UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

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

Geometrie computationala

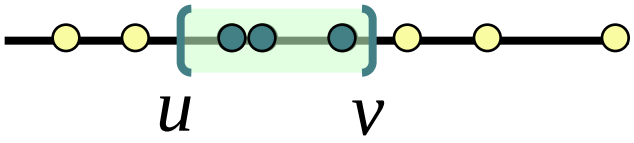
22. Cautari in spatii ortogonale: Cautari 1D si 2D

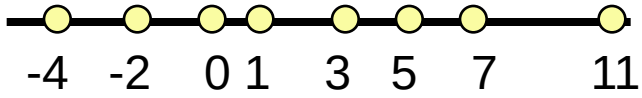
Cautari in 1D (1)

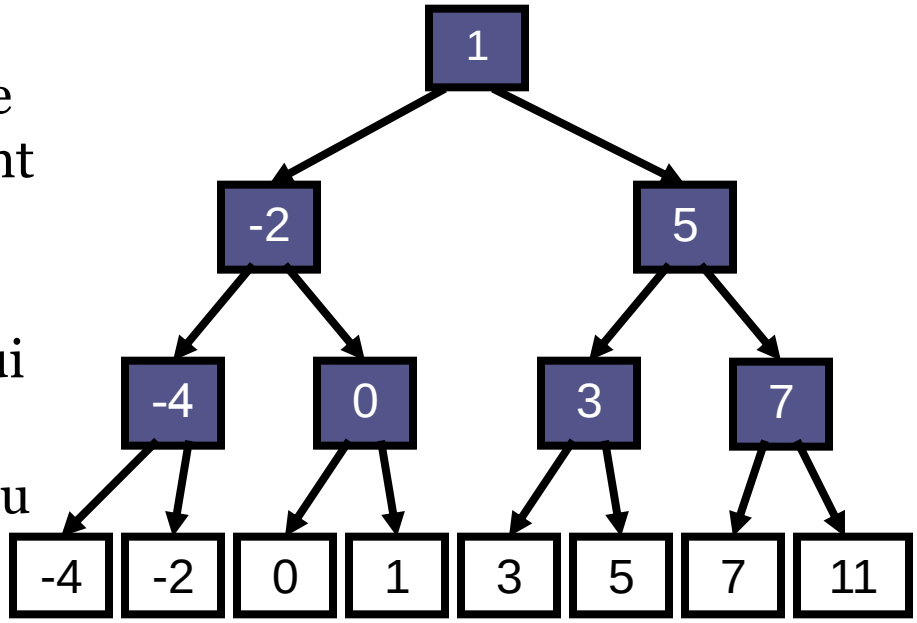
Punctele sunt numere reale, intervalul de cautare este definit de doua numere u si v.

Algoritm:

- Se sorteaza punctele in timp $O(n \lg n)$.
- Se stocheaza punctele intr-un arbore binar echilibrat alea carui frunze sunt punctele.
- Fiecare nod al arborelui stocheaza cea mai **mare** valoare a subarborelui sau **stang**.
- Se efectueaza o cautare binara pentru u si v in lista, in timpul $O(\lg n)$.
- Se listeaza toate valorile din intervalul de cautare.







