Karlsruhe Institute of Technology

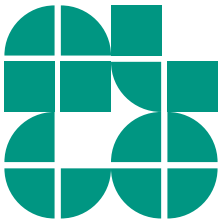
Vorlesung Algorithmische Geometrie

Well-Separated Pair Decomposition

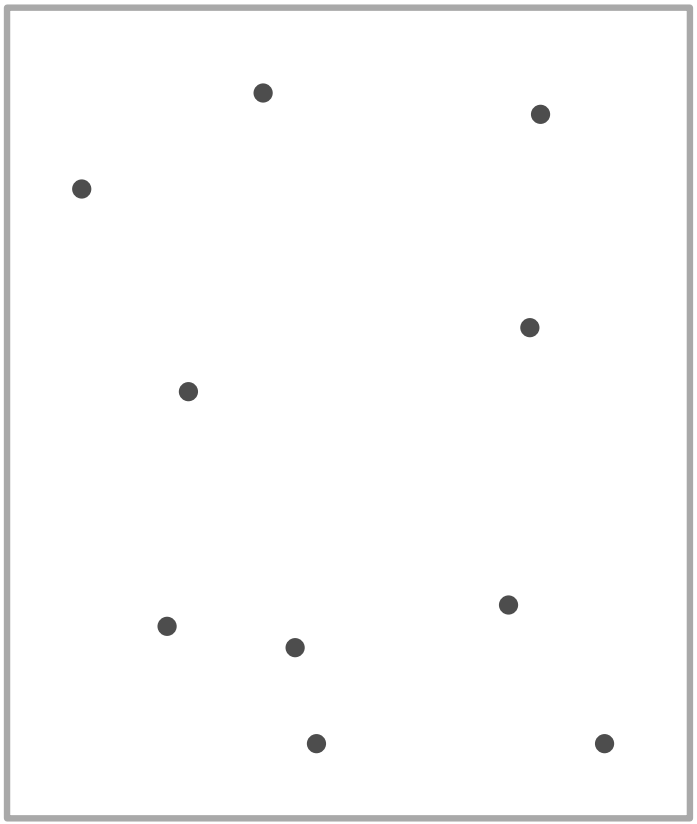
LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Martin Nöllenburg

28.06.2011



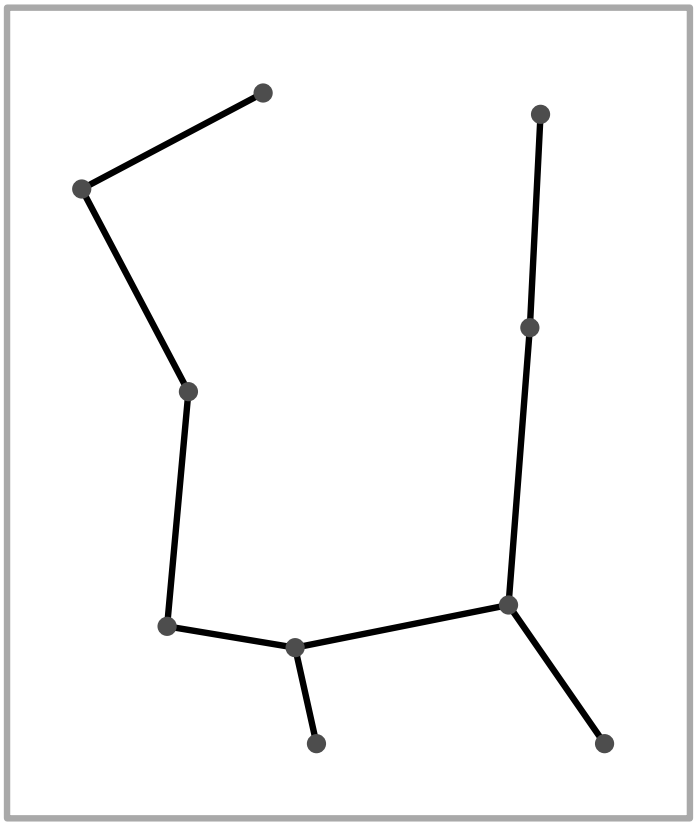
Motivation: Spanner



Aufgabe:

Eine Menge von Städten soll über ein neues Straßennetz miteinander verbunden werden.

Motivation: Spanner

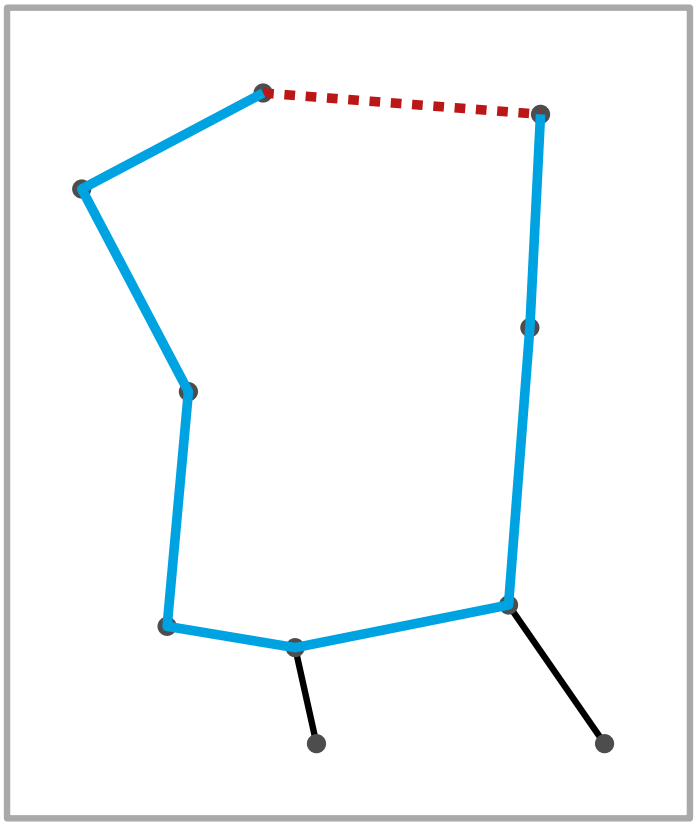


Aufgabe:

Eine Menge von Städten soll über ein neues Straßennetz miteinander verbunden werden.

1. Idee: Euklid. min. Spannbaum

Motivation: Spanner



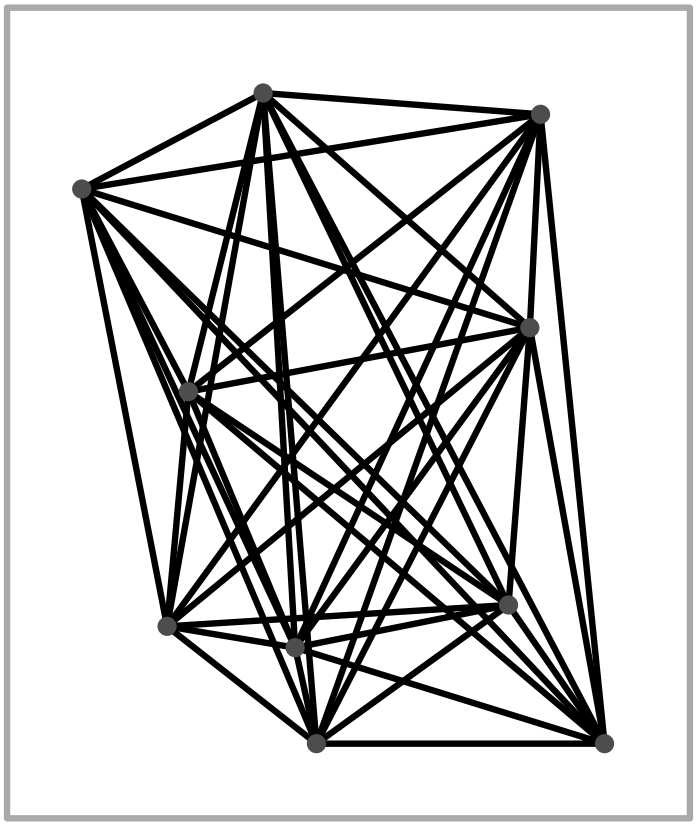
Aufgabe:

Eine Menge von Städten soll über ein neues Straßennetz miteinander verbunden werden.

Allerdings soll für kein Paar (x, y) der Weg im Straßennetz viel länger als die Distanz $||xy||$ sein.

1. Idee: Euklid. min. Spannbaum

Motivation: Spanner



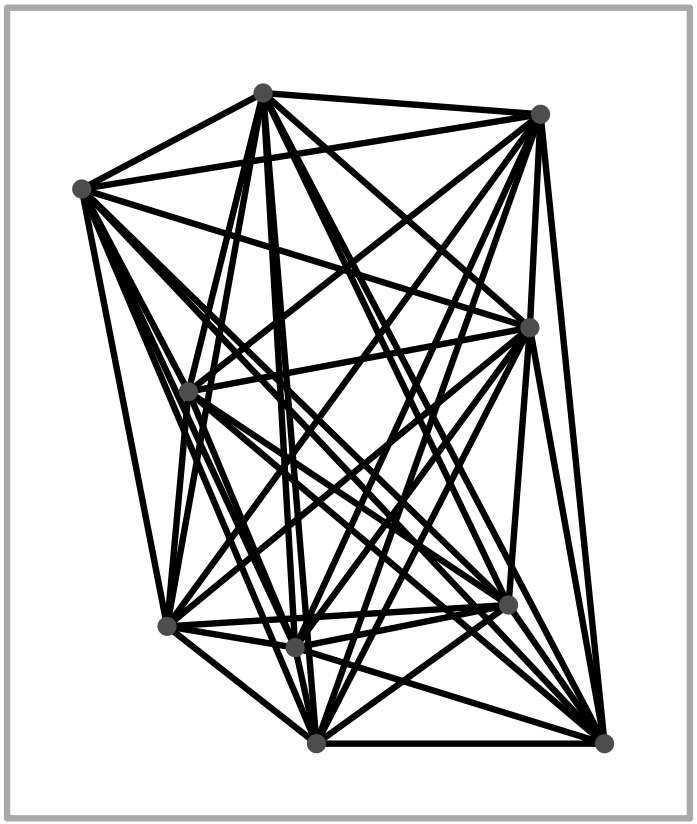
Aufgabe:

Eine Menge von Städten soll über ein neues Straßennetz miteinander verbunden werden.

Allerdings soll für kein Paar (x, y) der Weg im Straßennetz viel länger als die Distanz $||xy||$ sein.

