Probleme de căutare și localizare

Mihai-Sorin Stupariu

Sem. I, 2019-2020

Căutare ortogonală — motivație

Exemplu.

Baza de date a unei bănci: informații numerice referitoare la clienți: data nașterii, număr de copii, venitul lunar, valoarea depozitelor, valoarea ratelor de plată, valoarea comisioanelor plătite anual, etc. $\rightarrow$ stocarea se realizează folosind puncte dintr-un spațiu numeric d-dimensional $\mathbb{R}^d$.

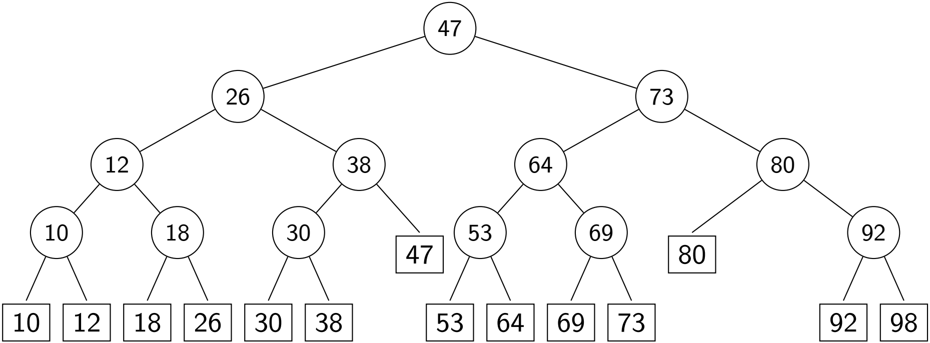
A identifica un "grup-țintă" de clienți (de exemplu pentru lansarea unui produs), având anumite caracteristici — e.g. vârsta între 30-40 ani, 2-4 copii, un venit lunar între 3000-5000 lei, etc. revine la efectuarea căutări prin care să fie determinate punctele situate într-un "paralelipiped" d-dimensional.

Căutare 1-dimensională: formularea problemei

Cadru. Fie $M = \{a_1, a_2, \ldots, a_n\}$ o mulțime de numere reale. Fie $I = [x, x'] \subset \mathbb{R}$ un interval real. Se dorește determinarea elementelor lui M situate în intervalul I.

Structura de date utilizată: Arbore binar de căutare echilibrat.

Exemplu de arbore $\mathcal{T}$



Rezultatul principal - căutare 1D

Teoremă. *Fie M o mulțime de n puncte din $\mathbb{R}$. Mulțimea M poate fi memorată într-un arbore binar de căutare echilibrat, folosind $O(n)$ memorie și cu timp de construcție $O(n \log n)$. Determinarea unor puncte dintr-un interval I poate fi realizată cu complexitate-timp $O(k + \log n)$, unde k este numărul de puncte din $M \cap I$.*

Rezultatul principal - căutare 2D

Teoremă. *Fie M o mulțime de n puncte din planul $\mathbb{R}^2$. Un arbore de intervale (range tree) pentru M necesită $O(n \log n)$ memorie și poate fi construit în timp $O(n \log n)$. Determinarea unor puncte dintr-un dreptunghi D poate fi realizată cu complexitate-timp $O(k + \log^2 n)$, unde k este numărul de puncte din $M \cap D$.*

Localizarea punctelor – problematizare

- Căutare cu Google Maps
- *Interogare pentru localizarea unui punct:* dată o hartă și un punct p, indicat prin cordonatele sale, să se determine regiunea hărții în care este situat p.
- *Harta:* subdiviziune planară, formată din vârfuri, (semi)muchii, fețe.
- Necesități: pre-procesare a informației; interogare rapidă.