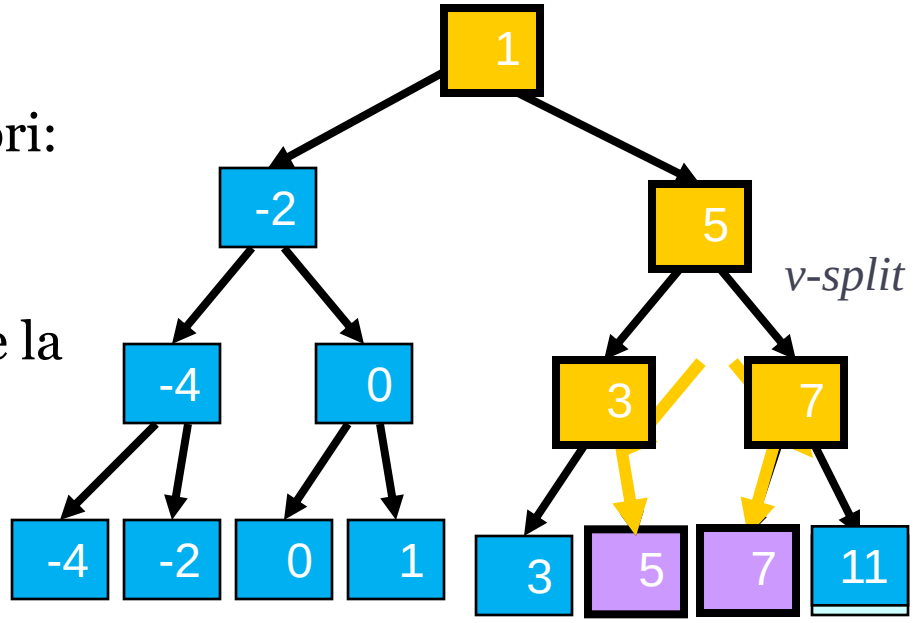
Complexitatea cautarii: $O(\lg n + k)$

Cautari in 1D (2)

- Se gasesc cele doua limite ale intervalului de cautare in frunzele *u* si *v*.
- Se raporteaza toate frunzele in *subarbori maximali* intre *u* si *v*.
- Se marcheaza punctul (*v-split*) la care traseele de cautare *v* se despart.
- Se continua continuarea celor doua limite, raportand valorile in subarbori:

Cand se ajunge la capatul stang (sau drept) al intervalului: Daca se merge la stanga (sau dreapta) se raporteaza intregul subarbore drept (respectiv stang). Cand se atinge o frunza, se verifica valoarea ei.

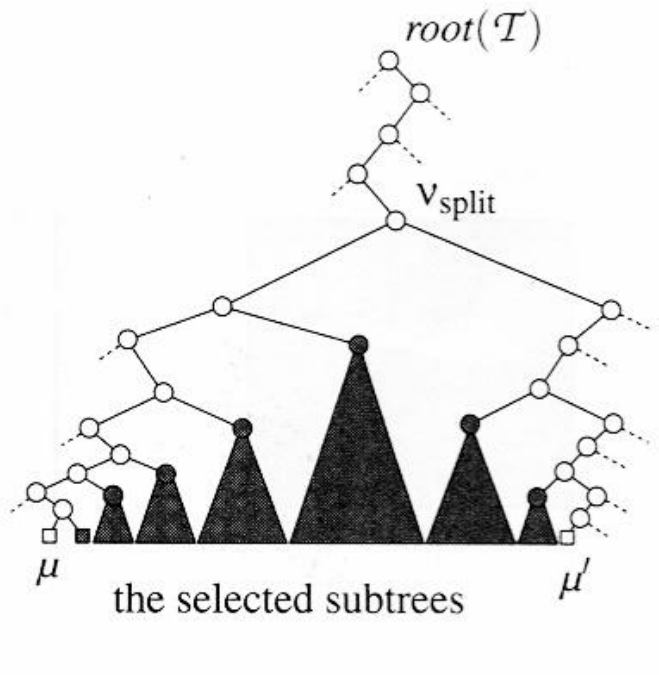
Interval la intrare: 3.5-8.2



Cautari in 1D (3)

FINDSPLITNODE($\mathcal{T}, x, x'$)
Input. A tree $\mathcal{T}$ and two values x and x' with $x \leqslant x'$.
Output. The node ν where the paths to x and x' split, or the leaf where both paths end.

1. $\nu \leftarrow root(\mathcal{T})$
2. **while** ν is not a leaf **and** ($x' \leqslant x_\nu$ **or** $x > x_\nu$)
3. **do if** $x' \leqslant x_\nu$
4. **then** $\nu \leftarrow lc(\nu)$
5. **else** $\nu \leftarrow rc(\nu)$
6. **return** ν



Cautari in 1D (4)

Algorithm 1DRANGEQUERY($\mathcal{T}$, $[x : x']$)
Input. A range tree $\mathcal{T}$ and a range $[x : x']$.
Output. All points that lie in the range.

1. $v_{split} \leftarrow$ FINDSPLITNODE($\mathcal{T}, x, x'$)
2. **if** v_{split} is a leaf
3. **then** Check if the point stored at v_{split} must be reported.
4. **else** (* Follow the path to x and report the points in subtrees right of the path. *)
5. $v \leftarrow lc(v_{split})$
6. **while** v is not a leaf
7. **do if** $x \leqslant x_v$
8. **then** REPORTSUBTREE($rc(v)$)
9. $v \leftarrow lc(v)$
10. **else** $v \leftarrow rc(v)$
11. Check if the point stored at the leaf v must be reported.
12. Similarly, follow the path to x', report the points in subtrees left of the path, and check if the point stored at the leaf where the path ends must be reported.

