UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Geometrie computationala

9. Intersectii. Intersectia segmentelor

Probleme de intersectii (1)

1. **Intersectii de segmente**: fiind date n segmente, sa se gaseasca intersectiile dintre ele in mod eficient.
   - Cel mai nefavorabil caz: $k = n(n-1)/2 = O(n^2)$ intersectii.
   - Algoritm optimal: $O(n \log n + k)$ timp and $O(n)$ spatiu.
2. **Intersectie de poligoane**
   - Intersectia a doua poligoane simple nu este un poligon simplu.
   - Algoritm optimal: $O(n \log n + k)$ timp, $O(n+m)$ pentru poligoane convexe.
3. **Suprapunere de subdiviziuni:** fiind date doua subdiviziuni planare, sa se calculeze suprapunerea lor. Generalizare a punctului 2.

Probleme de intersectii (2)

4. **Intersectii si aranjamente liniare**: Fiind date *n* linii (semispatii), sa se calculeze intersectiile lor si regiunile pe care le definesc.

5. **Intersectii de fete si poliedre**
   - Intersectia a doua fete poligonale sau triunghiulare in spatiu
   - Intersectia a doua (sau mai multe) poliedre.

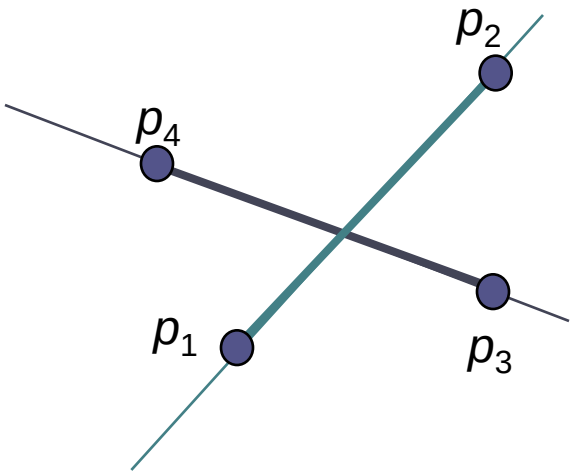
Multe aplicatii:

- Operatii de baza in grafica (ex. clipping)
- Map overlay
- Inginerie si design VLSI
- Domenii non-geometrice: baze de date, paralelizare

Intersectia segmentelor

<u>Teorema</u>: Segmentele (p_1,p_2) si (p_3,p_4) se intersecteaza in interiorul lor daca si numai daca:

- p_1 si p_2 se afla pe parti diferite ale liniei p_3p_4
- p_3 si p_4 se afla pe parti diferite ale liniei p_1p_2



Cazuri speciale:

Gasirea punctului de intersectie

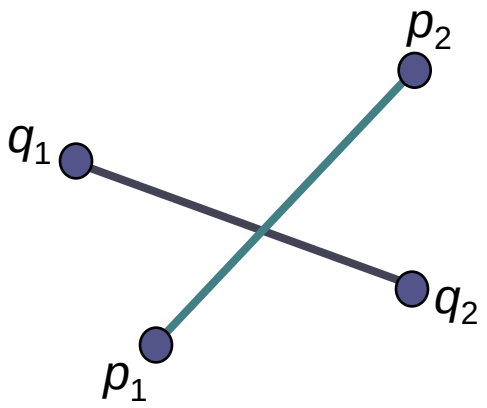
$$p(t) = p_1 + (p_2 - p_1)t \qquad 0 \le t \le 1$$

$$q(s) = q_1 + (q_2 - q_1)s \qquad 0 \le s \le 1$$

Se rezolva ecuatia pentru *t* si *s*:

p(*t*) = q(*s*)