1. Idee: Euklid. min. Spannbaum
2. Idee: vollständiger Graph

Motivation: Spanner



Aufgabe:

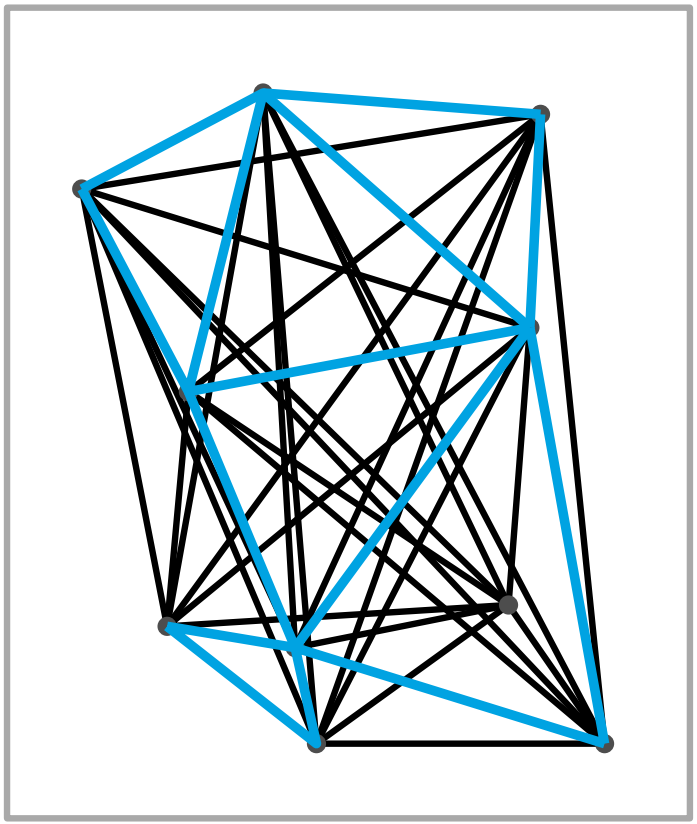
Eine Menge von Städten soll über ein neues Straßennetz miteinander verbunden werden.

Allerdings soll für kein Paar (x, y) der Weg im Straßennetz viel länger als die Distanz $||xy||$ sein.

Die Baukosten sollen im Rahmen bleiben, also z.B. nur $O(n)$ Kanten.

1. Idee: Euklid. min. Spannbaum
2. Idee: vollständiger Graph

Motivation: Spanner



Aufgabe:

Eine Menge von Städten soll über ein neues Straßennetz miteinander verbunden werden.

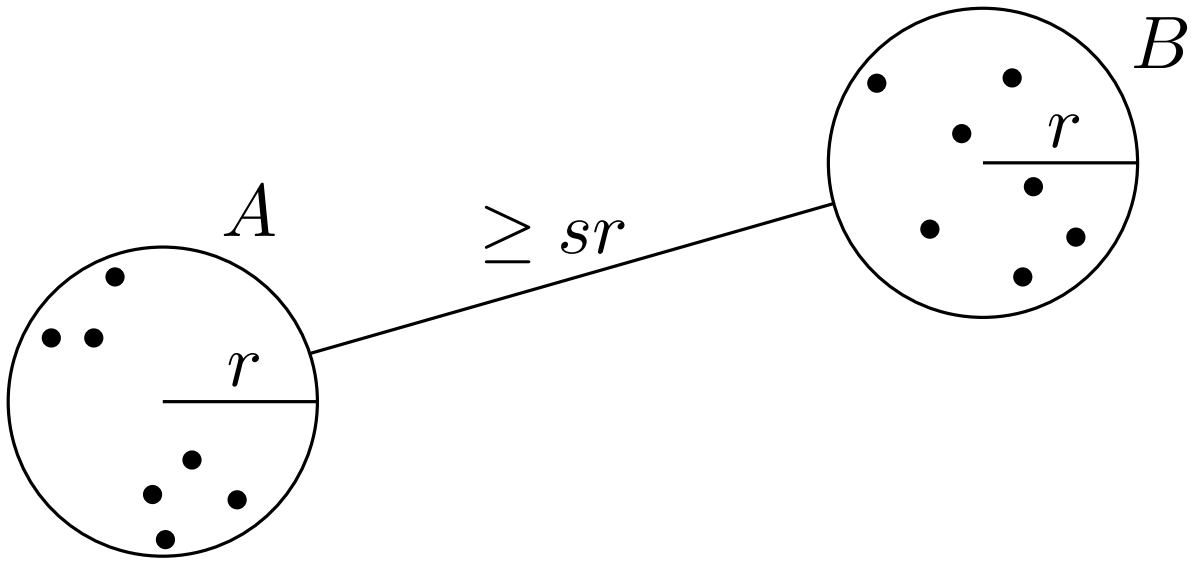
Allerdings soll für kein Paar (x, y) der Weg im Straßennetz viel länger als die Distanz $||xy||$ sein.

Die Baukosten sollen im Rahmen bleiben, also z.B. nur $O(n)$ Kanten.

1. Idee: Euklid. min. Spannbaum
2. Idee: vollständiger Graph
3. Idee: **sparse t-Spanner**

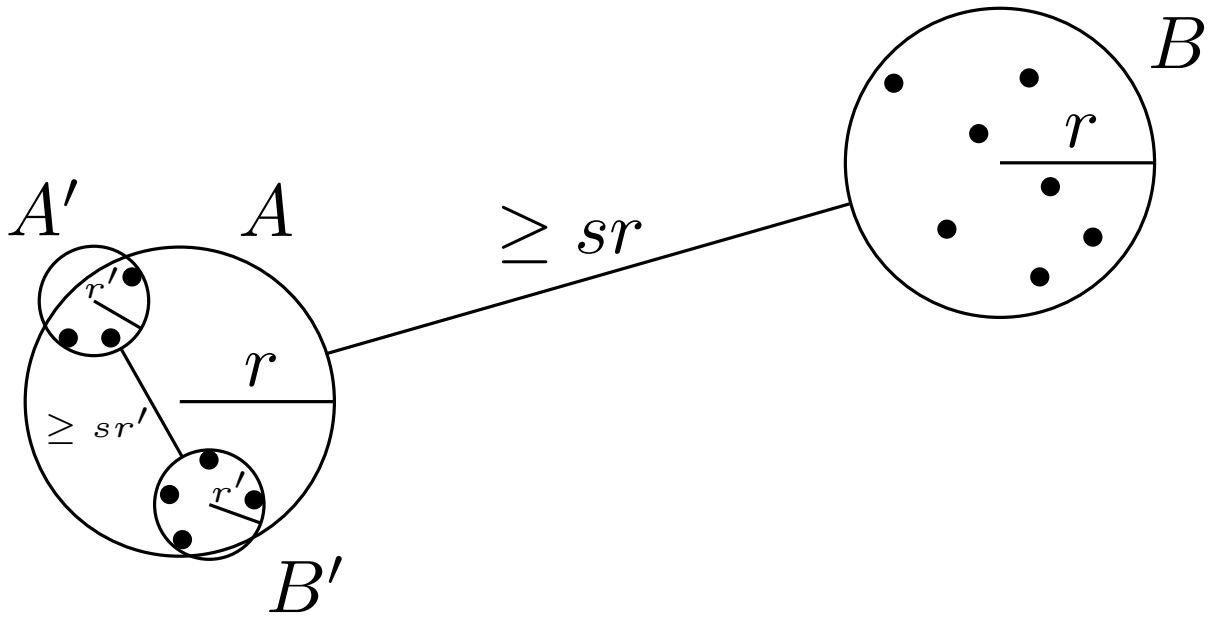
Well-Separated Pairs

Def.: Ein Paar disjunkter Punktmengen A und B im $\mathbb{R}^d$ heißt s-**well separated** für ein $s > 0$, falls A und B jeweils von einer Kugel mit Radius r überdeckt werden und der Abstand der beiden Kugeln mindestens sr ist.



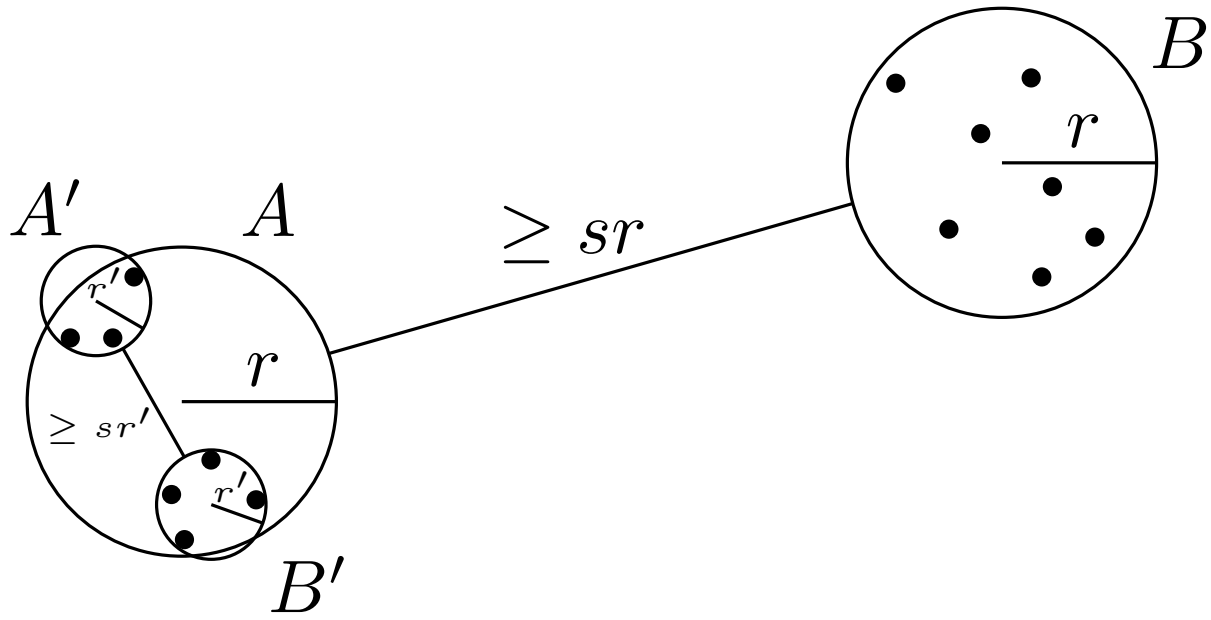
Well-Separated Pairs

Def.: Ein Paar disjunkter Punktmengen A und B im $\mathbb{R}^d$ heißt s-**well separated** für ein $s > 0$, falls A und B jeweils von einer Kugel mit Radius r überdeckt werden und der Abstand der beiden Kugeln mindestens sr ist.



Well-Separated Pairs

Def.: Ein Paar disjunkter Punktmengen A und B im $\mathbb{R}^d$ heißt s-**well separated** für ein $s > 0$, falls A und B jeweils von einer Kugel mit Radius r überdeckt werden und der Abstand der beiden Kugeln mindestens sr ist.



Beob.:

- s-well separated $\Rightarrow$ s'-well separated für alle $s' \leq s$
- Singletons $\{a\}$ und $\{b\}$ sind s-well separated für alle $s > 0$

Well-Separated Pair Decomposition (WSPD)

Karlsruhe Institute of Technology

Für wohlsepariertes Paar $\{A, B\}$ gilt, dass der Abstand für alle Punktpaare in $A \otimes B = \{\{a, b\} \mid a \in A, b \in B, a \neq b\}$ ähnlich ist.

Ziel: $o(n^2)$-Datenstruktur, die den Abstand aller $\binom{n}{2}$ Paare von Punkten einer Menge $P = \{p_1, \ldots, p_n\}$ approximiert.