Cautari in 1D: analiza la runtime

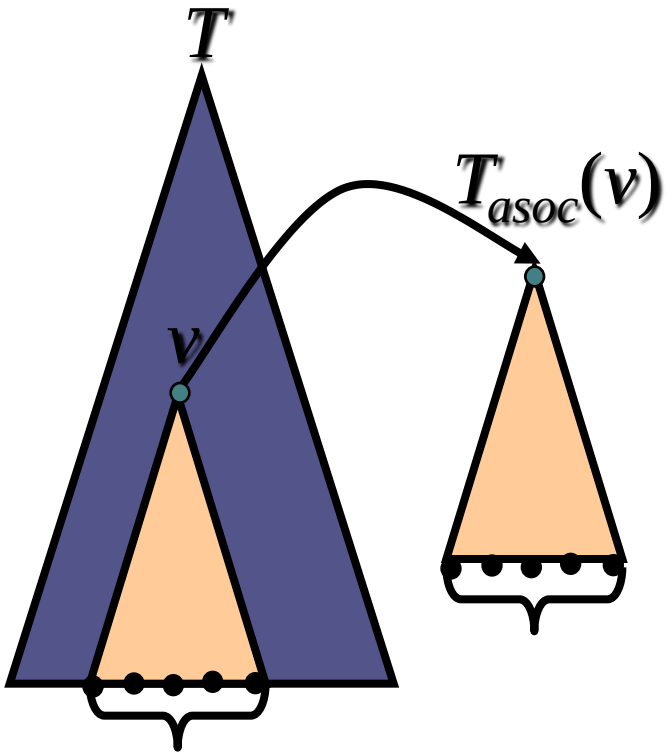
- k: dimensiunea de iesire
- Frunze: $O(k)$ timp
- Noduri interne: $O(k)$ timp (arbore binar)
- Trasee: $O(\lg n)$ timp
- Total: $O(\lg n + k)$ timp
- Cel mai nefavorabil caz: $k = n \rightarrow \Theta(n)$ timp

Cautari in 2D (1)

Se generalizeaza cautarea in 1D

Constructie:

- Se construieste un arbore ordonat dupa coordonatele x.
- Fiecare punct interior v contine un pointer la un arbore secundar, ce contine toate punctele subarborelui primar, ordonate dupa coordonata y.
- Punctele sunt stocate doar in subarborii secundari.



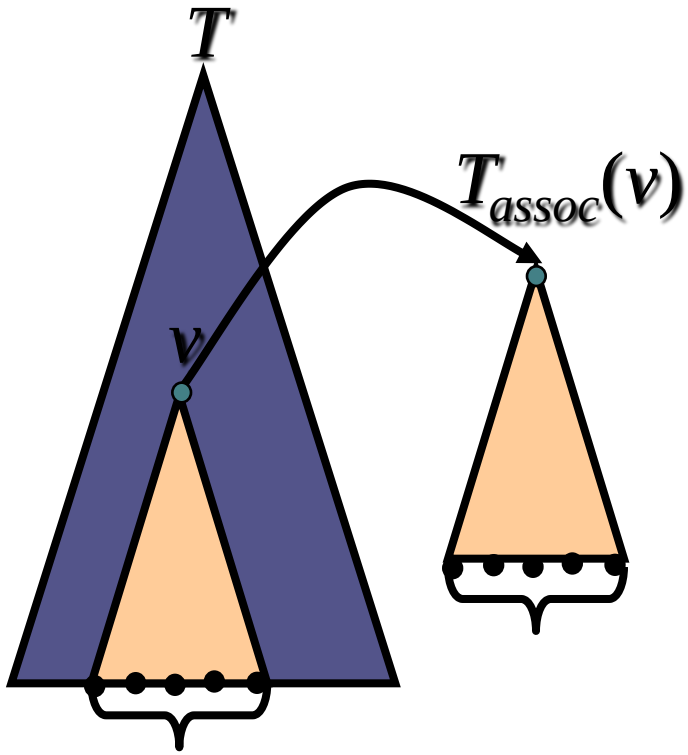
sortat dupa x *sortat dupa y*

Cautari in 2D (2)

Cautare:

- Fiind dat un interval 2D, simulam o cautare 1D si gasim subarborii sortati dupa x.

