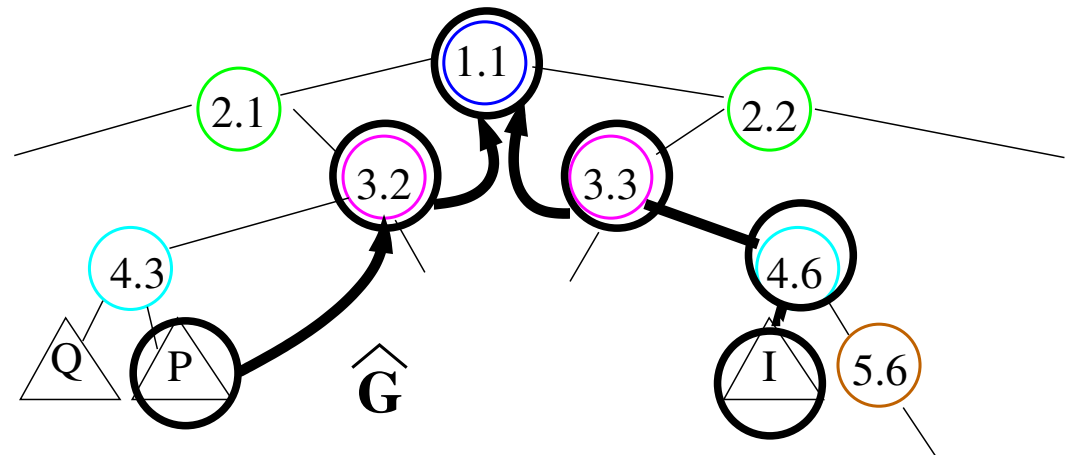
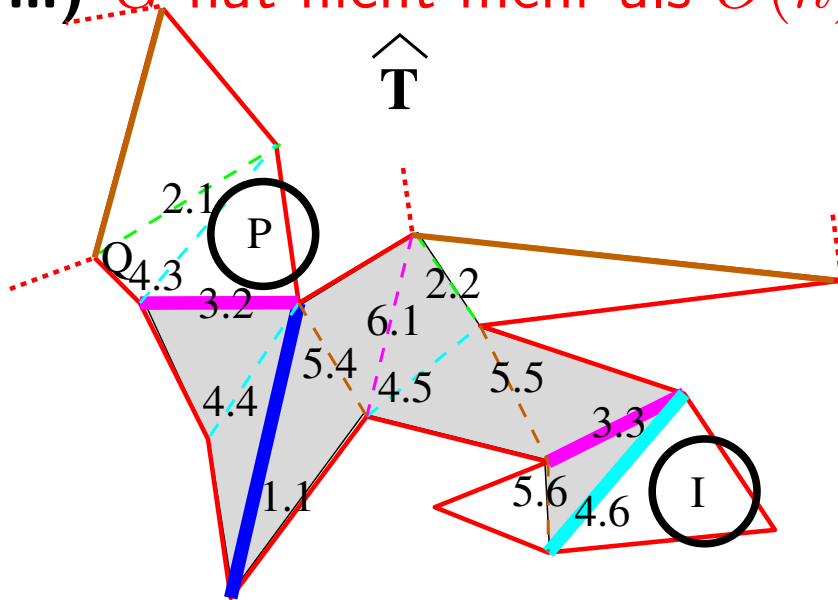


# Offline Bewegungsplanung: Preprocessing und Durchmesser

Elmar Langetepe  
University of Bonn

# Eigenschaften von $\hat{G}$ : Lemma 1.13

- i) Pfad zwischen zwei Dreiecken entlang sukzessiver Diagonalen existiert!
- ii) Wir finden den Weg in  $O(\log n)$  Zeit!
- iii)  $\hat{G}$  hat nicht mehr als  $O(n)$  Kanten!



