Übung 9 – Algorithmen II

Sebastian Lamm, Demian Hespe – *lamm@kit.edu,hespe@kit.edu*

`http://algo2.iti.kit.edu/AlgorithmenII_WS18.php`

Institut für Theoretische Informatik - Algorithmik II

Themenübersicht

- Geometrie
  (was und warum?)
- Sweepline
  - allgemeines Prinzip
  - Beispiel: Berechnung einer Skyline
- Linienschnitt
  (überlappende Liniensegmente)
- Rechts oder Links?
  (Orientierung eines Punktes zu einer Linie)

Geometrische Algorithmen

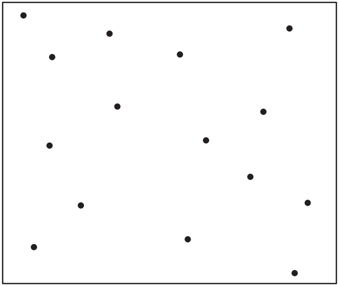
- geomtetrische Varianten bekannter Probleme
  - Spezialfälle oft einfacher
  - allgemeines TSP: nicht approximierbar (wenn P $\neq$ NP)
  - metric TSP: 1.5-Approximation
  - euclidean TSP: ϵ-Approximation
    - Laufzeit in $O(n(\log n)^{O(\frac{1}{\epsilon}\sqrt{d})^{d-1}})$

- geometrisch motivierte Probleme
  - Punktlokalisierung
  - Bewegungsplanung (Robotik)
  - Sichtbarkeitsgraphen/Prüfung
  - Streckenschnitt
  - ...

Geometrische Methoden

Datenstrukturen

- Baumstrukturen
  - Interval Tree (1-dim)
  - Quad Tree (2-dim)
  - **B**inary**S**pace**P**artition Tree (3-dim)
  - k-d-Tree (n-dim)
  - Wavelet-Tree (2-dim)
- Facetten
  - Delaunay Triangulierung
  - Voronoi Diagramm



Strukturierter Zugriff

- Sweepline
  - sortiert
  - topologisch sortiert

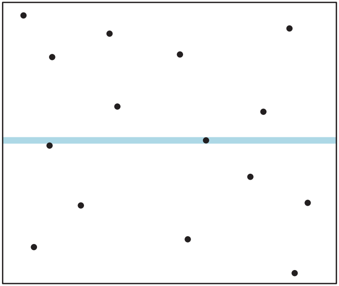
Geometrische Methoden

Datenstrukturen

- Baumstrukturen
  - Interval Tree (1-dim)
  - Quad Tree (2-dim)
  - **B**inary**S**pace**P**artition Tree (3-dim)
  - k-d-Tree (n-dim)
  - Wavelet-Tree (2-dim)
- Facetten
  - Delaunay Triangulierung
  - Voronoi Diagramm

Strukturierter Zugriff

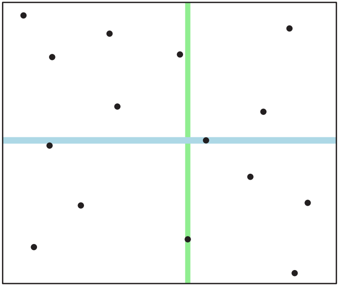
- Sweepline
  - sortiert
  - topologisch sortiert



Geometrische Methoden

Datenstrukturen

- Baumstrukturen
  - Interval Tree (1-dim)
  - Quad Tree (2-dim)
  - **B**inary**S**pace**P**artition Tree (3-dim)
  - k-d-Tree (n-dim)
  - Wavelet-Tree (2-dim)
- Facetten
  - Delaunay Triangulierung
  - Voronoi Diagramm