Well-Separated Pair Decomposition (WSPD)

Karlsruhe Institute of Technology

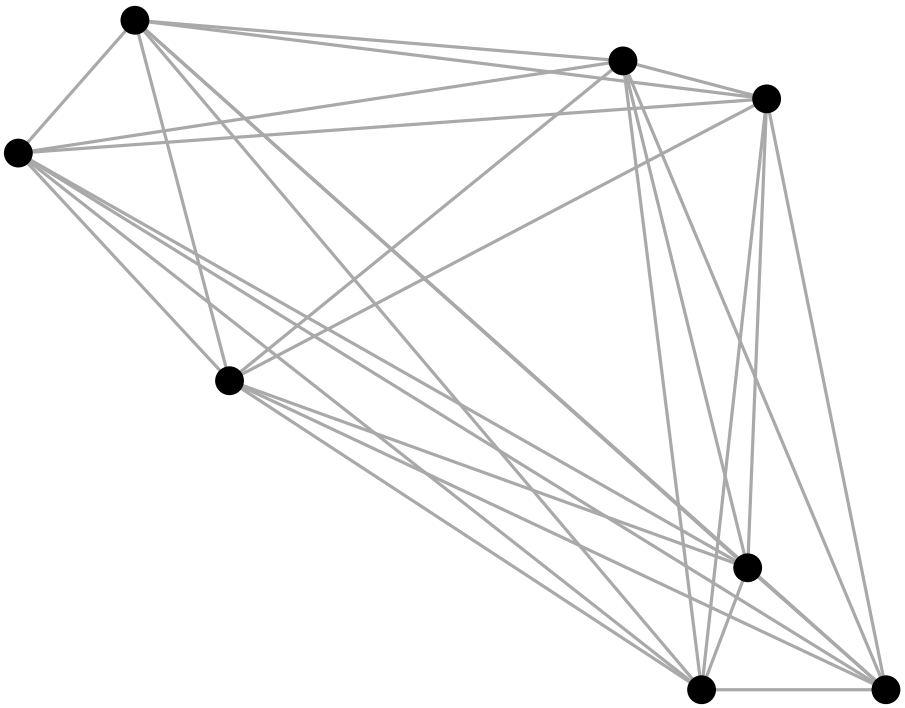
Für wohlsepariertes Paar $\{A, B\}$ gilt, dass der Abstand für alle Punktpaare in $A \otimes B = \{\{a, b\} \mid a \in A, b \in B, a \neq b\}$ ähnlich ist.

Ziel: $o(n^2)$-Datenstruktur, die den Abstand aller $\binom{n}{2}$ Paare von Punkten einer Menge $P = \{p_1, \ldots, p_n\}$ approximiert.

Def.: Für eine Punktmenge P und ein $s > 0$ ist eine s-**well separated pair decomposition** (s-WSPD) eine Menge von Paaren $\{\{A_1, B_1\}, \ldots, \{A_m, B_m\}\}$ mit

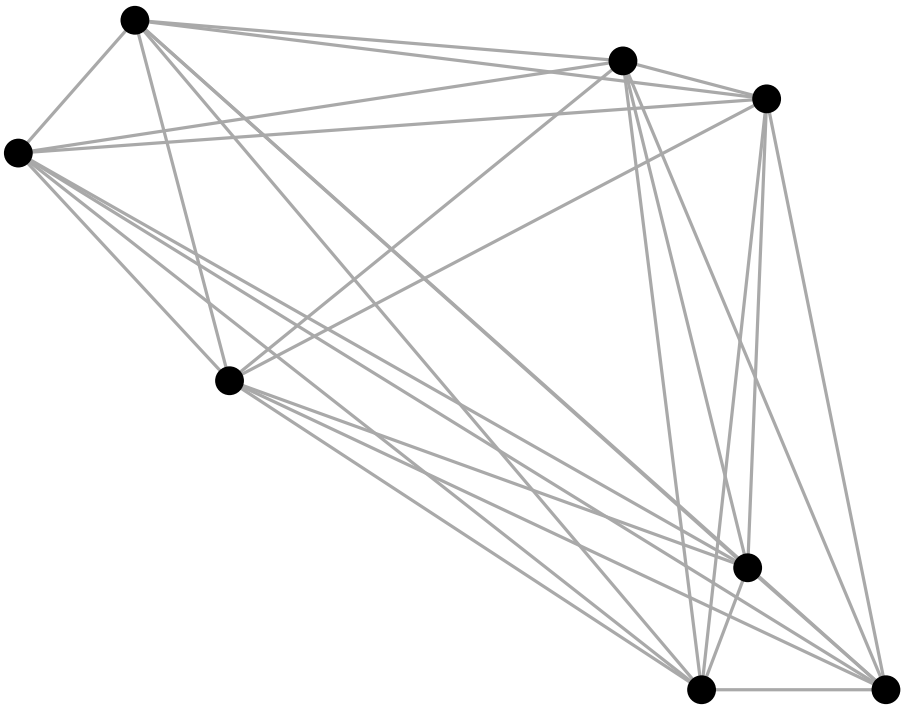
- $A_i, B_i \subset P$ für alle i
- $A_i \cap B_i = \emptyset$ für alle i
- $\bigcup_{i=1}^{m} A_i \otimes B_i = P \otimes P$
- $\{A_i, B_i\}$ s-well separated für alle i