### iii) Komplexität von $\hat{G}$

- Untere Kanten von  $v$  aus:  $\max$   
 $2 \times \text{height}(v)$

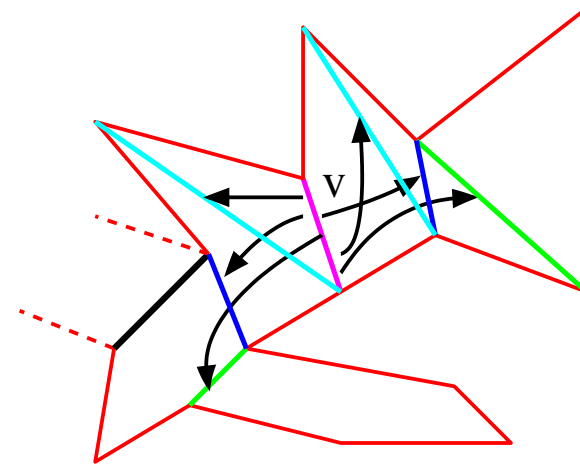
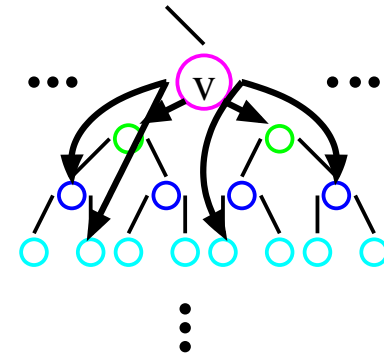
- Balance: Teilbaum bei  $v$   
 hat  $\geq C \cdot \left(\frac{3}{2}\right)^{\text{height}(v)}$  Blätter  
 (Tafel)

- Anzahl Knoten der Höhe  $h$ :

$$\leq \frac{n}{C \left(\frac{3}{2}\right)^h} = \frac{n}{C} \left(\frac{2}{3}\right)^h$$

- Sum. über alle Höhen:

$$\sum_{h=1}^{\log_3 n} (2h) \times \left( \left(\frac{2}{3}\right)^h \times \frac{n}{C} \right) \in O(n)$$



# Konstruktion $\hat{G}$

- Cutting-Theorem (Übung): konstruktiv!!
- Durchlauf von  $T^*$  von den Blättern aus!
- Während des Aufbaus: Insgesamt  $O(n)$  viele Diagonalen überschreiten?
- Aufbau geht auch in  $O(n)$ ■

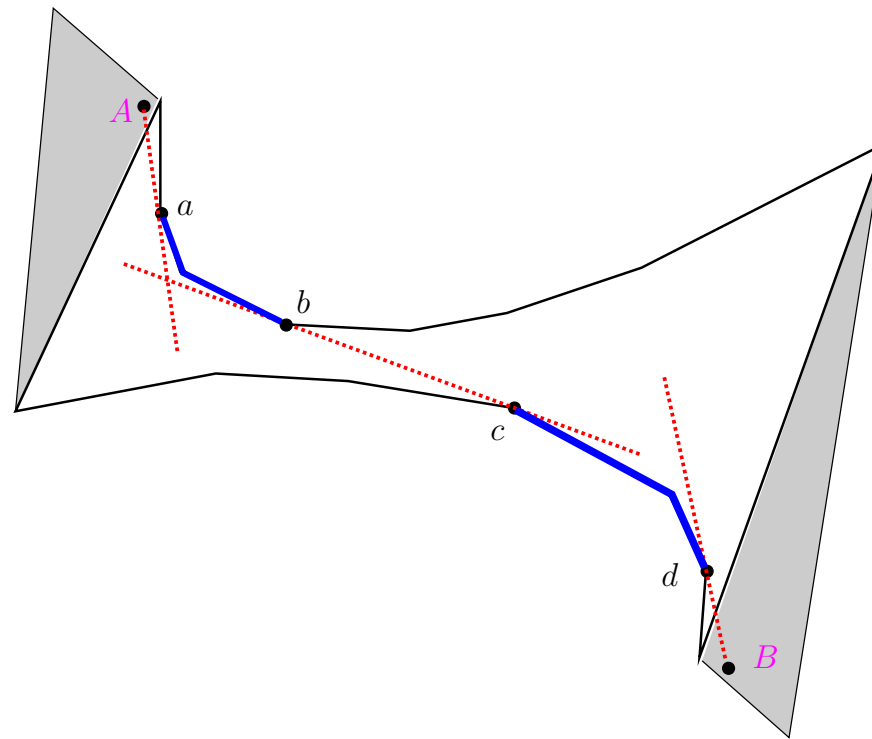
# Zusammenfassung des Problems/Analyse

1. Berechne Triangulation  $T$  und Dual  $T^*$ :  $O(n)$
2. Berechne hierarch. bal. Baum  $\hat{T}$ , Sch.-Graph  $\hat{G}$ :  $O(n)$
3. Komplexität  $\hat{G}$ :  $O(n)$
4. Berechne *alle* Sanduhren von  $\hat{G}$ :  $O(n)$
5. Navigation zw. Dreiecken in  $\hat{G}$ : Sequenz v. Diagonalen:  $O(\log n)$
6. Konkat. Sanduhren für finale Sanduhr:  $O(\log n)$
7. Berechne Shortest Path aus final. Sanduhr:  $O(\log n + k)$