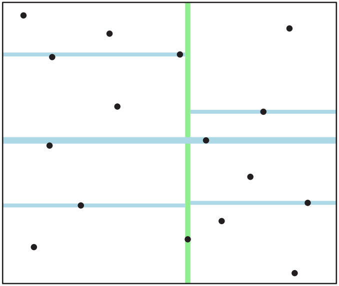
Strukturierter Zugriff

- Sweepline
  - sortiert
  - topologisch sortiert

Geometrische Methoden

Datenstrukturen

- Baumstrukturen
  - Interval Tree (1-dim)
  - Quad Tree (2-dim)
  - **B**inary**S**pace**P**artition Tree (3-dim)
  - k-d-Tree (n-dim)
  - Wavelet-Tree (2-dim)
- Facetten
  - Delaunay Triangulierung
  - Voronoi Diagramm



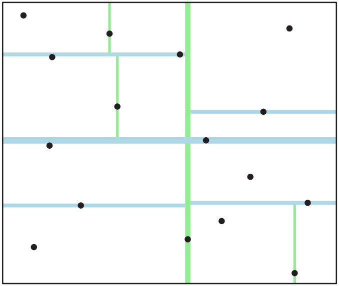
Strukturierter Zugriff

- Sweepline
  - sortiert
  - topologisch sortiert

Geometrische Methoden

Datenstrukturen

- Baumstrukturen
  - Interval Tree (1-dim)
  - Quad Tree (2-dim)
  - **B**inary**S**pace**P**artition Tree (3-dim)
  - k-d-Tree (n-dim)
  - Wavelet-Tree (2-dim)
- Facetten
  - Delaunay Triangulierung
  - Voronoi Diagramm



Strukturierter Zugriff

- Sweepline
  - sortiert
  - topologisch sortiert

Sweep-Line

Allgemein

Idee

- strukturierte Abarbeitung eines Problems
- nutze Nähe aus
  - geometrisch nahe Objekte beeinflussen sich
  - geometrisch weit entfernte Objekte (nahezu) unabhängig

im Allgemeinen

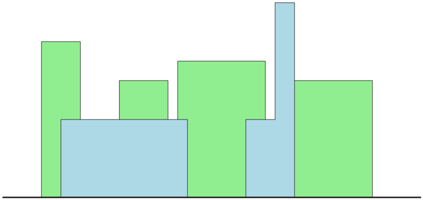
- reduziere n-dim $\rightarrow$ $(n-1)$-dim

Sweep-Line

Beispiel: Skyline

Berechnung einer Skyline

- Höhenänderungen sind einzig relevante Punkte
- jede Änderung definiert eindimensionales Problem (Maximumsbildung)

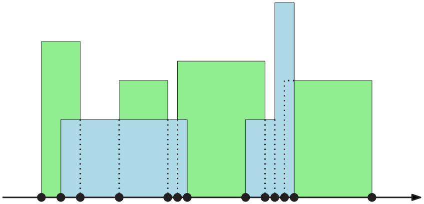


Sweep-Line

Beispiel: Skyline

Berechnung einer Skyline

- Höhenänderungen sind einzig relevante Punkte
- jede Änderung definiert eindimensionales Problem
  (Maximumsbildung)

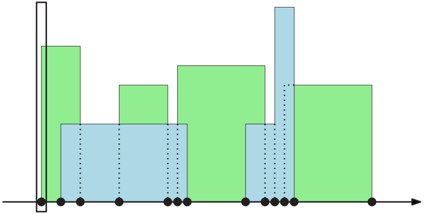


Sweep-Line

Beispiel: Skyline

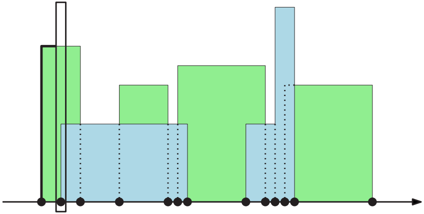
Berechnung einer Skyline

- Höhenänderungen sind einzig relevante Punkte
- jede Änderung definiert eindimensionales Problem
  (Maximumsbildung)