Beispiel

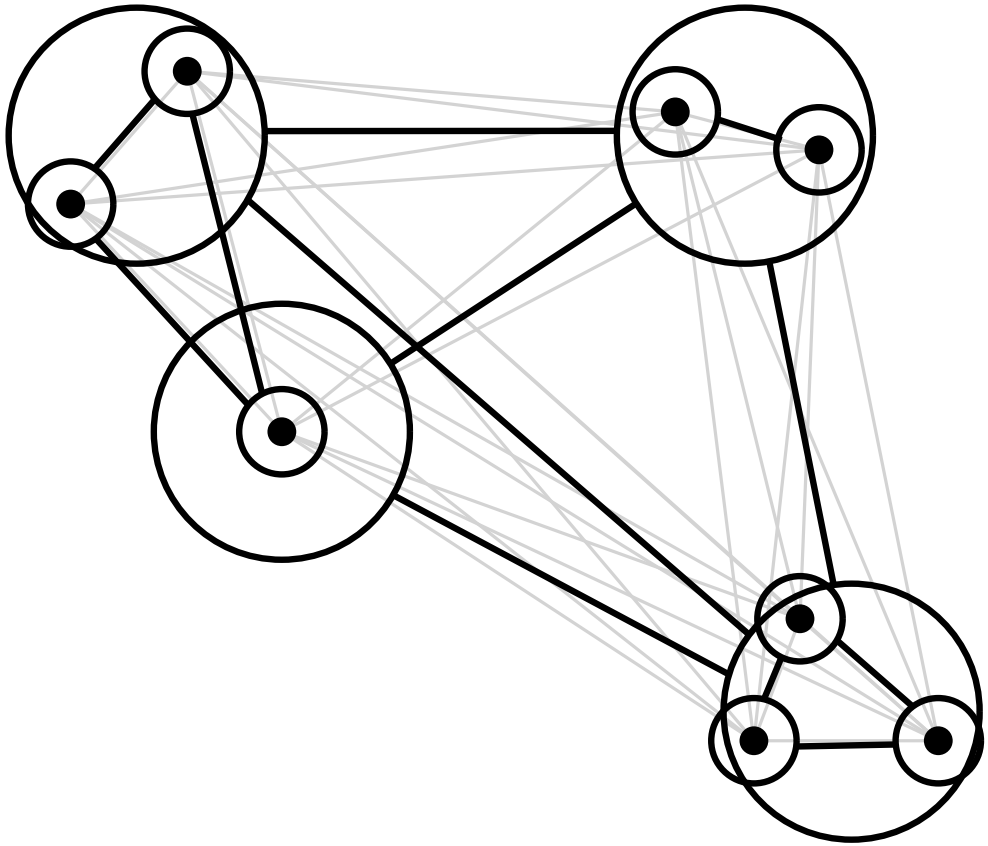


28 Punktpaare

Beispiel

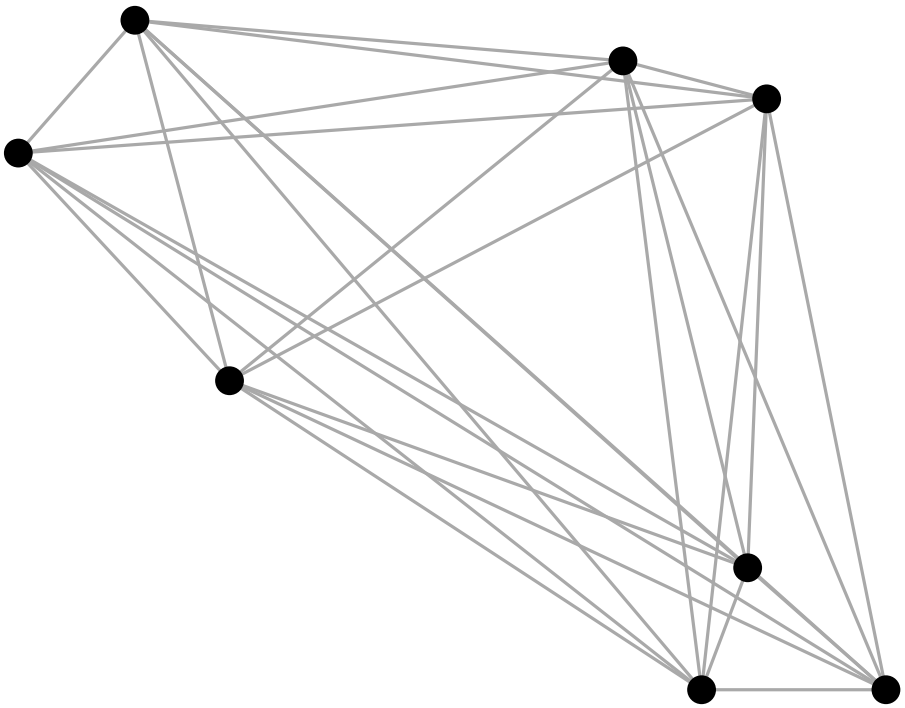


28 Punktpaare

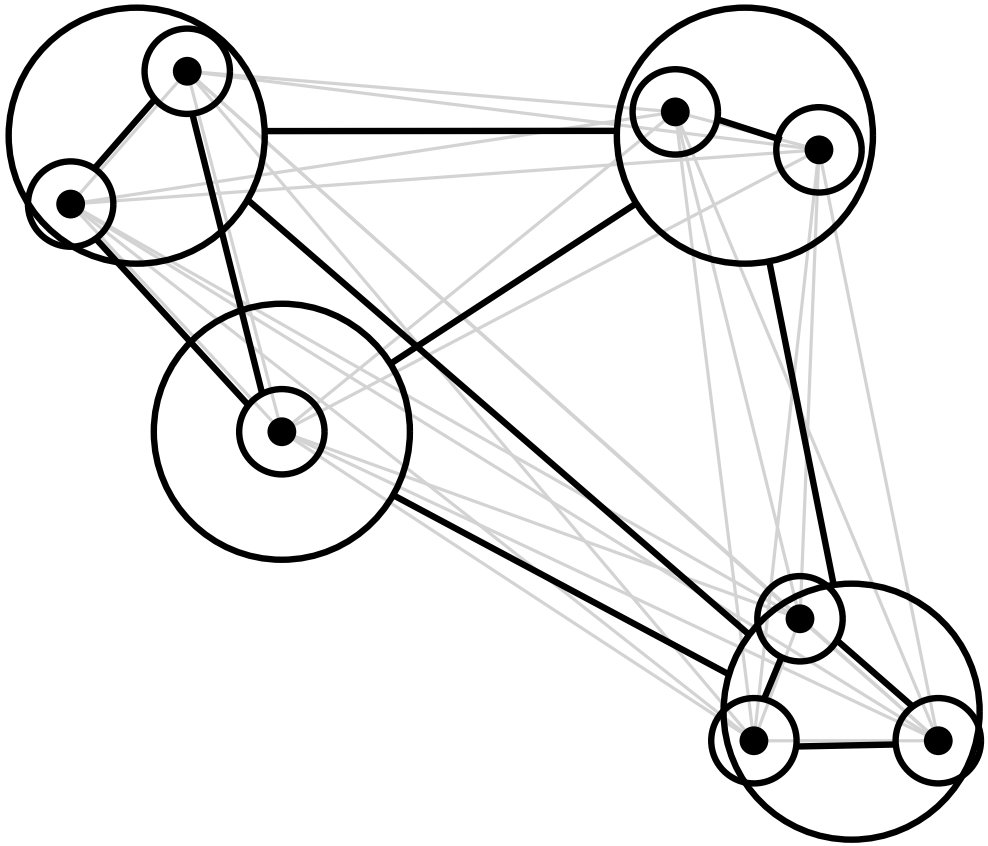


12 s-well separated pairs

Beispiel



28 Punktpaare

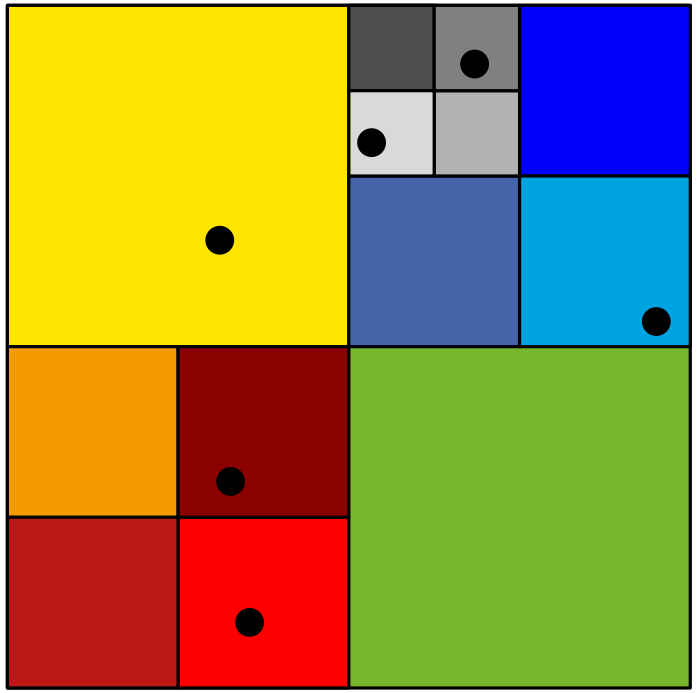


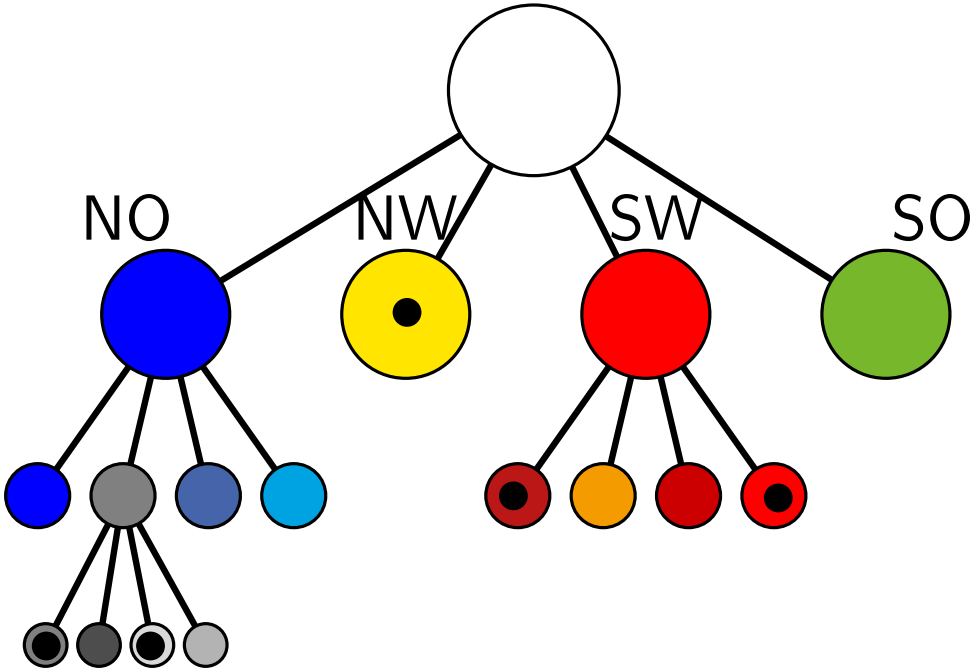
12 s-well separated pairs

WSPD der Größe $O(n^2)$ ist trivial. Geht es auch in $O(n)$?

Wiederholung: Quadtrees

Def.: Ein **Quadtree** $\mathcal{T}(P)$ für eine Punktmenge P ist ein Wurzelbaum, in dem jeder innere Knoten vier Kinder hat. Jeder Knoten entspricht einem Quadrat und die Quadrate der Blätter bilden eine Unterteilung des Wurzelquadrats.





Wiederholung: Quadtrees

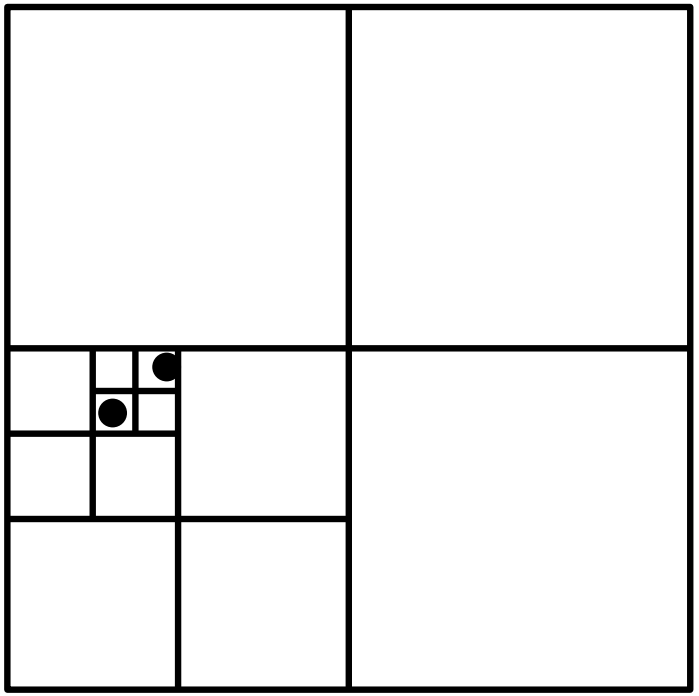
Def.: Ein **Quadtree** $\mathcal{T}(P)$ für eine Punktmenge P ist ein Wurzelbaum, in dem jeder innere Knoten vier Kinder hat. Jeder Knoten entspricht einem Quadrat und die Quadrate der Blätter bilden eine Unterteilung des Wurzelquadrats.

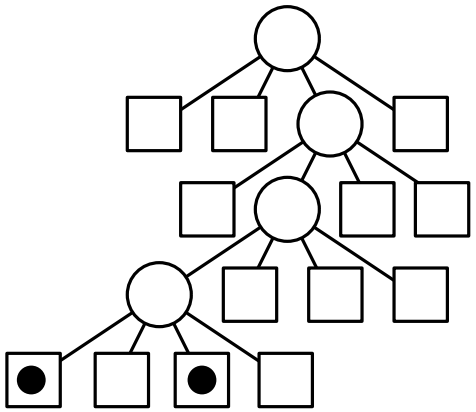
Lemma 1: Die Tiefe d von $\mathcal{T}(P)$ ist höchstens $\log(s/c) + 3/2$, wobei c der kleinste Abstand in P ist und s die Seitenlänge des Wurzelquadrats Q.

Satz 1: Ein Quadtree $\mathcal{T}(P)$ für n Punkte und mit Tiefe d hat $O((d+1)n)$ Knoten und kann in $O((d+1)n)$ Zeit konstruiert werden.

Komprimierte Quadtrees

Def.: Ein **komprimierter** Quadtree ist ein Quadtree, bei dem Pfade nicht-separierender innerer Knoten zu einer Kante komprimiert sind. Die Kante trägt ein Label zur Rekonstruktion der Pfadstruktur.

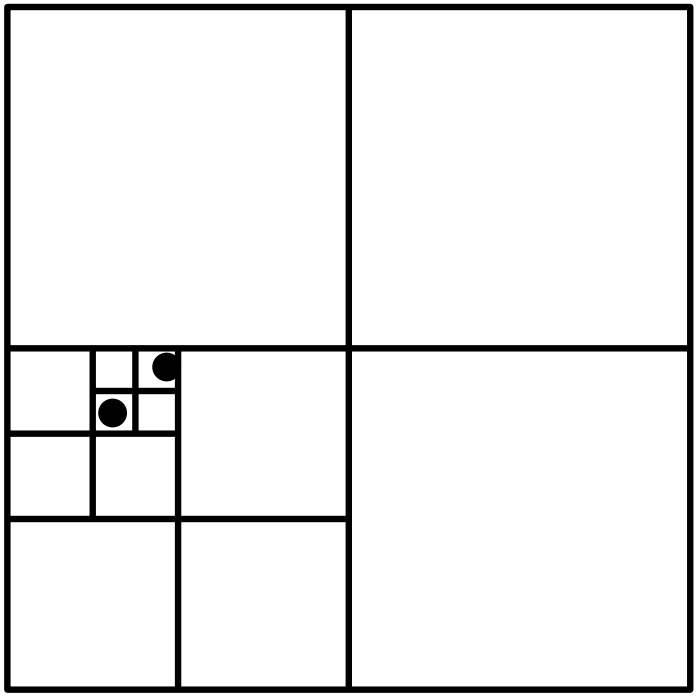




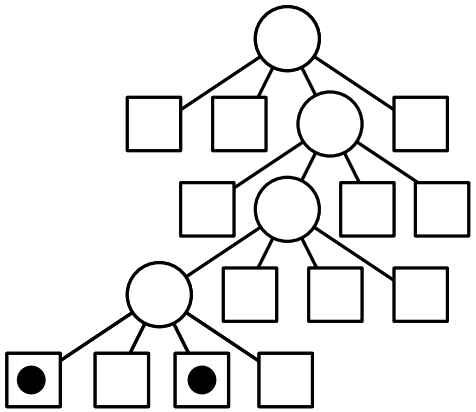
Quadtree

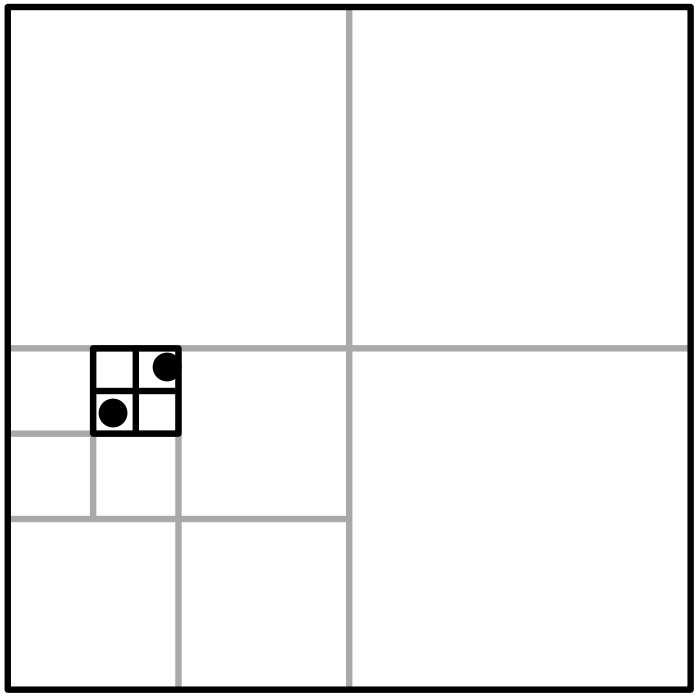
Komprimierte Quadtrees

Def.: Ein **komprimierter** Quadtree ist ein Quadtree, bei dem Pfade nicht-separierender innerer Knoten zu einer Kante komprimiert sind. Die Kante trägt ein Label zur Rekonstruktion der Pfadstruktur.



