

Übung 10 – Algorithmen II

Hans-Peter Lehmann, Daniel Seemaier – {hans-peter.lehmann, daniel.seemaier}@kit.edu
http://algo2.iti.kit.edu/AlgorithmenII_WS21.php

Institut für Theoretische Informatik - Algorithmik II

```
result = current_weight;
return true;

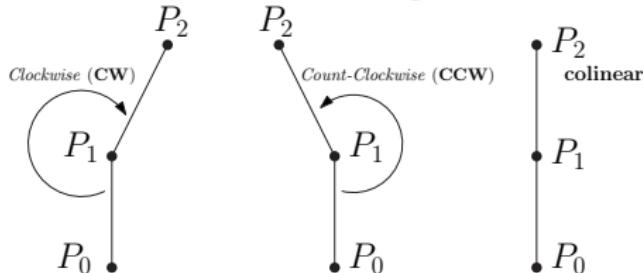
for( EdgeID eid = graph.edgeBegin( current ); eid != graph.edgeEnd( current ); ++eid ){
    const Edge & edge = graph.getEdge( eid );
    COUNTING( statistic_data.inc( DijkstraStatisticData::TOUCHED_EDGES ) );
    if( edge.forward ){
        COUNTING( statistic_data.inc( DijkstraStatisticData::RELAXED_EDGES ) );
        Weight new_weight = edge.weight + current_weight;
        GUARANTEE( new_weight >= current_weight, std::runtime_error, "Weight overflow detected." );
        if( !priority_queue.isReached( edge.target ) ){
            COUNTING( statistic_data.inc( DijkstraStatisticData::SUCCESSFULLY_RELAXED_EDGES ) );
            COUNTING( statistic_data.inc( DijkstraStatisticData::REACHED_NODES ) );
            priority_queue.push( edge.target, new_weight );
        } else {
            if( priority_queue.getCurrentKey( edge.target ) > new_weight ){
                COUNTING( statistic_data.inc( DijkstraStatisticData::UNREACHABLE_RELAXED_EDGES ) );
                priority_queue.decreaseKey( edge.target, new_weight );
            }
        }
    }
}
```

Themenübersicht

- Besprechung der Evaluation
- Geometrische Algorithmen
 - Graham's Scan
 - Range queries

Punktorientierung

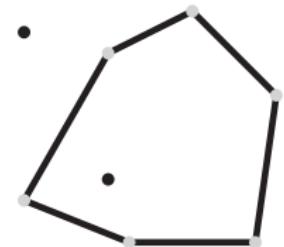
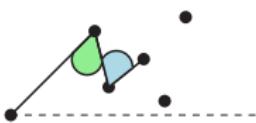
- Gegeben drei Punkte $P_0, P_1, P_2 \Rightarrow$ bestimme Orientierung
- Sei $\vec{a} = \overrightarrow{P_0P_1}$ und $\vec{b} = \overrightarrow{P_1P_2}$
- $CCW(P_0, P_1, P_2) = a_x \cdot b_y - a_y \cdot b_x$
- $CCW(P_0, P_1, P_2) > 0 \Rightarrow \text{CCW}$
- $CCW(P_0, P_1, P_2) = 0 \Rightarrow \text{Colinear}$
- $CCW(P_0, P_1, P_2) < 0 \Rightarrow \text{CW}$