Sweep-Line

Beispiel: Skyline



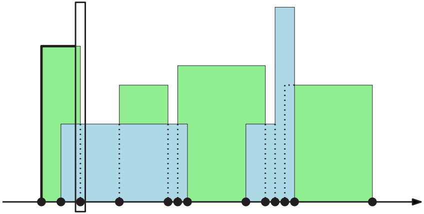
Berechnung einer Skyline

- Höhenänderungen sind einzig relevante Punkte
- jede Änderung definiert eindimensionales Problem
  (Maximumsbildung)

Sweep-Line
Beispiel: Skyline

Berechnung einer Skyline

- Höhenänderungen sind einzig relevante Punkte
- jede Änderung definiert eindimensionales Problem
  (Maximumsbildung)

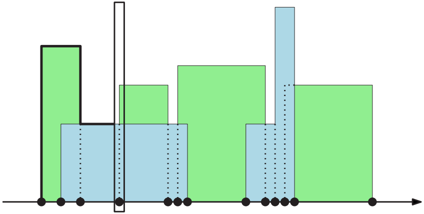


Sweep-Line

Beispiel: Skyline

Berechnung einer Skyline

- Höhenänderungen sind einzig relevante Punkte
- jede Änderung definiert eindimensionales Problem
  (Maximumsbildung)

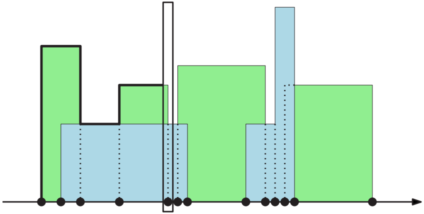


Sweep-Line

Beispiel: Skyline

Berechnung einer Skyline

- Höhenänderungen sind einzig relevante Punkte
- jede Änderung definiert eindimensionales Problem (Maximumsbildung)

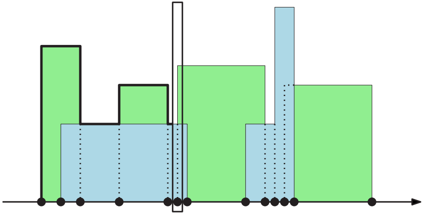


Sweep-Line

Beispiel: Skyline

Berechnung einer Skyline

- Höhenänderungen sind einzig relevante Punkte
- jede Änderung definiert eindimensionales Problem (Maximumsbildung)

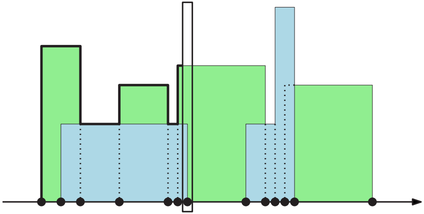


Sweep-Line

Beispiel: Skyline

Berechnung einer Skyline

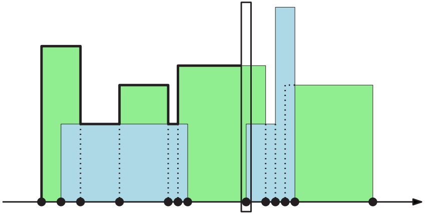
- Höhenänderungen sind einzig relevante Punkte
- jede Änderung definiert eindimensionales Problem
  (Maximumsbildung)



Sweep-Line
Beispiel: Skyline

Berechnung einer Skyline

- Höhenänderungen sind einzig relevante Punkte
- jede Änderung definiert eindimensionales Problem
  (Maximumsbildung)

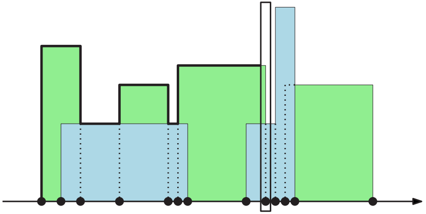


Sweep-Line

Beispiel: Skyline

Berechnung einer Skyline

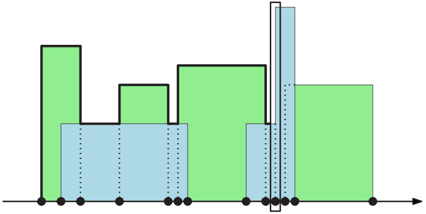
- Höhenänderungen sind einzig relevante Punkte
- jede Änderung definiert eindimensionales Problem (Maximumsbildung)