Se verifica $t \in [0,1], s \in [0,1]$



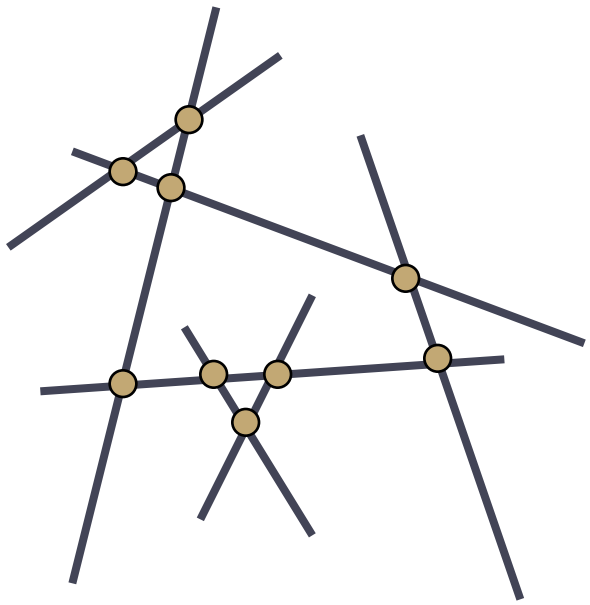
Intersectia segmentelor

Problema: Fiind date n segmente in plan, sa se calculeze toate intersectiile lor.

Alta varianta: Exista o pereche de segmente care se intersecteaza?

Presupunere:

- Nici un segment nu este vertical.
- Nu exista segmente coliniare.
- Nu exista trei segmente ce se intersecteaza intr-un punct comun.



Nr segmente: n
Nr intersectii: k
$0 \le k \le n(n-1)/2$

Algoritm naiv: Se verifica fiecare pereche de segmente pentru a gasi o intersectie. Complexitate de timp: $\Theta(n^2)$.

Intersectia segmentelor

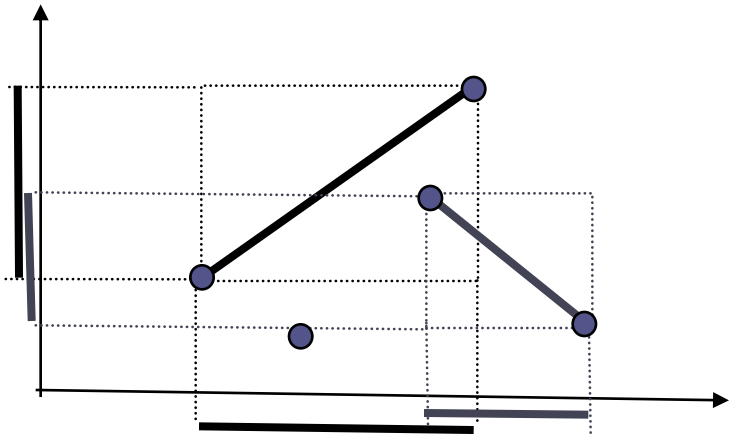
Scop:

Algoritm sensibil la iesire

- $O(n \log n + k \log n)$ timp
- $O(n)$ spatiu
- Nu este optimal, dar bun pentru inceput

Idee:

- Segmentele apropiate sunt candidate pentru intersectie.
- Se urmaresc schimbarile in distanta dintre segmente..



Daca proiectiile pe x si pe y nu se intersecteaza, nu avem intersectii de segmente.

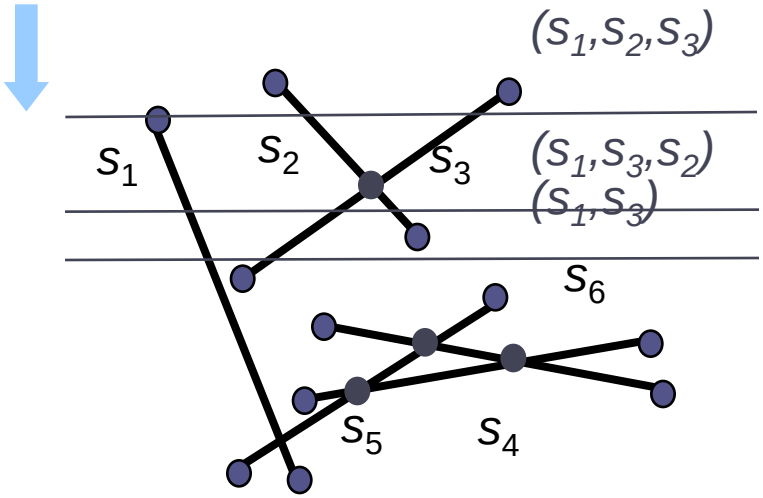
Algoritmul “Linie de cautare” (*Line Sweep*)