■ Unterproblem von Algorithmen

- Graham Scan
- Test auf Enthaltensein

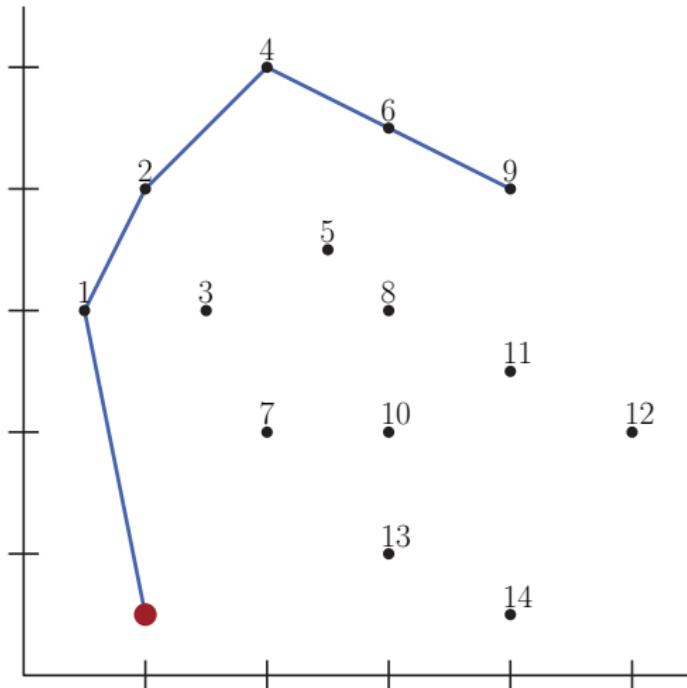
Punkt in konvexem Polygon



Konvexe Hülle

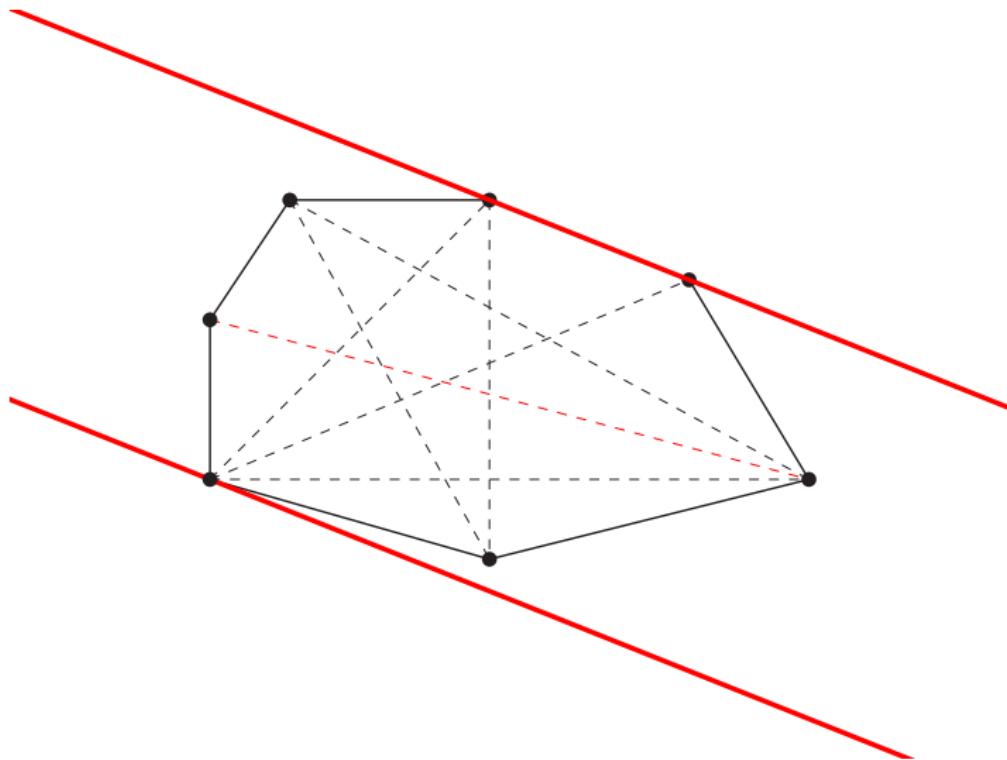
Graham-Scan

1. Finde Punkt P_0 mit kleinster y -Koordinate
 - Gibt es mehrere Punkte mit gleicher y -Koordinate, dann nehme Punkt mit kleinsten x -Koordinate
2. Sortiere alle Punkte in absteigendem Winkel relativ zu P_0
3. Iteriere über alle Punkte P_i ($i > 2$) und betrachte Dreieck $H_{k-1}H_kP_i$ (wobei H_i i -ter Punkt in der aktuellen konvexen Hülle)
 - *Rechtsknick*: Füge P_i zur konvexen Hülle H hinzu
 - *Linksknick*: Lösche H_k aus bisheriger konvexer Hülle H



