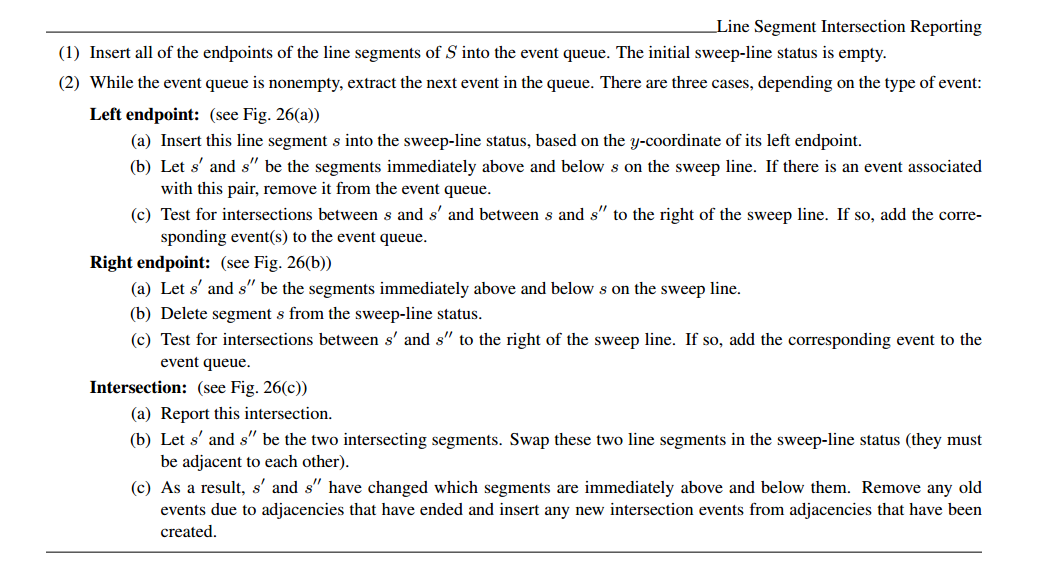


### Hàng đợi ưu tiên Q

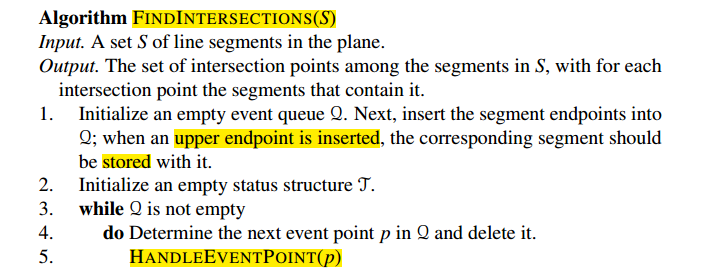
Q lưu tọa độ của 3 loại event point.

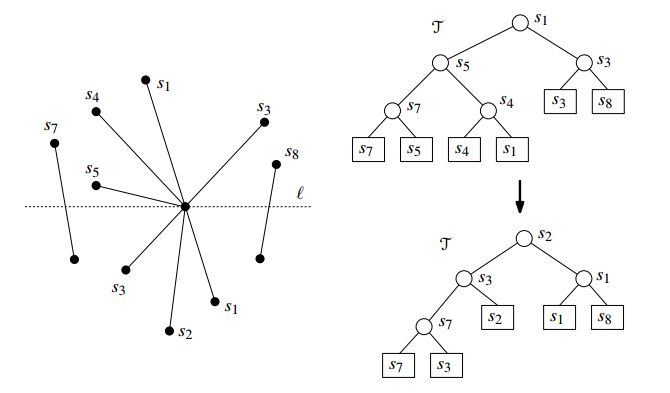
## Nội dung thuật toán

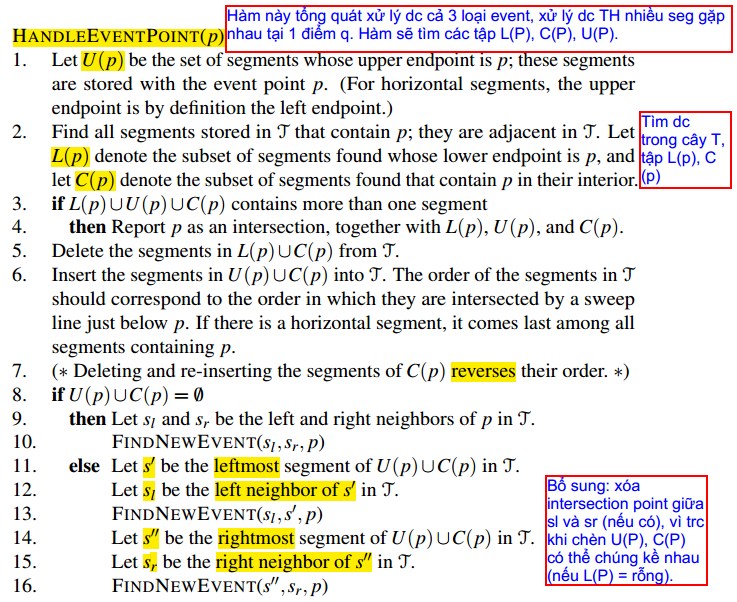
Hình sau viết chi tiết các bước của thuật toán, rất dễ hiểu, sử dụng sweep line là đường song song Oy.



Thuật toán trên chưa xử lý 1 số trường hợp đặc biệt như: có nhiều đoạn thẳng giao nhau tại cùng 1 điểm. Cách cài đặt tiếp theo (sweep line song song Ox) sẽ giải quyết được vấn đề này, nhưng khó hiểu hơn.







## Độ phức tạp

Độ phức tạp về thời gian là O(nlogn + klogn) với k là số giao điểm. Nếu có O(n2) giao điểm thì thuật toán này tồi hơn thuật toán brute-force.

Độ phức tạp về bộ nhớ là O(n) (nếu xóa event của 2 đoạn thẳng không còn kề nhau khỏi Q).

# Kết luận

Đây là bài toán giao của các đoạn thẳng. Ngoài ra còn có các dạng phức tạp hơn như giao của các vùng trong mặt phẳng. Bài toán này được phân tích trong mục 2.3 tài liệu [1].

# Tài liệu tham khảo

1. [*Computational Geometry - Algorithms and Applications*](http://www.cs.uu.nl/geobook/) by de Berg, Cheong, van Kreveld, and Overmars, third edition, 2008
2. <http://www.cs.uu.nl/docs/vakken/ga/slides2a.pdf>
3. <https://www.cs.umd.edu/class/fall2014/cmsc754/lectures.shtml>