Adiacentele segmentelor in raport cu o linie orizontala (sau verticala) se schimba local:

- s_i si s_j trebuie sa fie adiacente pentru a aparea o intersectie.
- (s_i,s_j) inainte de intersectie.
- (s_j,s_i) dupa intersectie.

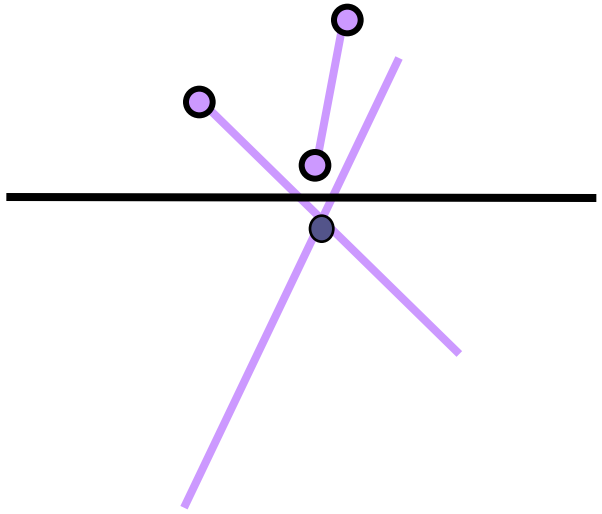
Se monitorizeaza adiacentele dintre segmente pe masura ce linia orizontala se misca de sus in jos.

Procesul se opreste doar asupra *punctelor eveniment*!



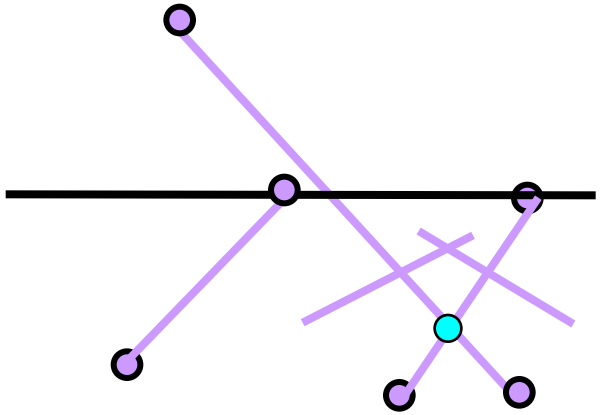
Teorema de adiacenta

Teorema: Inainte ca o intersectie sa apara (la o apropiere infinitesimala de aceasta), cele doua segmente sunt adiacente din punctul de vedere al liniei de cautare.



Demonstratie:

In practica: Privim inainte: oricand doua segmente devin adiacente de-a lungul liniei de cautare, vom cauta intersectia acestora sub linia de cautare.

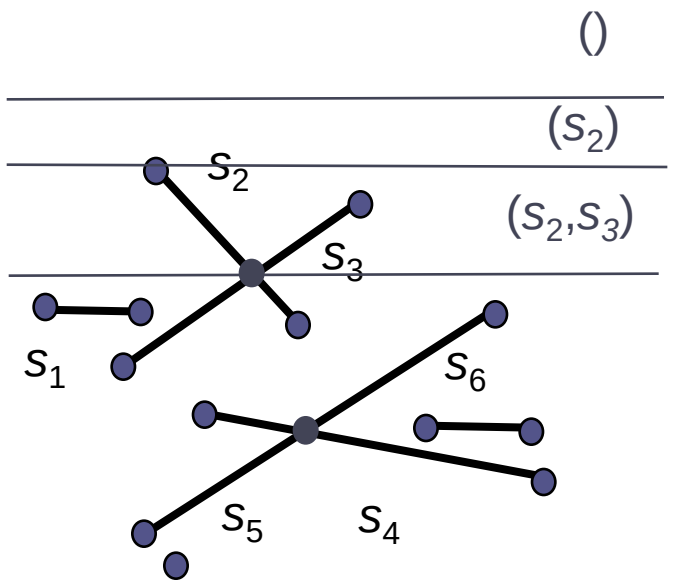


Sweep Line (1)

- Un **eveniment** este oricare punct final de segment sau punct de intersectie.
- Se translateaza o linie orizontala de cautare de sus in jos.