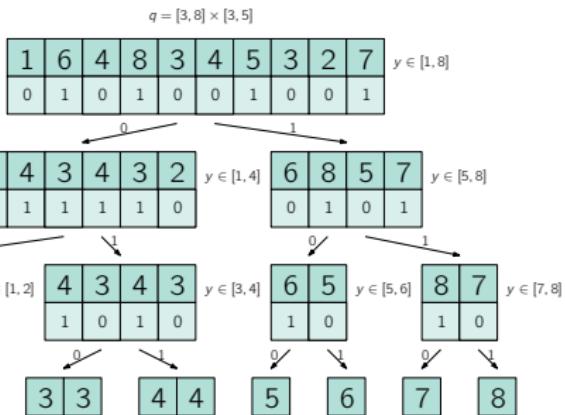
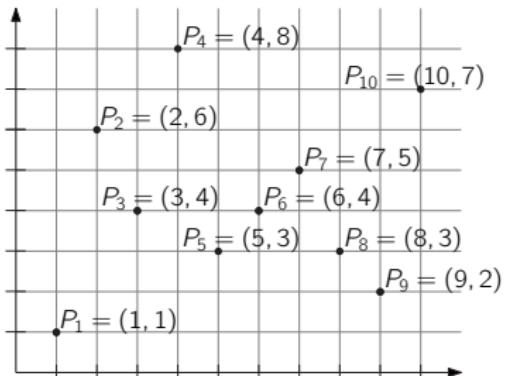
Konvexe Hülle

Anwendung: Durchmesser von Punktmenge: Rotating Calipers



2D Range Queries

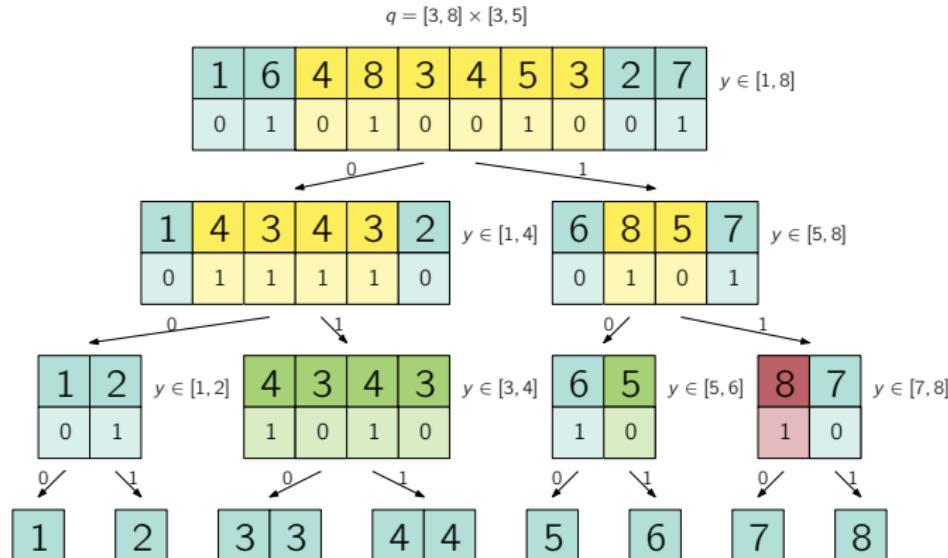
Wavelet Tree



Auf einem $n \times n$ Grid braucht ein Wavelet Tree $n \log n + o(n \log n)$ Bits Speicherplatz

2D Range Queries

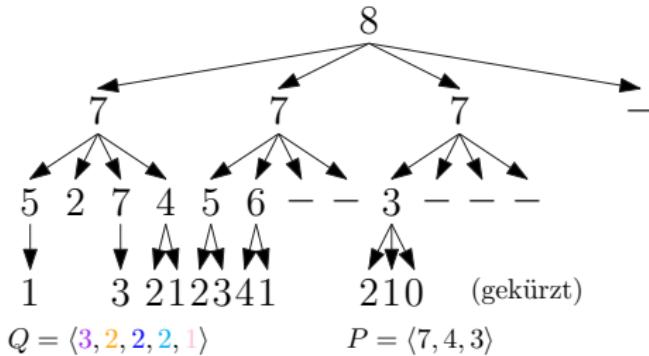
Wavelet Tree - Count Operation



2D Range Queries

K² Treaps

5			8	5		7	6
1			2	3		4	1
	7	4	2				
7	3	1					
			7				
				3	2		
					1	0	





Feierabend!