Query: Start  $A \in P$ , Ziel  $B \in I$ : Löse 5), 6) und 7)!!

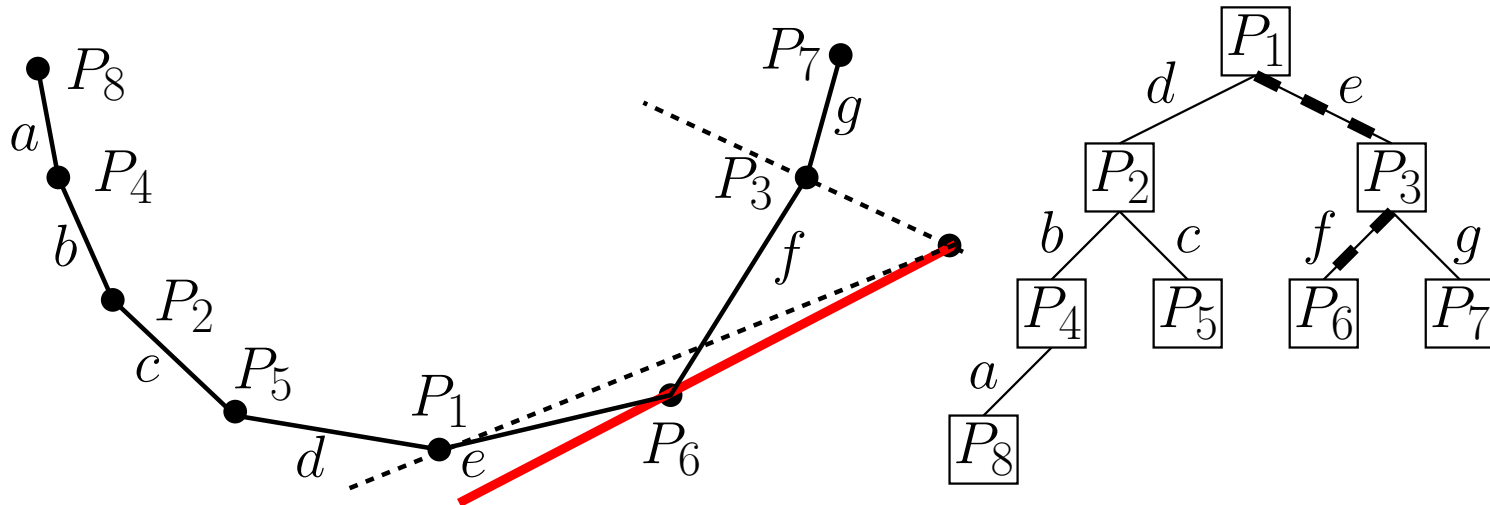
# Berechne Shortest Path aus finaler Sanduhr

- Finale Sanduhr, Ziel und Start ■
- Data structure: Tangentenpunkte in logarithm. Zeit ■
- Länge in  $O(1)$ , Pfad in  $O(k)$  ■



# Datenstruktur Hourglass

- Ketten der Sanduhren in bal. Baum speichern ■
- Tangente in logarithm. Zeit berechnen ■
- Länge in  $O(1)$ , Pfad in  $O(k)$  ■



# Effektiver Aufbau aller Sanduhren in $O(n)$

- Sanduhr  $S(d_i, d_j)$  mit  $d_i$  und  $d_j$  ■
- Aus den Sanduhren mit  $d'$  in  $O(\log(\text{Komp}(S(d_i, d_j))))$  ■
- Entspricht  $O(\text{height}(d'))$  viele solcher Sanduhren ■
- Abschätzen durch  $O(\text{height}(d_i))$  (größere Höhe  $d_i$ ) ■
- $2 * \text{height}(d_i)$  viele solcher  $d'$  für  $d_i$  ■
- Für alle Diagonalen! ■
- Sum. über alle Höhen:  $\sum_{h=1}^{\log_3 n} 2 \times \left( \left( \frac{2}{3} \right)^h \times n \right) \times h^2 \in O(n)$  ■
- **Lemma 1.14** ■