Se pastreaza doua structuri de date:

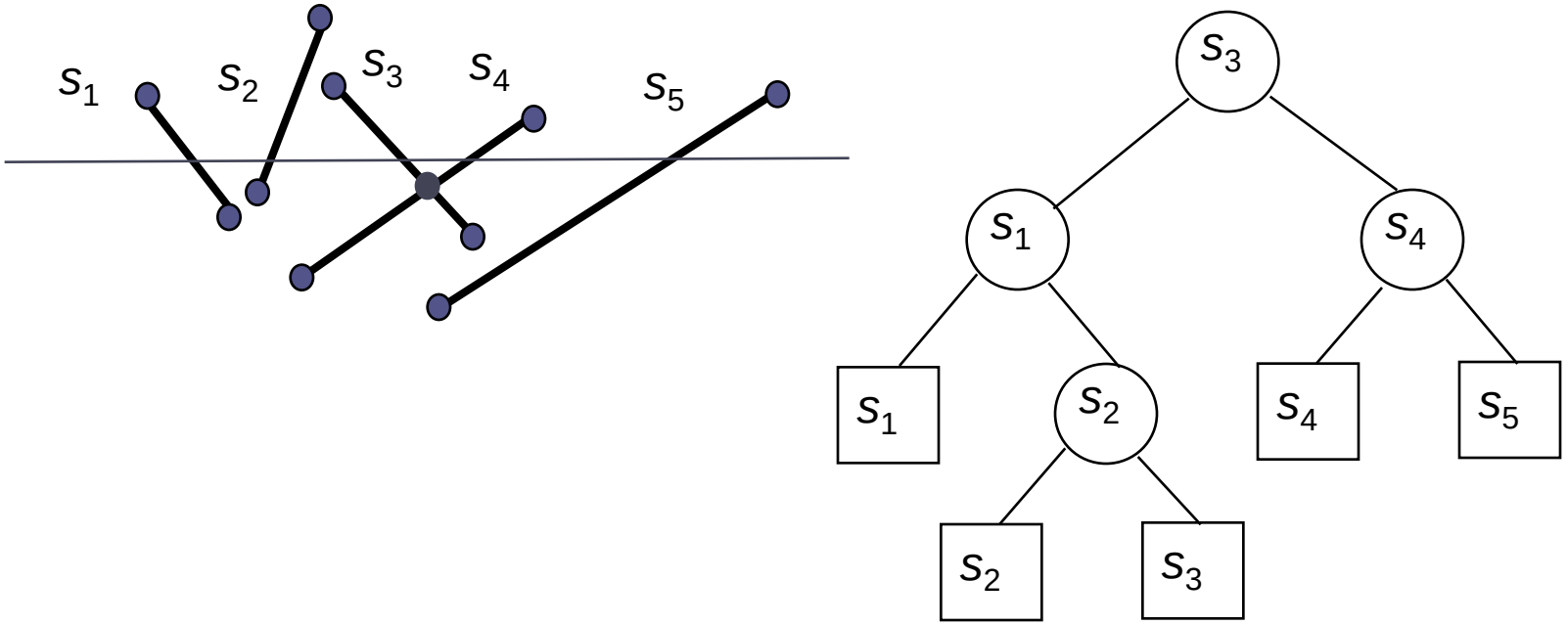
- Un priority queue de evenimente Q, sortat dupa coordonata y.
- Starea liniei de cautare L: se pastreaza segmentele ce sunt intersectate curent de linia de cautare, sortate dupa coordonata y.