Sweep-Line
Beispiel: Skyline

Berechnung einer Skyline

- Höhenänderungen sind einzig relevante Punkte
- jede Änderung definiert eindimensionales Problem
  (Maximumsbildung)

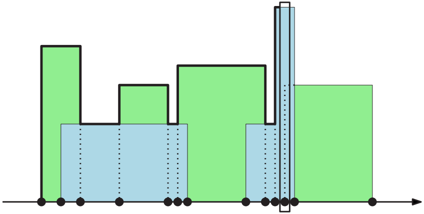


Sweep-Line

Beispiel: Skyline

Berechnung einer Skyline

- Höhenänderungen sind einzig relevante Punkte
- jede Änderung definiert eindimensionales Problem (Maximumsbildung)

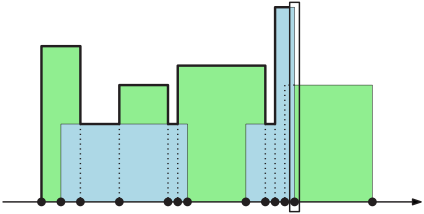


Sweep-Line

Beispiel: Skyline

Berechnung einer Skyline

- Höhenänderungen sind einzig relevante Punkte
- jede Änderung definiert eindimensionales Problem (Maximumsbildung)

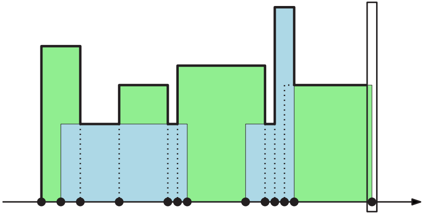


Sweep-Line

Beispiel: Skyline

Berechnung einer Skyline

- Höhenänderungen sind einzig relevante Punkte
- jede Änderung definiert eindimensionales Problem
  (Maximumsbildung)

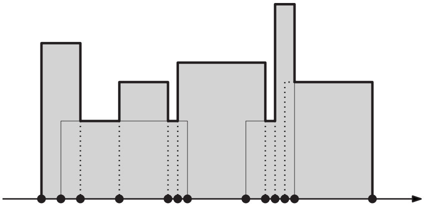


Sweep-Line

Beispiel: Skyline

Berechnung einer Skyline

- Höhenänderungen sind einzig relevante Punkte
- jede Änderung definiert eindimensionales Problem (Maximumsbildung)

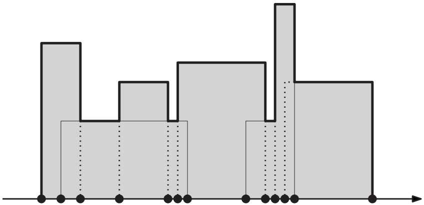


Sweep-Line

Beispiel: Skyline

Berechnung einer Skyline

- Höhenänderungen sind einzig relevante Punkte
- jede Änderung definiert eindimensionales Problem (Maximumsbildung)



Problem: effiziente Lösung des 1-dimensionalen Problems

- ineffizient: $\mathcal{O}(n^2)$ (vergleiche Linienschnitt)
- Ziel hier: Algorithmus mit $O(n \log n)$ Zeit

One-Dimensional Problem

Sortierte Liste

- Array $\rightarrow \mathcal{O}(n)$ für Einfügen/Löschen
- Linked List $\rightarrow \mathcal{O}(n)$ für Positionsbestimmung

One-Dimensional Problem

Sortierte Liste

- Array $\rightarrow \mathcal{O}(n)$ für Einfügen/Löschen
- Linked List $\rightarrow \mathcal{O}(n)$ für Positionsbestimmung

Lösung: Priority Queue

- alle Operationen in maximal $\mathcal{O}(\log n)$ möglich

Pseudocode (1/2)

Data: List of buildings (begin, height, end)
Result: Skyline coordinates (x, height)
$i \leftarrow 0$;
foreach $(b, h, e) \in L$ **do**
  $L' \leftarrow L' \cup (b, h, \text{“}b\text{“}, i) \cup (e, g, \text{“}e\text{“}, i)$;
  $i \leftarrow i + 1$;
end
sort L' in lexicographical order;
priority queue $q \leftarrow \emptyset$;
...