# Konstruktion $\hat{G}$ + Sanduhren

- Cutting-Theorem (Übung): konstruktiv!!
- Durchlauf von  $T^*$  möglich
- Dabei Sanduhren aufbauen möglich
- Effektiv auch in  $O(n)$ !

# Sanduhrenlemma: Lemma 1.15

Sanduhr  $S(d_i, d_j)$  zwischen  $d_i$  und  $d_j$  mit  $m(d_i, d_j)$  Diagonalen.

■ Datenstruktur mit folgender Eigenschaft existiert:

- i) Entfernung zwischen Punkten in  $D_i$  und  $D_j$  in  $O(\log(m(d_i, d_j)))$  ■
- ii) Kürzeste Wege zwischen Punkten in  $D_i$  und  $D_j$  in  $O(\log(m(d_i, d_j)) + k)$  ■
- iii) Konkatenation zweier Sanduhren  $S(d_i, d_j)$  und  $S(d_j, d_l)$  zu einer Sanduhr in Zeit  $O(\log(m(d_i, d_j)) + \log(m(d_j, d_l)))$  ■

Beweis: Skizze für iii)!!! ■

# Algorithmus 1.6

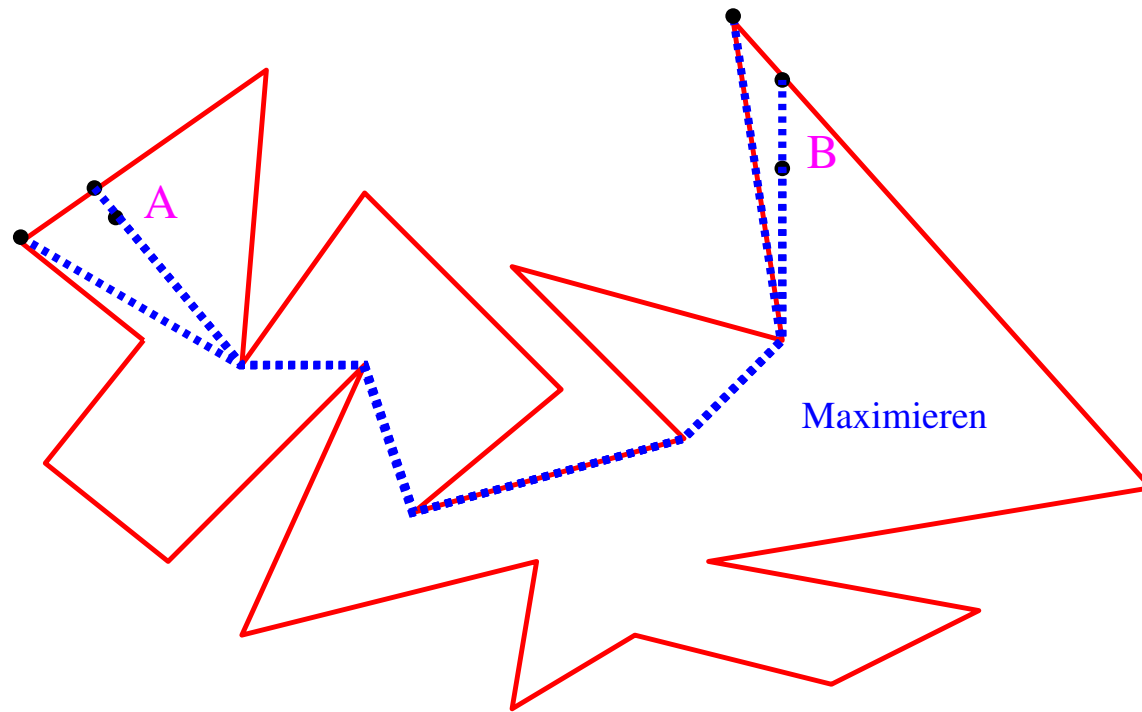
- Preprocessing:
  - $\hat{G}$  + Rooted Tree
  - Sanduhren für Kanten
  - Lokalisation Dreiecke
- Query,  $p, q$ :
  - Lokalisation  $D_p$  und  $D_q$
  - Pfad zw.  $D_p$  und  $D_q$  in  $\hat{G}$
  - Sanduhr  $S(d_p, d_q)$  aus Sanduhren entlang des Pfades
  - Länge oder kürzester Weg