Nr evenimente ≤ $n + k$
Nr segmente in $L \leq n$

Arbore binar echilibrat

- Arbore binar echilibrat, sortat dupa coordonata x.
- Inserarea, stergerea, si operatiile in vecinatate in $O(\log n)$

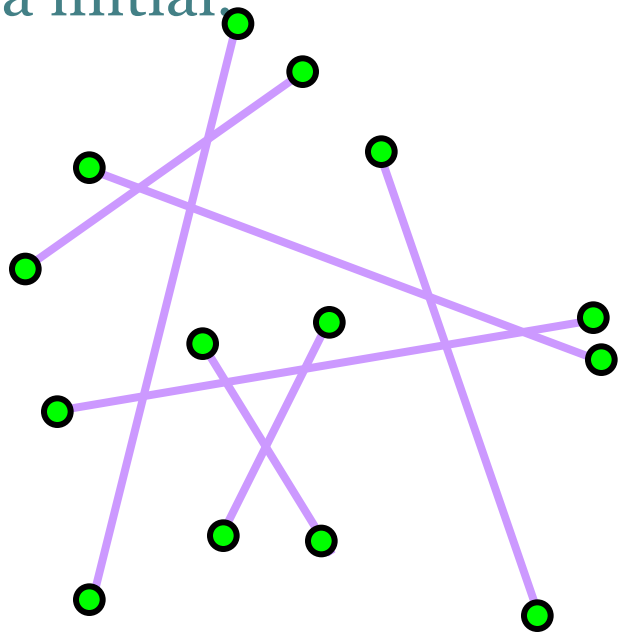


Sweep Line (2)

Initializare:

- Se pun toate capetele de segmente in coada de evenimente Q, sortate dupa coordatele y.
  Timp: $O(n \log n)$.
- Starea liniei de cautare este goala initial.

Algoritmul continua prin inserarea, stergerea si tratarea evenimentelor discrete din coada Q pana cand aceasta devine goala. Totodata, algoritmul mentine lista L.

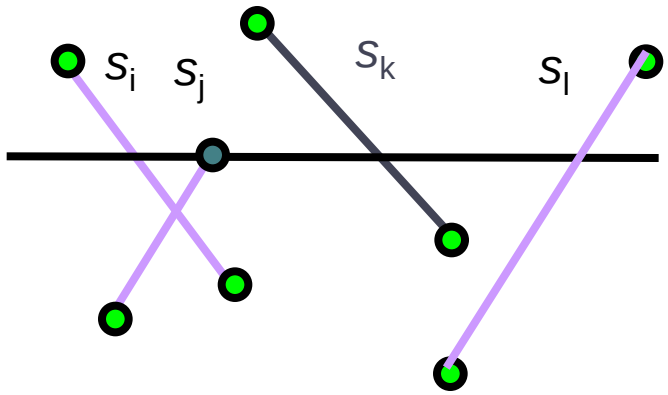


Tratarea evenimentelor: inceputul segmentului

Eveniment de tipul A: inceputul segmentului (capat superior)

- Se localizeaza pozitia segmentului.
- Se insereaza segmentul in starea liniei de cautare.
- Se verifica intersectia sub linia de cautare, cu segmente imediat la stanga sau la dreapta segmentului in cauza. Se adauga punctele de intersectie (daca exista si daca nu au fost deja adaugate) in coada de evenimente.

Complexitate: n evenimente, $O(\log n)$ timp pentru fiecare $\rightarrow$ $O(n \log n)$ total.