Quadtree

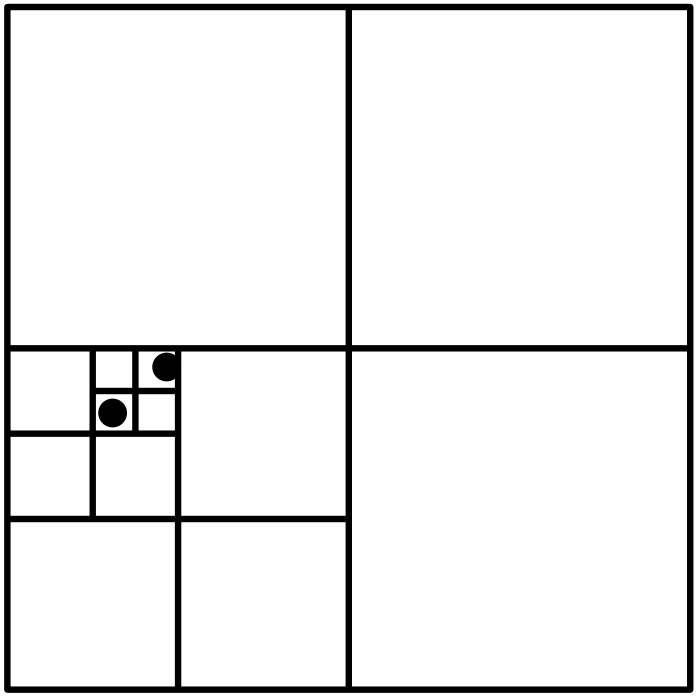




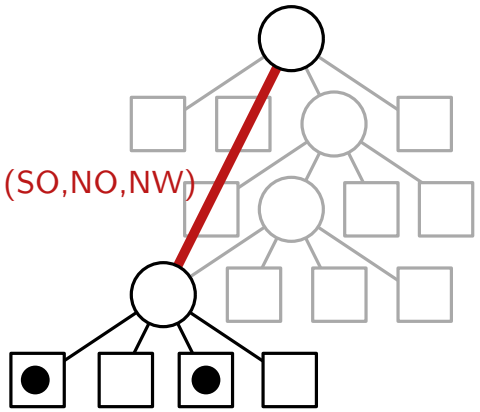
komprimierter Quadtree

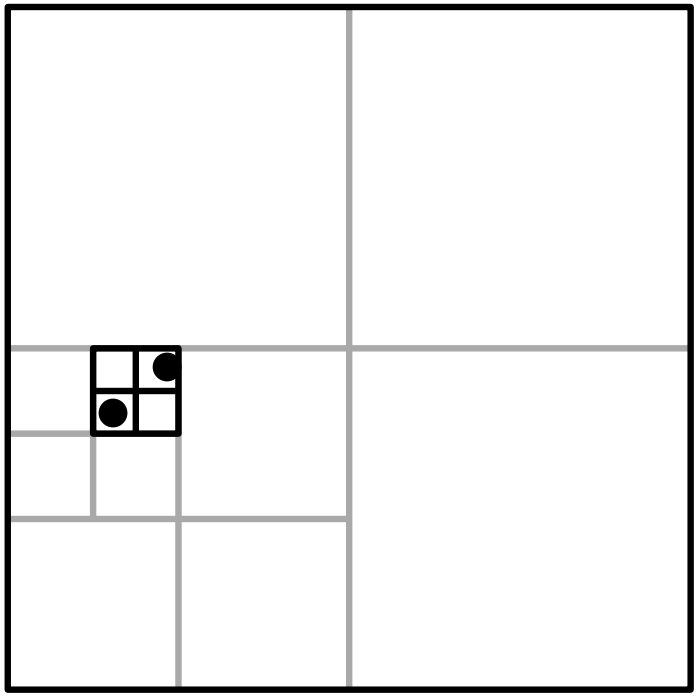
Komprimierte Quadtrees

Def.: Ein **komprimierter** Quadtree ist ein Quadtree, bei dem Pfade nicht-separierender innerer Knoten zu einer Kante komprimiert sind. Die Kante trägt ein Label zur Rekonstruktion der Pfadstruktur.



Quadtree



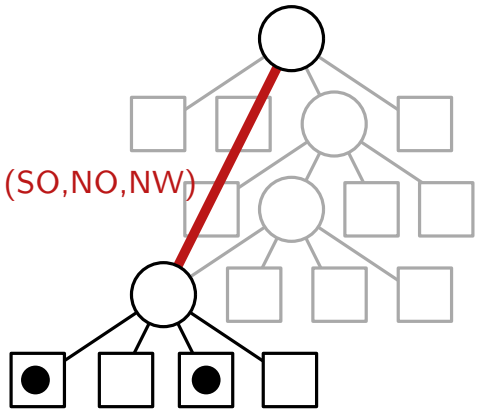


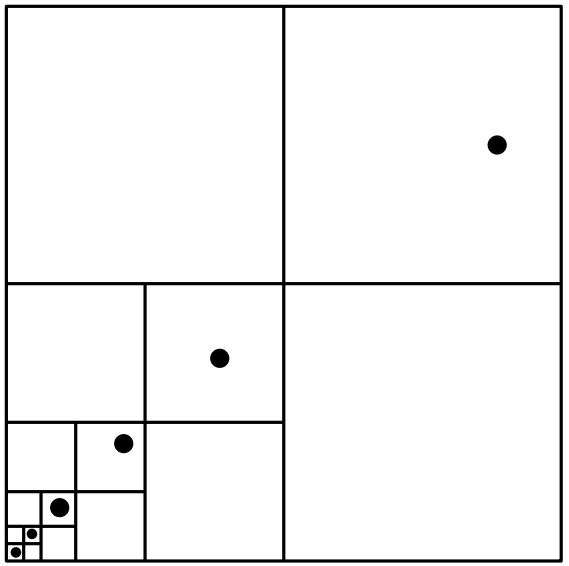
komprimierter Quadtree

Eigenschaften komprimierter Quadtrees

Beob.:

- innere Knoten teilen ihre Punktmenge in ≥ 2 nichtleere Teile $\Rightarrow$ max. $n-1$ innere Knoten
- Tiefe kann $d = n$ sein, so dass der Algorithmus aus VL11 $O(n^2)$ Zeit zur Konstruktion braucht



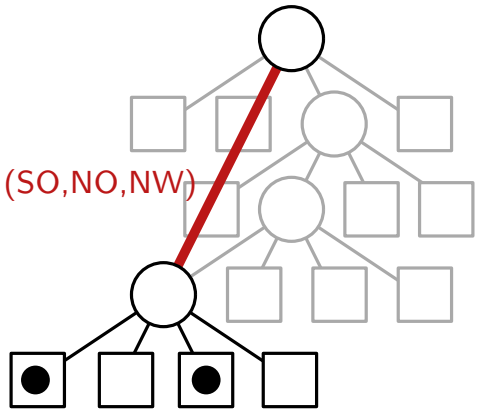


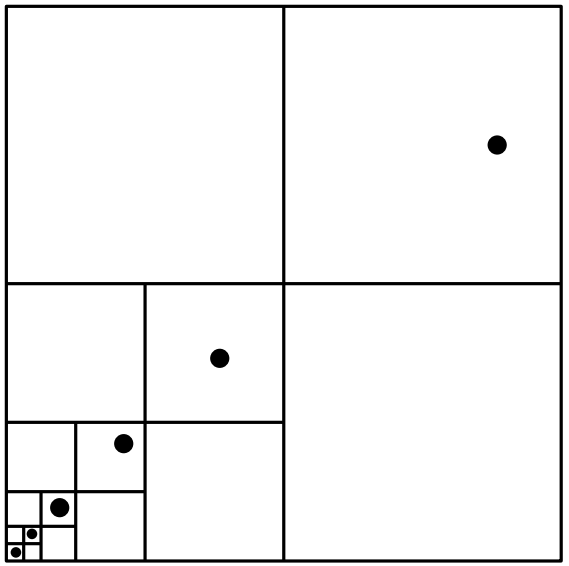
Eigenschaften komprimierter Quadtrees

Beob.:
- innere Knoten teilen ihre Punktmenge in ≥ 2 nichtleere Teile $\Rightarrow$ max. $n-1$ innere Knoten
- Tiefe kann $d = n$ sein, so dass der Algorithmus aus VL11 $O(n^2)$ Zeit zur Konstruktion braucht

Satz: Ein komprimierter Quadtree für n Punkte im $\mathbb{R}^d$ für festes d kann in $O(n \log n)$ Zeit konstruiert werden.

(hier ohne Beweis)





Packungslemma

Lemma 2: Gegeben sei eine Kugel K mit Radius r im $\mathbb{R}^d$ und eine Menge X paarweise disjunkter Quadtree-Zellen mit Seitenlänge $\geq x$, die K schneiden. Dann gilt

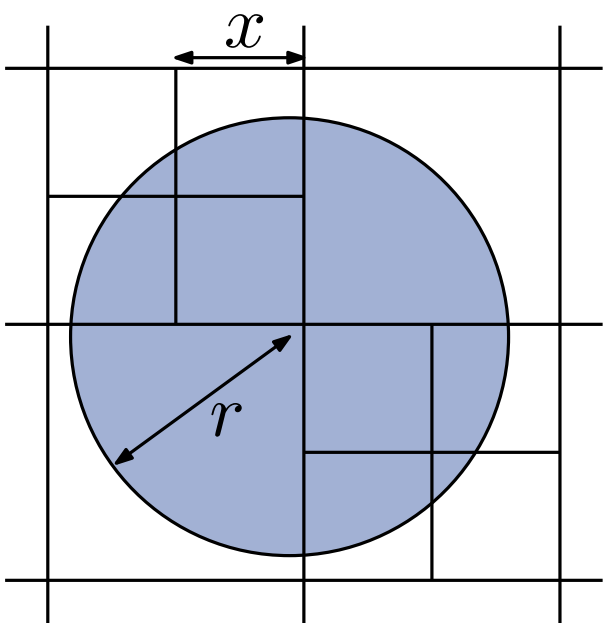
$$|X| \leq (1 + \lceil 2r/x \rceil)^d.$$

Packungslemma

Lemma 2: Gegeben sei eine Kugel K mit Radius r im $\mathbb{R}^d$ und eine Menge X paarweise disjunkter Quadtree-Zellen mit Seitenlänge $\geq x$, die K schneiden. Dann gilt

$$|X| \leq (1 + \lceil 2r/x \rceil)^d.$$

Beweis:

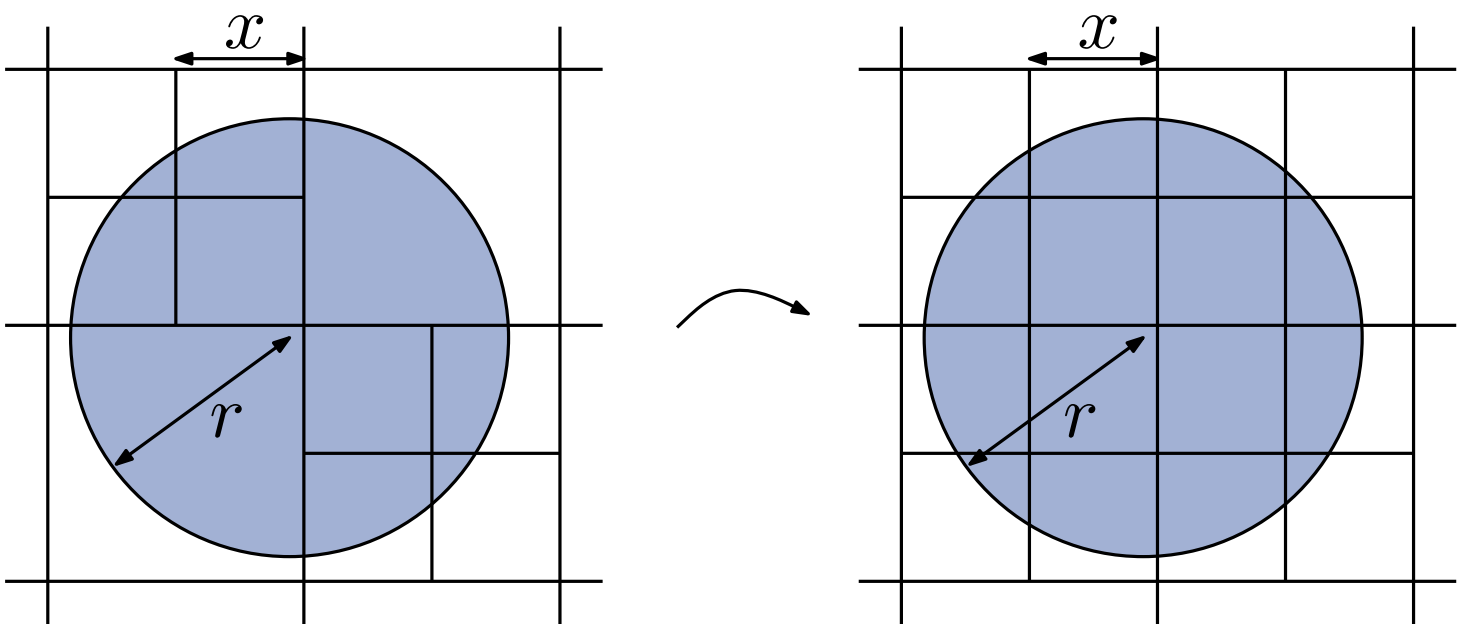


Packungslemma

Lemma 2: Gegeben sei eine Kugel K mit Radius r im $\mathbb{R}^d$ und eine Menge X paarweise disjunkter Quadtree-Zellen mit Seitenlänge $\geq x$, die K schneiden. Dann gilt

$$|X| \leq (1 + \lceil 2r/x \rceil)^d.$$

Beweis:

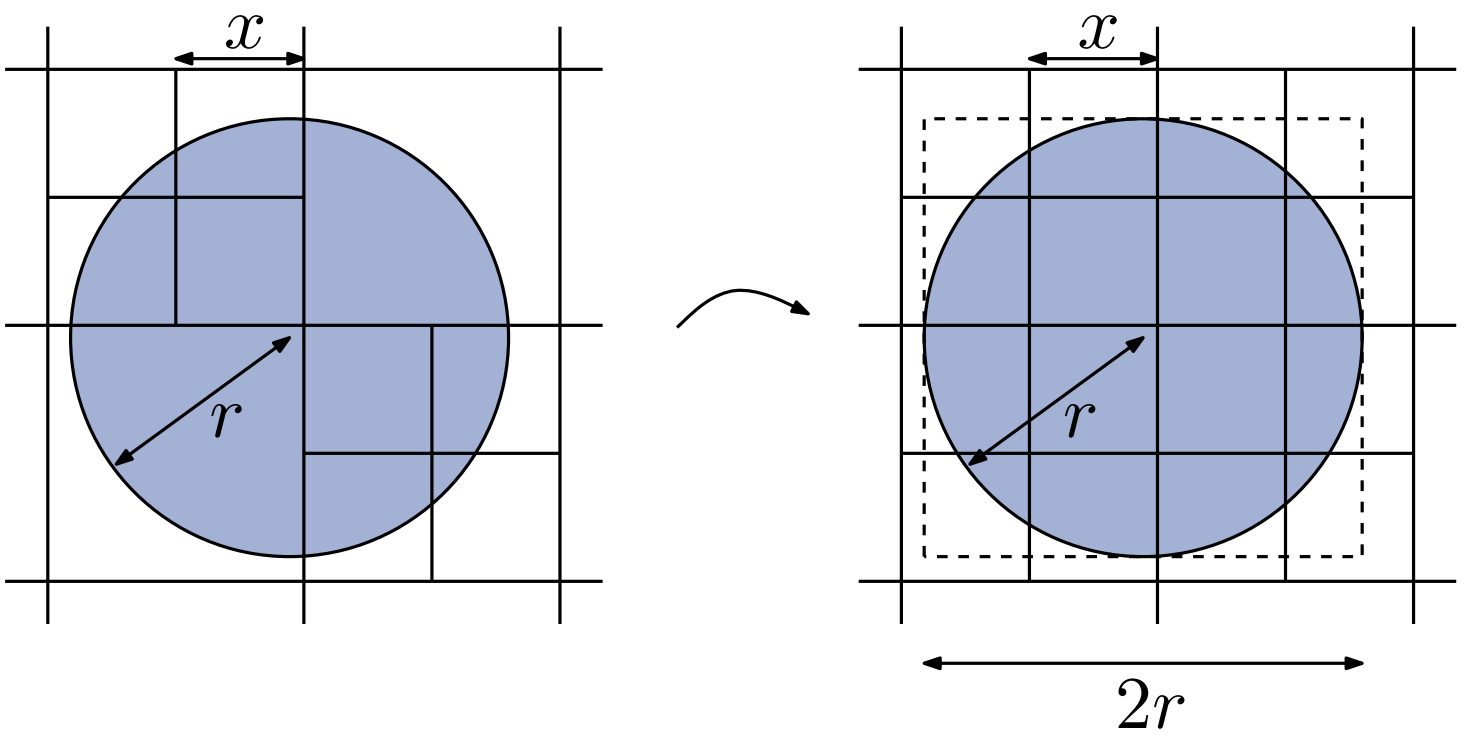


Packungslemma

Lemma 2: Gegeben sei eine Kugel K mit Radius r im $\mathbb{R}^d$ und eine Menge X paarweise disjunkter Quadtree-Zellen mit Seitenlänge $\geq x$, die K schneiden. Dann gilt

$$|X| \leq (1 + \lceil 2r/x \rceil)^d.$$

Beweis:



Repräsentanten und Level

Def.: Für jeden Knoten u eines Quadtrees $\mathcal{T}(P)$ der Punktmenge P sei $P_u = Q_u \cap P$ die Menge der Punkte im zugehörigen Quadrat Q_u. In jedem Blatt u definiere den Repräsentanten

$$\text{rep}(u) = \begin{cases} p & \text{falls } P_u = \{p\} \ (u \text{ ist Blatt}) \\ \emptyset & \text{sonst.} \end{cases}$$

Für einen inneren Knoten v setze $\text{rep}(v) = \text{rep}(u)$ für ein nichtleeres Kind u von v.

Repräsentanten und Level

Def.: Für jeden Knoten u eines Quadtrees $\mathcal{T}(P)$ der Punktmenge P sei $P_u = Q_u \cap P$ die Menge der Punkte im zugehörigen Quadrat Q_u. In jedem Blatt u definiere den Repräsentanten

$$\text{rep}(u) = \begin{cases} p & \text{falls } P_u = \{p\} \ (u \text{ ist Blatt}) \\ \emptyset & \text{sonst.} \end{cases}$$

Für einen inneren Knoten v setze $\text{rep}(v) = \text{rep}(u)$ für ein nichtleeres Kind u von v.

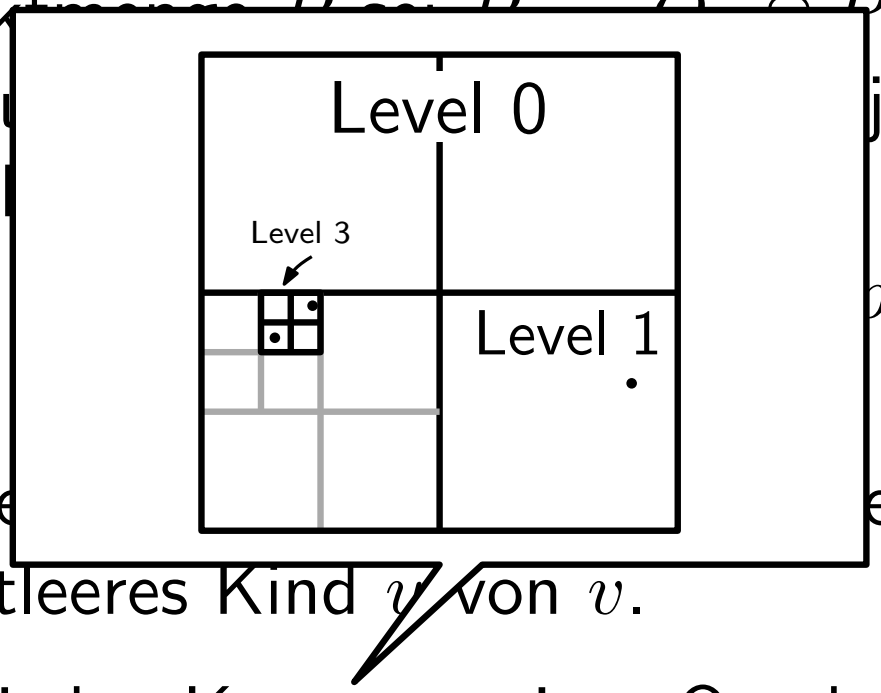
Def.: Für jeden Knoten u eines Quadtrees $\mathcal{T}(P)$ sei $\text{level}(u)$ das Level von u im zugeh. *unkomprimierten* Quadtree. Es gilt $\text{level}(u) \leq \text{level}(v)$ gdw. $\text{Fläche}(Q_u) \geq \text{Fläche}(Q_v)$.

Repräsentanten und Level

Def.: Für jeden Knoten u eines Quadtrees $\mathcal{T}(P)$ der Punktmenge P sei $P_u := Q_u \cap P$ die Menge der Punkte im zugeh. Quadrat Q_u. Für jedem Blatt u definiere den Repräsentanten

$$\text{rep}(u) := \{p \in P_u : p \text{ ist Eckpunkt von } Q_u\} \text{ (u ist Blatt)}$$

Für einen inneren Knoten v sei $\text{rep}(v) = \text{rep}(u)$ für ein nichtleeres Kind u von v.