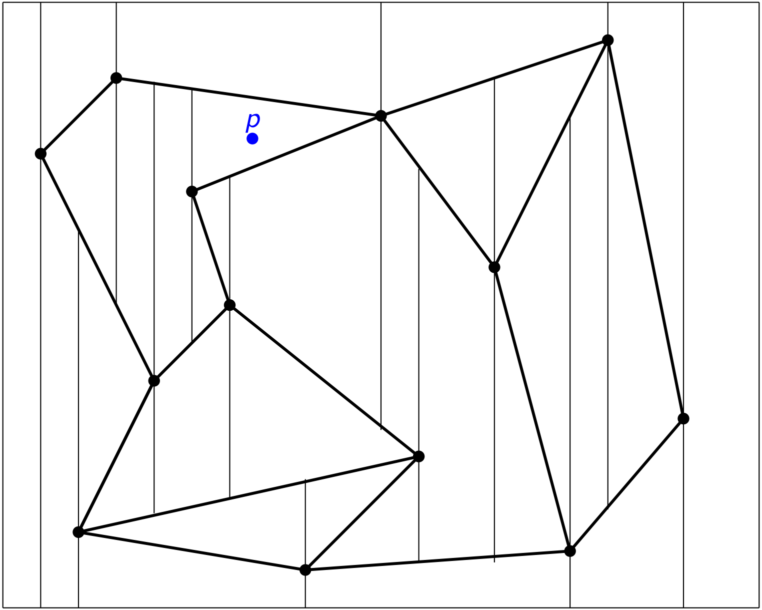
Formalizare

- Fie $\mathscr{S}$ o subdiviziune planară cu n muchii. *Problema localizării unui punct* revine la
  - a reține informațiile referitoare la $\mathscr{S}$ pentru a putea răspunde la interogări de tipul:
  - dat un punct p, se raportează fața f care îl conține pe p; în cazul în care p este situat pe un segment sau coincide cu un vârf, este precizat acest lucru.
- Lucrul cu coordonate: folosirea relației de ordine!

Simplificări și ipoteze

- Se consideră o mulțime S de n segmente astfel ca oricare două dintre ele (i) fie nu au niciun punct comun; (ii) fie au un vârf comun.
- *Simplificare 1:* Se consideră un dreptunghi D cu laturile paralele cu axele de coordonate care include toată subdiviziunea inițială.
- *Simplificare 2:* Se presupune că nu există două vârfuri (extremități ale segmentelor din S) distincte care au aceeași coordonată x (în particular nu există segmente verticale).
- *Concluzie:* Se consideră o mulțime de n segmente S care verifică ipotezele de mai sus: *mulțime de segmente în poziție generală*. **Harta trapezoidală / descompunere verticală / descompunere cu trapeze** (*trapezoidal map*) $\mathscr{T}(S)$ a lui S este subdiviziunea indusă de S, dreptunghiul D și de extensiile verticale inferioare și superioare (concept introdus de Seidel, 1991).

Exemplu - hartă trapezoidală



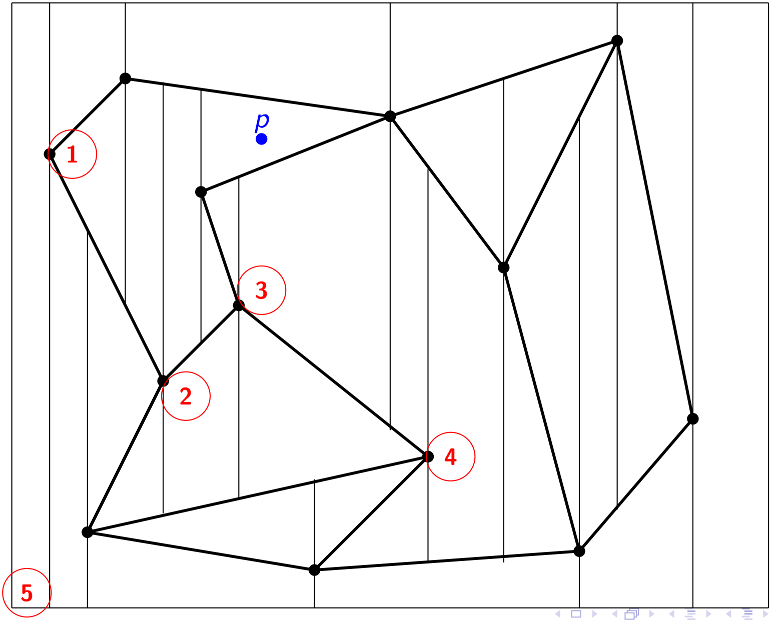
Hărți trapezoidale – probleme studiate

- Descrierea obiectelor geometrice din care sunt formate – ce informații se rețin?
- Aspecte legate de complexitate?
- Structuri de date adecvate?
- Un algoritm eficient?

Descrierea obiectelor

- **Lema 1.** Fie S o mulțime de segmente în poziție generală. Fiecare față a unei hărți trapezoidale $\mathscr{T}(S)$ are una sau două margini verticale și exact două margini ne-verticale.
  De fapt: fiecare față este un trapez, sau un dreptunghi sau un triunghi (ultimele putând fi privite drept cazuri particulare de trapeze).
- Ce informații geometrice sunt reținute pentru un trapez?
- $\mathtt{t}(T)$, $\mathtt{b}(T)$, $\mathtt{lp}(T)$, $\mathtt{rp}(T)$ determină în mod unic un trapez fixat T.
  $\mathtt{t}(T)$, $\mathtt{b}(T)$ sunt **segmente**, iar $\mathtt{lp}(T)$, $\mathtt{rp}(T)$ sunt **vârfuri** (extremități ale segmentelor)
- Există cinci cazuri posibile pentru marginea stângă $\mathtt{lp}$ (analog pentru marginea dreaptă $\mathtt{lp}$).