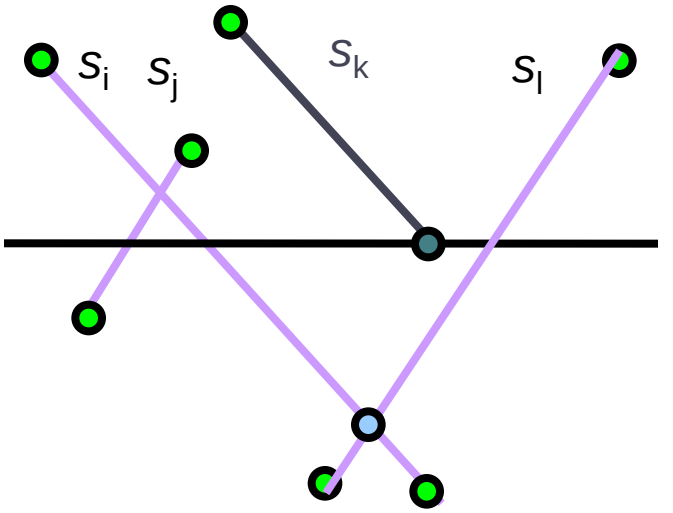


Tratarea evenimentelor: sfarsitul segmentului

Eveniment de tipul B: sfarsitul segmentului (capat inferior)

- Se localizeaza pozitia segmentului.
- Se sterge segmentul din starea liniei de cautare.
- Se verifica intersectia sub linia de cautare, cu segmente imediat la stanga sau la dreapta segmentului in cauza. Se adauga punctul de intersectie (daca exista si daca nu a fost deja adaugat) in coada de evenimente.

Complexitate: n evenimente, $O(\log n)$ timp pentru fiecare $\rightarrow$ $O(n \log n)$ total.

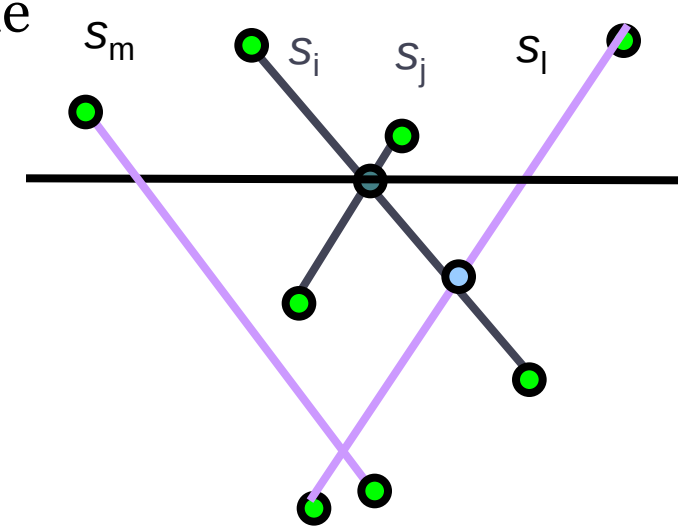


Tratarea evenimentelor: intersectie

Eveniment de tipul C: punct de intersectie

- Se raporteaza punctul.
- Se interschimba respectivele doua segmente in starea liniei de cautare.
- Pentru noile segmente la stanga si la dreapta – se verifica intersectia cu segmentul imediat adiacent la linia de cautare (daca exista). Se insereaza punctul de intersectie (daca exista, si daca nu a fost deja inserat) in coada de evenimente.

Complexitate: k evenimente, O(log n) fiecare→ O(k log n) in total.



k este dimensiunea la iesire.

Analiza complexitatii

- Structuri de date:
  - Coada de evenimente: heap
  - Starea liniei de cautare: arbore binar echilibrat
- Orice operatie pe heap/arbore necesita $O(\log n)$ timp.

- Complexitate totala de timp: $O((n+k) \log n)$.
  - Daca $k \approx n^2$ algoritmul se comporta mai slab decat cel naiv.
  - Cand $k = o(n^2/\log n)$ algoritmul sweep line este mai rapid.
- Complexitate spatiala totala: $O(n+k)$.

Algoritmi de intersectie a segmentelor

- Naiv $O(n^2)$
- Bentley-Ottman, 1988: $O((n+k) \log n)$
- Edelsbrunner-Chazelle, 1990: $O(n \log n + k)$ necesita $O(n \log n)$ spatiu, complex.
- Clarkson-Shor, Mumuley, 1992: $O(n \log n + k)$ cu spatiu $O(n)$
- Balaban, 1995: $O(n \log n + k)$ cu spatiu $O(n)$, optimal determinist.