# Guibas/Hershberger: Laufzeiten

- Datenstrukturen:  $\hat{G}$ , Rooted Tree, Sanduhren, Trapezzerlegung
- Preprocessing Zeit:  $O(n)$
- Komplexität:  $O(n)$
- Lokalisation Dreiecke:  $O(\log n)$
- Pfad in  $\hat{G}$  in  $O(\log n)$
- Konkatenation Sanduhren in  $O(\log^2 n)$  ( $O(\log n)$ )
- Query:  $O(\log n + k)$  oder  $O(\log n)$  für die Länge
- **Theorem 1.16**

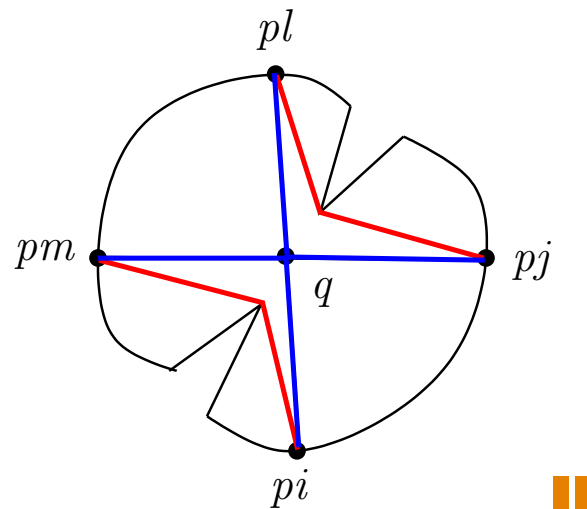
## 1.2.3 Durchmesser einfacher Polygone

- Einfaches Polygon  $P$
- Längster Kürzester Weg zwischen zwei Punkten
- Endpunkte sind Ecken des Polygons  $n^2$  Kandidatenpaare
- Formal:  $\max_{p_i, p_j \text{ Ecken von } P} d(p_i, p_j)$



# Idee der Berechnung:

- $p_i, p_j, p_l, p_m$  entlang des Randes ■
- Monge Eigenschaft:  $d(p_i, p_m) + d(p_j, p_l) \leq d(p_i, p_l) + d(p_j, p_m)$  ■
- Wege  $\pi(p_i, p_l)$  und  $\pi(p_j, p_m)$  schneiden sich ■
- Dreiecksungleichungen anwenden!! ■



# Matrix $A$ mit schöner Eigenschaft!

$$\begin{array}{c}
 \text{1} \\
 \text{2} \\
 \text{3} \\
 \vdots \\
 n-1 \\
 n
 \end{array}
 \begin{array}{c}
 \text{1} \quad \text{2} \quad \text{3} \quad \dots \quad n-1 \quad n \\
 \left( \begin{array}{cccccc}
 1-n & d(1,2) & d(1,3) & \dots & d(1,n-1) & d(1,n) \\
 1-n & 2-n & d(2,3) & \dots & d(2,n-1) & d(2,n) \\
 1-n & 2-n & 3-n & \dots & d(3,n-1) & d(3,n) \\
 \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
 1-n & 2-n & 3-n & \dots & \dots & d(n-1,n) \\
 1-n & 2-n & 3-n & \dots & \dots & 0
 \end{array} \right)
 \end{array}$$

$d(i, j)$  entspricht  $d(p_i, p_j)$  und  $p_1, p_2, \dots, p_n$  Knoten entlang des Randes von  $P$ !