- In loc sa raportam intregi subarbori, invocam o cautare in arborii secundari sortati dupa y, si raporam doar punctele din intervalul de interogare.



Cautari in 2D (3)

Algorithm BUILD2DRANGETREE(P)
Input. A set P of points in the plane.
Output. The root of a 2-dimensional range tree.

1. Construct the associated structure: Build a binary search tree $\mathcal{T}_{\text{assoc}}$ on the set P_y of y-coordinates of the points in P. Store at the leaves of $\mathcal{T}_{\text{assoc}}$ not just the y-coordinate of the points in P_y, but the points themselves.
2. **if** P contains only one point
3. **then** Create a leaf v storing this point, and make $\mathcal{T}_{\text{assoc}}$ the associated structure of v.
4. **else** Split P into two subsets; one subset P_{left} contains the points with x-coordinate less than or equal to x_{mid}, the median x-coordinate, and the other subset P_{right} contains the points with x-coordinate larger than x_{mid}.
5. $v_{\text{left}} \leftarrow$ BUILD2DRANGETREE(P_{left})
6. $v_{\text{right}} \leftarrow$ BUILD2DRANGETREE(P_{right})
7. Create a node v storing x_{mid}, make v_{left} the left child of v, make v_{right} the right child of v, and make $\mathcal{T}_{\text{assoc}}$ the associated structure of v.
8. **return** v

Complexitatea construirii arborelui de cautare in 2D

- Aceeasi ca a arborelui 1D, inafara faptului ca la fiecare nivel arborii secondari sunt de asemenea construiti.
- **Teorema:** Complexitatea spatiala este $\Theta(n \log n)$.
- **Demonstratie:** Dimensiunea arborelui primar este $\Theta(n)$. Fiecare din cele $\Theta(\log n)$ niveluri ale sale corespunde unei colectii de arbori secundari ce contine toate cele n puncte.
- Complexitate de timp (analiza naiva):

$$T(n) = \begin{cases} O(1) & n = 1 \\ O(n \lg n) + 2T\left(\frac{n}{2}\right) & n > 1 \end{cases}$$

$$= O(n \lg^2 n)$$

Complexitatea construirii arborelui de cautare in 2D

Imbunatatire:

- Sursa de ineficienta: se sorteaza repetat dupa coordonata y!
- Se va sorta dupa y **o singura data**, si se vor copia datele in apelurile recursive (in timp liniar).
- Ecuatia recursiva rezultanta este:

$$T(n) = \begin{cases} \mathrm{O}(1) & n = 1 \\ \mathrm{O}(n) + 2T\left(\frac{n}{2}\right) & n > 1 \end{cases}$$

$$= \mathrm{O}(n \lg n)$$

Cautari in 2D (4)

Algorithm 2DRANGEQUERY($\mathcal{T}, [x : x'] \times [y : y']$)
Input. A 2-dimensional range tree $\mathcal{T}$ and a range $[x : x'] \times [y : y']$.
Output. All points in $\mathcal{T}$ that lie in the range.

1. $v_{split} \leftarrow$ FINDSPLITNODE($\mathcal{T}, x, x'$)
2. **if** v_{split} is a leaf
3. **then** Check if the point stored at v_{split} must be reported.
4. **else** (* Follow the path to x and call 1DRANGEQUERY on the subtrees right of the path. *)
5. $v \leftarrow lc(v_{split})$
6. **while** v is not a leaf
7. **do if** $x \leqslant x_v$
8. **then** 1DRANGEQUERY($\mathcal{T}_{assoc}(rc(v)), [y : y']$)
9. $v \leftarrow lc(v)$
10. **else** $v \leftarrow rc(v)$
11. Check if the point stored at v must be reported.
12. Similarly, follow the path from $rc(v_{split})$ to x', call 1DRANGEQUERY with the range $[y : y']$ on the associated structures of subtrees left of the path, and check if the point stored at the leaf where the path ends must be reported.

Complexitatea cautarii in 2D

Ecuatia de recurenta:

$$T(n) = O(\lg n) + \sum_{v} (\lg n + k_v) = O(\lg^2 n + k)$$

Traversarea structurii primare

Apeluri catre structura secundara

Traversare a structurii secundare

Raportare