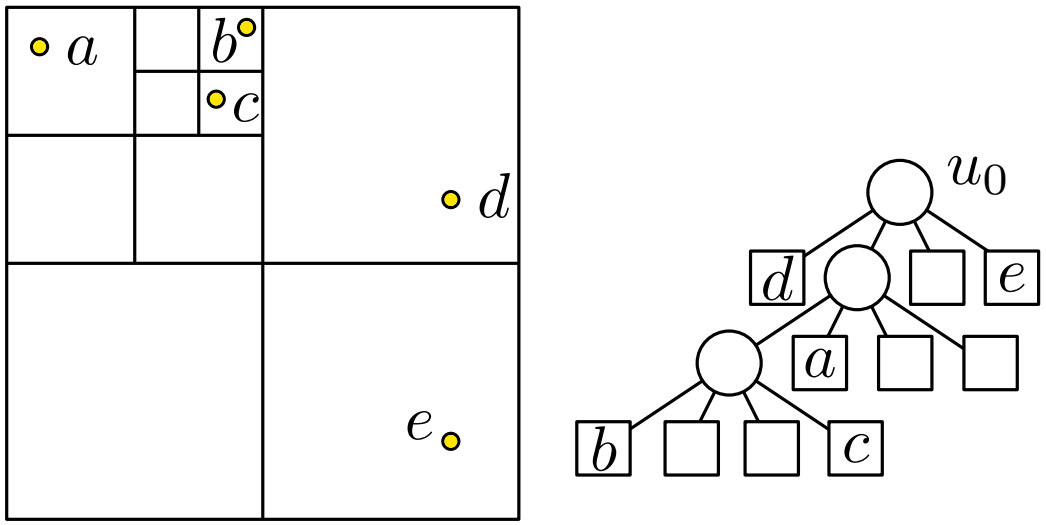


Def.: Für jeden Knoten u eines Quadtrees $\mathcal{T}(P)$ sei $\text{level}(u)$ das Level von u im zugeh. *unkomprimierten* Quadtree. Es gilt $\text{level}(u) \leq \text{level}(v)$ gdw. $\text{Fläche}(Q_u) \geq \text{Fläche}(Q_v)$.

Konstruktion einer WSPD

wsPairs$(u, v, \mathcal{T}, s)$

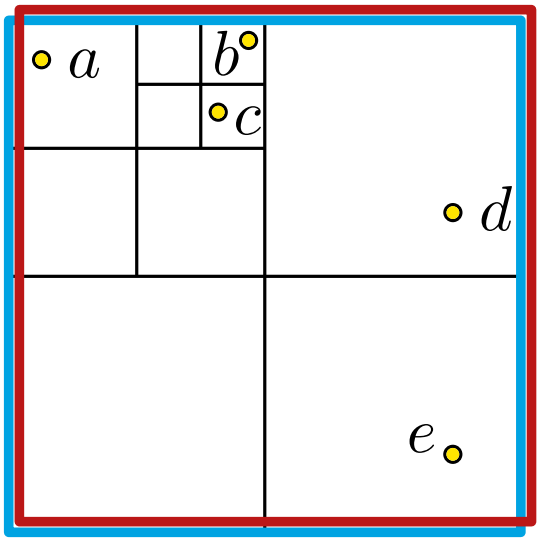
Input: Quadtree-Knoten u, v, Quadtree $\mathcal{T}$, $s > 0$
Output: WSPD für $P_u \otimes P_v$
if rep$(u) = \emptyset$ oder rep$(v) = \emptyset$ oder Blätter $u = v$ **then return** $\emptyset$
else if P_u und P_v s-well separated **then return** $\{\{u, v\}\}$
else
  if level$(u) >$ level(v) **then** tausche u und v
  $(u_1, \ldots, u_m) \leftarrow$ Kinder von u in $\mathcal{T}$
  return $\bigcup_{i=1}^{m}$ wsPairs$(u_i, v, \mathcal{T}, s)$

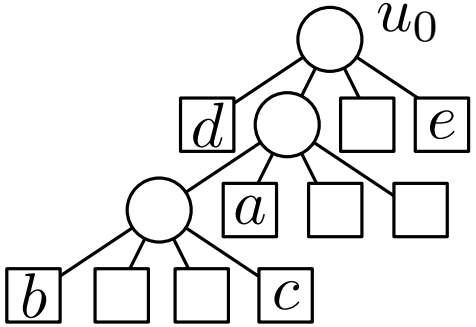


Konstruktion einer WSPD

wsPairs$(u, v, \mathcal{T}, s)$

Input: Quadtree-Knoten u, v, Quadtree $\mathcal{T}$, $s > 0$
Output: WSPD für $P_u \otimes P_v$
if rep$(u) = \emptyset$ oder rep$(v) = \emptyset$ oder Blätter $u = v$ **then return** $\emptyset$
else if P_u und P_v s-well separated **then return** $\{\{u, v\}\}$
else
  if level$(u) >$ level(v) **then** tausche u und v
  $(u_1, \ldots, u_m) \leftarrow$ Kinder von u in $\mathcal{T}$
  return $\bigcup_{i=1}^{m}$ wsPairs$(u_i, v, \mathcal{T}, s)$

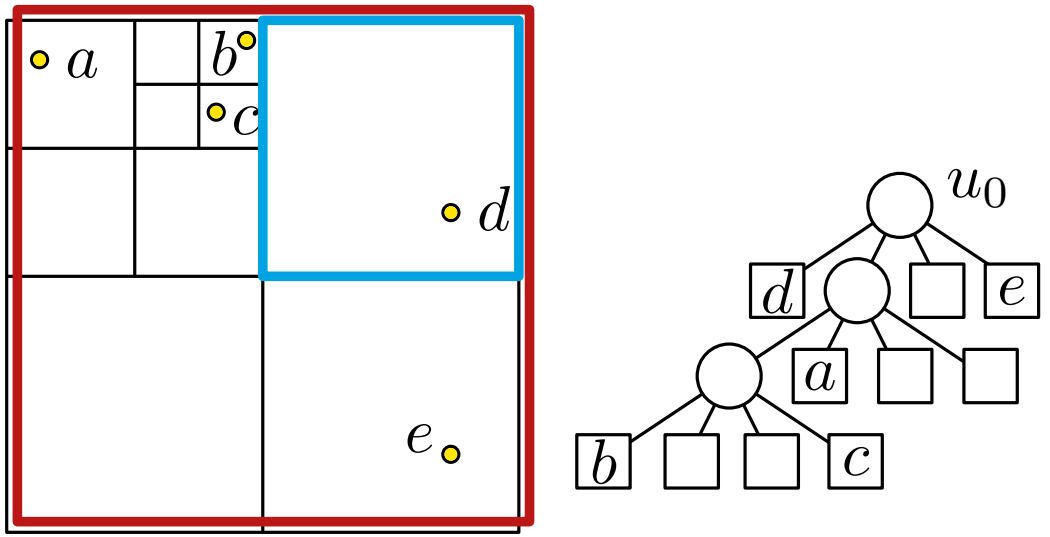




Konstruktion einer WSPD

wsPairs$(u, v, \mathcal{T}, s)$

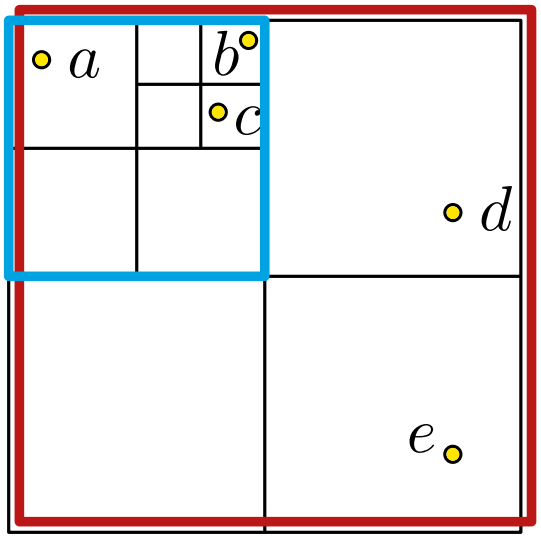
Input: Quadtree-Knoten u, v, Quadtree $\mathcal{T}$, $s > 0$
Output: WSPD für $P_u \otimes P_v$
if rep$(u) = \emptyset$ oder rep$(v) = \emptyset$ oder Blätter $u = v$ **then return** $\emptyset$
else if P_u und P_v s-well separated **then return** $\{\{u, v\}\}$
else
  if level$(u) >$ level(v) **then** tausche u und v
  $(u_1, \ldots, u_m) \leftarrow$ Kinder von u in $\mathcal{T}$
  return $\bigcup_{i=1}^{m}$ wsPairs$(u_i, v, \mathcal{T}, s)$

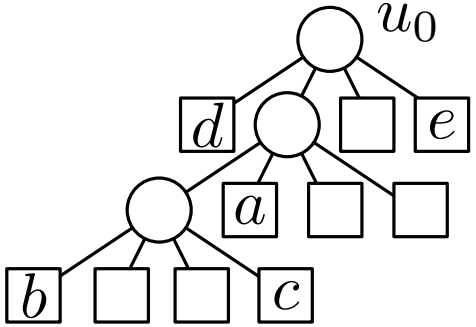


Konstruktion einer WSPD

wsPairs$(u, v, \mathcal{T}, s)$

Input: Quadtree-Knoten u, v, Quadtree $\mathcal{T}$, $s > 0$
Output: WSPD für $P_u \otimes P_v$
if rep$(u) = \emptyset$ oder rep$(v) = \emptyset$ oder Blätter $u = v$ **then return** $\emptyset$
else if P_u und P_v s-well separated **then return** $\{\{u, v\}\}$
else
  if level$(u) >$ level(v) **then** tausche u und v
  $(u_1, \ldots, u_m) \leftarrow$ Kinder von u in $\mathcal{T}$
  return $\bigcup_{i=1}^{m}$ wsPairs$(u_i, v, \mathcal{T}, s)$

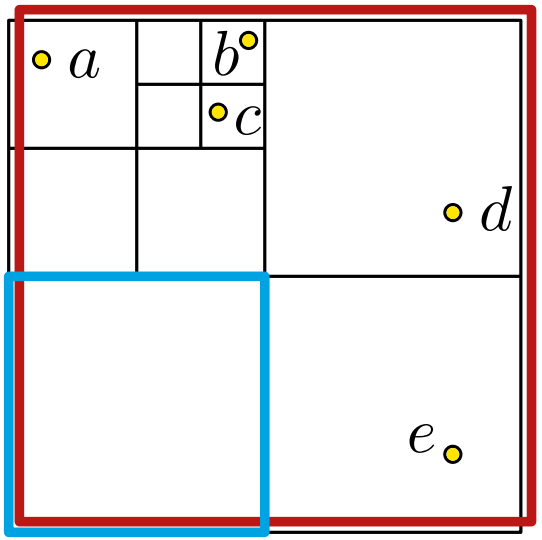


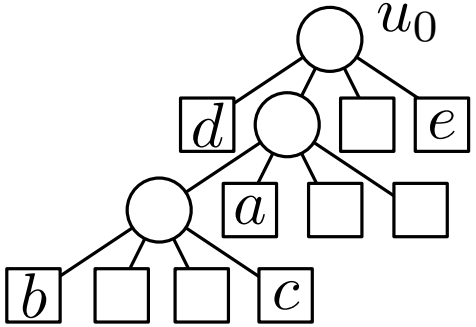


Konstruktion einer WSPD

wsPairs$(u, v, \mathcal{T}, s)$

Input: Quadtree-Knoten u, v, Quadtree $\mathcal{T}$, $s > 0$
Output: WSPD für $P_u \otimes P_v$
if rep$(u) = \emptyset$ oder rep$(v) = \emptyset$ oder Blätter $u = v$ **then return** $\emptyset$
else if P_u und P_v s-well separated **then return** $\{\{u, v\}\}$
else
  if level$(u) >$ level(v) **then** tausche u und v
  $(u_1, \ldots, u_m) \leftarrow$ Kinder von u in $\mathcal{T}$
  return $\bigcup_{i=1}^{m}$ wsPairs$(u_i, v, \mathcal{T}, s)$