## Def. 1.17: Monotone Matrix!

$n \times m$ -Matrix  $A = (a_{i\ell})$  heißt *monoton*, falls

$$\forall 1 \leq i < j \leq n, 1 \leq k < \ell \leq m : (a_{ik} < a_{i\ell} \Rightarrow a_{jk} < a_{j\ell}).$$

$$\begin{array}{cc} & \begin{array}{cc} k & \ell \end{array} \\ \begin{array}{c} i \\ j \end{array} & \left( \begin{array}{cc} a_{ik} & < & a_{i\ell} \\ & \Downarrow & \\ a_{jk} & < & a_{j\ell} \end{array} \right) \end{array}$$



# Linkeste Zeilenmaxima weiter nach rechts

$$\begin{pmatrix} 5 & 5 & 5 & 5 & 5 \\ 1 & 7 & 9 & 6 & 3 \\ 4 & 6 & 10 & 7 & 12 \end{pmatrix}$$

## Lemma 1.19: A ist monoton!

$$\begin{array}{c}
 1 \quad 2 \quad 3 \quad \dots \quad n-1 \quad n \\
 \begin{array}{c}
 1 \\
 2 \\
 3 \\
 \vdots \\
 n-1 \\
 n
 \end{array}
 \left( \begin{array}{cccccc}
 1-n & d(1,2) & d(1,3) & \dots & d(1,n-1) & d(1,n) \\
 1-n & 2-n & d(2,3) & \dots & d(2,n-1) & d(2,n) \\
 1-n & 2-n & 3-n & \dots & d(3,n-1) & d(3,n) \\
 \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
 1-n & 2-n & 3-n & \dots & \dots & d(n-1,n) \\
 1-n & 2-n & 3-n & \dots & \dots & 0
 \end{array} \right)
 \end{array}$$

■ Beweis!!! ■

## Theorem 1.20: Maxima für monotone Matrizen

- Monotone  $n \times m$ -Matrix  $A = (a_{il})$ ,  $m \geq n$ ■
- Alg.1.7 Spaltenreduktion (Spalten streichen): monotone  $n \times n$ -Matrix  $A'$ ■
- Zeilenmaxima  $(A, i)$  ■ aus Zeilenmaxima  $(A', i)$  gewinnen■
- Alg.1.8 Zeilenreduktion (gerade Zeilen auswählen): Monotone  $n \times m$ -Matrix  $B$ ■
- Zeilenmaxima ungerade Zeilen ■ aus Zeilenmaxima gerade Zeilen gewinnen■
- Rekursiv mit Spaltenreduktion (Alg. 1.7)!!■

## Alg. 1.7: Spaltenreduktion

Invariante:  $a_{1,1} \geq a_{1,2} \geq a_{1,3} \geq a_{1,4} \geq a_{1,5}$  ■

■  $a_{2,2} \geq a_{2,3} \geq a_{2,4} \geq a_{2,5}$

$a_{3,3} \geq a_{3,4} \geq a_{3,5}$

$a_{4,4} \stackrel{?}{<} a_{4,5}$

Falls nein, dann: ■  $a_{1,1} \geq a_{1,2} \geq a_{1,3} \geq a_{1,4} \geq a_{1,5}$  ■

$a_{2,2} \geq a_{2,3} \geq a_{2,4} \geq a_{2,5}$

$a_{3,3} \geq a_{3,4} \geq a_{3,5}$

$a_{4,4} \geq a_{4,5}$

↑  
↑  
↑  
↑

## Alg. 1.7: Spaltenreduktion

Invariante:  $a_{1,1} \geq a_{1,2} \geq a_{1,3} \geq a_{1,4} \quad a_{1,5}$  ■  
■  $a_{2,2} \geq a_{2,3} \geq a_{2,4} \quad a_{2,5}$   
 $a_{3,3} \geq a_{3,4} \quad a_{3,5}$   
 $a_{4,4} \stackrel{?}{<} a_{4,5}$

Falls ja, dann: ■  $a_{1,1} \geq a_{1,2} \geq a_{1,3} \quad a_{1,5}$  ■  
 $a_{2,2} \geq a_{2,3} \quad a_{2,5}$   
 $a_{3,3} \stackrel{?}{<} a_{3,5}$   
 $a_{4,5}$

Streiche Spalte 4, weil: ■

$$a_{i,4} \leq a_{i,3} \text{ für } i = 1, 2, 3 \text{ und } a_{i,4} < a_{i,5} \text{ für } i = 4, \dots, n \quad \blacksquare$$