Exemplu - hartă trapezoidală



Complexitate și alte aspecte cantitative

- **Lema 2.** Fie S o mulțime de n segmente în poziție generală. Harta trapezoidală $\mathscr{T}(S)$ conține cel mult $6n + 4$ vârfuri și cel mult $3n + 1$ trapeze.
- **Lema 3.** Fie S o mulțime de n segmente în poziție generală. Fiecare trapez T este adiacent cu cel mult patru trapeze (cel mult un vecin stânga superior, cel mult un vecin stânga inferior, cel mult un vecin dreapta superior, cel mult un vecin dreapta inferior).

Structura de date

- Înregistrări pentru segmentele din S și vârfuri (extremitățile segmentelor).
- Înregistrări pentru trapeze: pointeri `t`, `b`, `lp`, `rp` și pointeri către cei (cel mult) patru vecini.
- **Structura de căutare:** $\mathscr{D}$ este un graf orientat aciclic cu o singură rădăcină și exact o frunză pentru fiecare trapez din $\mathscr{T}(S)$.
- **Noduri și teste asociate:**
  - x-nod, etichetat cu o extremitate a unui segment; pentru un punct p testul asociat: *este punctul p situat la stânga sau la dreapta dreptei verticale care trece prin extremitatea memorată în acest nod?*
  - y-nod, etichetat cu un segment; pentru un punct p testul asociat: *este punctul p situat deasupra sau dedesubtul segmentului memorat în acest nod?*

Algoritm HARTATRAPEZOIDALA

- **Input.** O mulțime S de n segmente în poziție generală.
- **Output.** Harta trapezoidală $\mathcal{T}(S)$ și o structură de căutare $\mathcal{D} = \mathcal{D}(\mathcal{T}(S))$ pentru $\mathcal{T}(S)$, într-un dreptunghi D cu laturile paralele cu axele.

1. Determină dreptunghiul D.
2. Generează o permutare $s_1, s_2, \ldots, s_n$ a segmentelor din S.
3. **for** $i \leftarrow 1$ **to** n
4. **do** găsește mulțimea de trapeze $T_0, T_1, \ldots, T_k$ care intersectează segmentul s_i
5. elimină $T_0, \ldots, T_k$ și le înlocuiește cu trapezele nou apărute
6. elimină frunzele corespunzătoare din $\mathcal{D}$ și creează noi frunze, actualizează $\mathcal{D}$

Rezultatul principal

- **Teoremă.** *Fie S o mulțime de n segmente în poziție generală. Algoritmul* HARTATRAPEZOIDALA *determină harta trapezoidală $\mathscr{T}(S)$ și o structură de căutare $\mathscr{D} = \mathscr{D}(\mathscr{T}(S))$ pentru $\mathscr{T}(S)$ în timp mediu $O(n \log n)$. Memoria medie ocupată de structura de căutare este $O(n)$ și pentru un punct arbitrar p timpul mediu de localizare este $O(\log n)$.*

Algoritm DETERMINASPATIULIBER (S)

- **Input.** O mulțime $\mathcal{P}$ de poligoane disjuncte.
- **Output.** O hartă trapezoidală $\mathcal{C}_l$ a spațiului liber (pentru un robot-punct).

1. Fie S mulțimea muchiilor poligoanelor din $\mathcal{P}$.
2. Determină harta trapezoidală $\mathcal{T}(S)$, folosind algoritmul HARTATRAPEZOIDALA.
3. Elimină trapezele situate în interiorul poligoanelor și returnează subdiviziunea obținuta.

Algoritm DETERMINADRUM $(\mathcal{T}(\mathcal{C}_l), \mathcal{G}_d, M_{\text{start}}, M_{\text{end}})$

- **Input.** Harta trapezoidală $\mathcal{T}(\mathcal{C}_l)$ a spațiului liber, graful drumurilor $\mathcal{G}_d$, punctul de start M_{start}, punctul final M_{end}.
- **Output.** Un drum de la M_{start} la M_{end}, dacă există. În caz contrar, algoritmul precizează că nu există un drum.

caută trapeze în $\mathcal{T}(\mathcal{C}_l)$ conținând M_{start}, respectiv M_{end}.

if există Δ_{start}, respectiv Δ_{end} conținând M_{start}, respectiv M_{end}

    then fie v_{start} și v_{end} centrele Δ_{start}, Δ_{end} (noduri din $\mathcal{G}_d$)

        caută un drum în $\mathcal{G}_d$ de la v_{start} la v_{end} folosind BFS

        if există drum δ

            then indică drumul $[M_{\text{start}} v_{\text{start}}] \cup \delta \cup [v_{\text{end}} M_{\text{end}}]$

            else raportează că nu există drum de la M_{start} la M_{end}

    else raportează că nu există drum de la M_{start} la M_{end}

Rezultatul principal

- **Teoremă.** *Fie* $\mathcal{R}$ *un robot-punct care se deplasează într-o mulțime* S *de obstacole poligonale, având în total n muchii. Utilizând timp mediu de preprocesare* $O(n \log n)$ *pentru mulțimea* S*, un drum liber de coliziuni între două puncte fixate poate fi calculat (dacă există!) în timp mediu* $O(n)$*.*