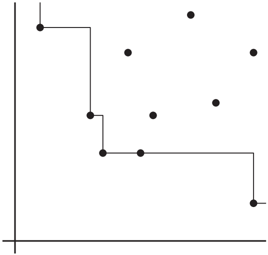
Pseudocode (2/2)

```
...
while L' ≠ ∅ do
    actpos ← first(L').pos;
    while L' ≠ ∅ and actpos = first(L').pos do
        (pos,height,label,index) ← popfirst(L');
        if label = "b" then
            add (q, (height,index));
        else
            remove (q,index);
        end
    end
    if q ≠ ∅ then
        print actpos,first(q).height;
    else
        print actpos,0;
    end
end
```

Skyline

Das eigentliche Skyline Problem:

- Berechnung nicht dominierter Punktmengen
  - multikriterielle Bewertung von Tupeln $(t_1, \ldots, t_n)$
  - Dominierungsbedingung:
    a dominiert *b* genau dann wenn
    - $\forall i : a_i \leq b_i$
    - $\exists i : a_i < b_i$



Linienschnitt

überlappende Liniensegmente

Sortierkriterien ($\mathcal{O}(n \log n)$)

1. Steigung des Segments
2. y-Achsenabschnitt
3. x-Koordinaten der Punkte

$\rightarrow$ gruppiert genau Elemente die auf einer Geraden liegen
((nicht zwingend überlappend))

Linearer Scan

- speichere aktive Elemente in Priority Queue ($\mathcal{O}(n \log n)$)
  - sortiert nach x-Koordinate
  - beenden aktuelle Segmente
- alle gleichzeitig aktiven Elemente überlappen

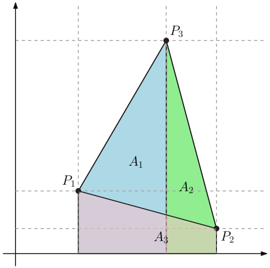
Punktorientierung

Vorzeichenbehaftete Fläche

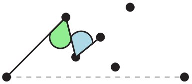
- Dreiecksfläche als Summe dreier Trapezflächen
- Verschiebung von P_1 nach $(0,0)$ ergibt:

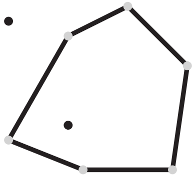
$$A = \frac{1}{2}\begin{vmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{vmatrix}$$

- Vorzeichen positiv gdw. P_1, P_2, P_3 CCW



- Unterproblem von Algorithmen
  - Graham Scan
  - Test auf Enthaltensein
    (Punkt in konvexem Polygon)

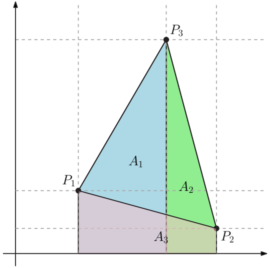




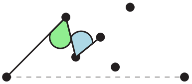
Punktorientierung

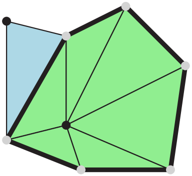
Vorzeichenbehaftete Fläche

- Dreiecksfläche als Summe dreier Trapezflächen
- Verschiebung von P_1 nach $(0,0)$ ergibt:
  $A = \frac{1}{2} \begin{vmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{vmatrix}$
- Vorzeichen positiv gdw. P_1, P_2, P_3 CCW



- Unterproblem von Algorithmen
  - Graham Scan
  - Test auf Enthaltensein
    (Punkt in konvexem Polygon)





Ende!

Feierabend!