## Alg. 1.7: Spaltenreduktion

Letzte Zeile erreicht!■

$$\begin{array}{cccccccc} \blacksquare a_{1,1} & \geq & a_{1,2} & \geq & a_{1,3} & \cdots & a_{1,n} & \geq & a_{1,n+1} & \cdots & a_{1,m} \\ & & a_{2,2} & \geq & a_{2,3} & \cdots & a_{2,n} & \geq & a_{2,n+1} & \cdots & a_{2,m} \\ & & & & \cdots & \cdots & & & \vdots & & \\ \blacksquare & & & & \cdots & \cdots & a_{n,n} & \geq & a_{n,n+1} & \cdots & a_{n,m} \end{array}$$

Komplette Spalte  $n + 1$  Streichen!!■

Weiter mit Vergleich:  $a_{n,n} <^? a_{n,n+2}$ ■

Beispiel!!!■

## Alg. 1.7: Spaltenreduktion

- Input: monotone  $n \times m$  Matrix  $A$ ,  $m \geq n$  ■
- Output: monotone  $n \times n$  Matrix  $A'$  ■
- Zeilenmaxima von  $A'$  und  $A$  identisch ■
- Analyse: ■
  - $O(m)$  Vergleiche ■
  - $O(m)$  Zeiger für Rekonstruktion ■

## Alg. 1.8: Zeilenmaxima

- Input: monotone  $n \times m$  Matrix  $B$
- Output: alle linkesten Zeilenmaxima  $\max(i)$ ,  $1 \leq i \leq n$ ■
- Algorithmus: ■
  - $C$  Zeilen von  $B$  mit geradem Index  $O(n)$
  - $C'$  durch Spaltenreduktion( $C$ )  $O(m)$  ■
  - Rekursiv: Zeilenmaxima( $C'$ ),  $\frac{n}{2} \times \frac{n}{2}$  Matrix  $T\left(\frac{n}{2}\right)$  ■
  - Rekonstruktion (Zeiger) Zeilenmaxima von  $C$   $O(m)$ ■
  - **Berechnung** Zeilenmaxima von  $B$   $O(m)$ ■



# Berechnung Zeilenmaxima von $B$

$$\begin{pmatrix} x & x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x & x \end{pmatrix}$$

$O(m)$  Schritte !!!

# Gesamtbeispiel

$$\begin{pmatrix} \Rightarrow 5 & 4 & 3 & 2 & 1 & 0 \\ 1 & 7 & 3 & \Rightarrow 9 & 4 & 3 \\ 4 & 6 & 5 & \Rightarrow 8 & 6 & 7 \\ 3 & 4 & 2 & 6 & 7 & \Rightarrow 8 \end{pmatrix}$$

Zeilenreduktion:

$$\begin{pmatrix} 1 & 7 & 3 & 9 & 4 & 3 \\ 3 & 4 & 2 & 6 & 7 & 8 \end{pmatrix}$$

Spaltenreduktion:

$$\begin{pmatrix} \Rightarrow 9 & 3 \\ 6 & \Rightarrow 8 \end{pmatrix}$$

Zeilenreduktion:

$$(6 \ 8)$$

Spaltenreduktion:

$$(\Rightarrow 8)$$

# Theorem 1.20: Maxima für monotone Matrizen

Laufzeitanalyse: ■

■

$$T(m) \leq T\left(\frac{n}{2}\right) \text{ (Rek.)} + C \times m \text{ (Sp.Red. + Rekonstr. + Ber.)}$$

$$\text{■} \leq T\left(\frac{n}{4}\right) + C\left(m + \frac{n}{2}\right)$$

$$\text{■} \vdots \quad \vdots$$

$$\text{■} \leq T(1) + C \left( m + n \sum_{i=1}^{\log n} \frac{1}{2^i} \right) \text{■} \in O(m) \text{■}$$

Nur  $O(m)$  viele Vergleiche!!! ■

## Theorem 1.21: Durchmesser eines Polygones

- $n \times n$  Matrix  $A$  (symbolisch)■
- Preprocessing Guibas/Hershberger  $O(n)$ ■
- Zeilenmaxima von  $A$  mit  $O(n)$  Vergleichen■
- Je Vergleich  $O(\log n)$  Aufwand für Wert■
- Gesamtlaufzeit:  $O(n \log n)$ ■