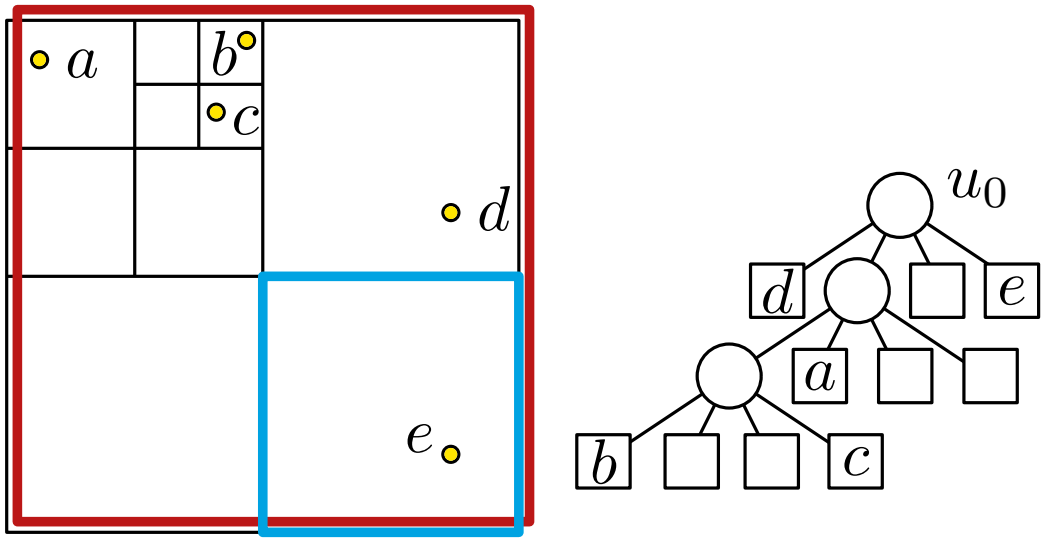




Konstruktion einer WSPD

wsPairs$(u, v, \mathcal{T}, s)$

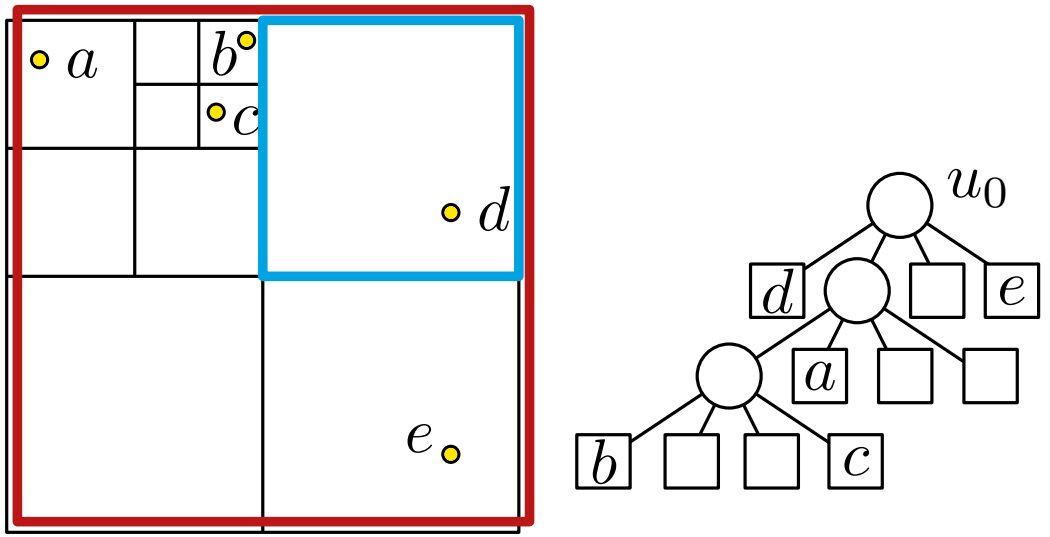
Input: Quadtree-Knoten u, v, Quadtree $\mathcal{T}$, $s > 0$
Output: WSPD für $P_u \otimes P_v$
if rep$(u) = \emptyset$ oder rep$(v) = \emptyset$ oder Blätter $u = v$ **then return** $\emptyset$
else if P_u und P_v s-well separated **then return** $\{\{u, v\}\}$
else
  if level$(u) >$ level(v) **then** tausche u und v
  $(u_1, \ldots, u_m) \leftarrow$ Kinder von u in $\mathcal{T}$
  return $\bigcup_{i=1}^{m}$ wsPairs$(u_i, v, \mathcal{T}, s)$



Konstruktion einer WSPD

wsPairs$(u, v, \mathcal{T}, s)$

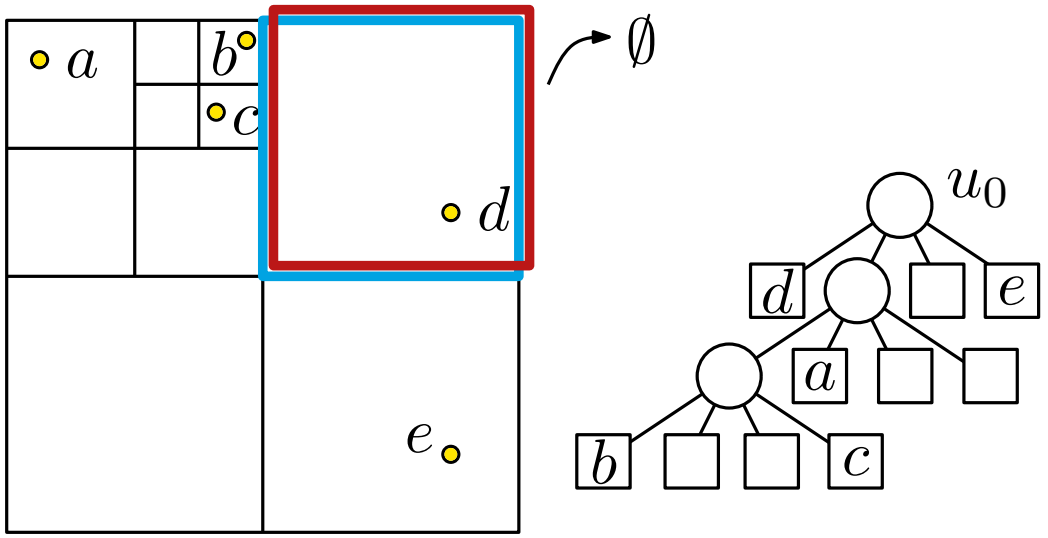
Input: Quadtree-Knoten u, v, Quadtree $\mathcal{T}$, $s > 0$
Output: WSPD für $P_u \otimes P_v$
if rep$(u) = \emptyset$ oder rep$(v) = \emptyset$ oder Blätter $u = v$ **then return** $\emptyset$
else if P_u und P_v s-well separated **then return** $\{\{u, v\}\}$
else
  if level$(u) >$ level(v) **then** tausche u und v
  $(u_1, \ldots, u_m) \leftarrow$ Kinder von u in $\mathcal{T}$
  return $\bigcup_{i=1}^{m}$ wsPairs$(u_i, v, \mathcal{T}, s)$



Konstruktion einer WSPD

wsPairs$(u, v, \mathcal{T}, s)$

Input: Quadtree-Knoten u, v, Quadtree $\mathcal{T}$, $s > 0$
Output: WSPD für $P_u \otimes P_v$
if rep$(u) = \emptyset$ oder rep$(v) = \emptyset$ oder Blätter $u = v$ **then return** $\emptyset$
else if P_u und P_v s-well separated **then return** $\{\{u, v\}\}$
else
  if level$(u) >$ level(v) **then** tausche u und v
  $(u_1, \ldots, u_m) \leftarrow$ Kinder von u in $\mathcal{T}$
  return $\bigcup_{i=1}^{m}$ wsPairs$(u_i, v, \mathcal{T}, s)$



Konstruktion einer WSPD

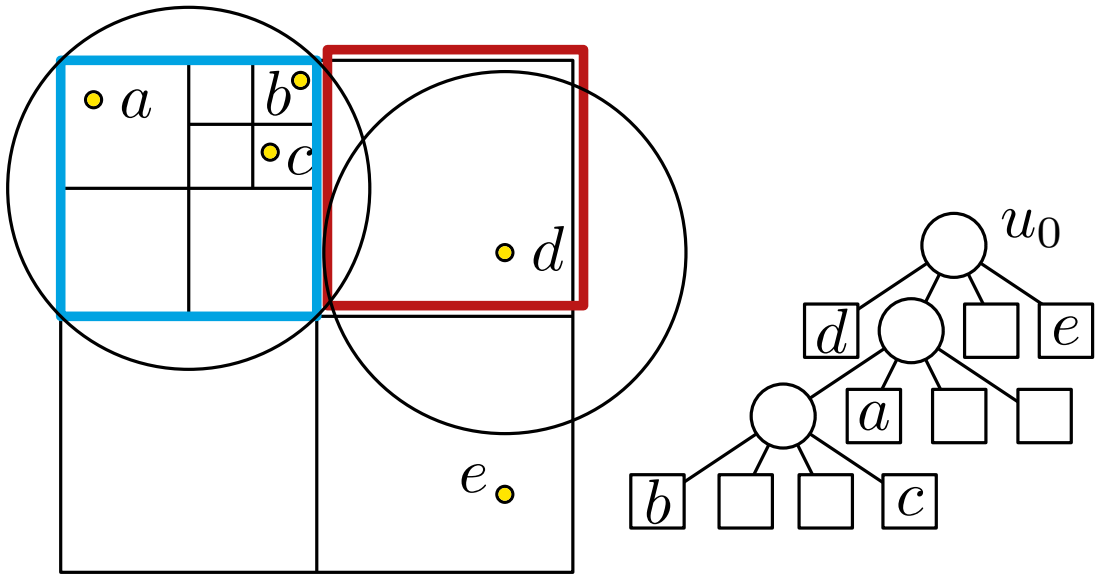
wsPairs$(u, v, \mathcal{T}, s)$

Input: Quadt

Output: WS

Kreise um Q_u und Q_v (bzw. Radius 0 für Punkt in Blatt)
vergrößere kleineren Kreis und prüfe Abstand $\geq sr$

if rep$(u) = \emptyset$ oder rep$(v) = \emptyset$ oder Blätter $u = v$ **then return** $\emptyset$
else if P_u und P_v s-well separated **then return** $\{\{u, v\}\}$
else
  if level$(u) >$ level(v) **then** tausche u und v
  $(u_1, \ldots, u_m) \leftarrow$ Kinder von u in $\mathcal{T}$
  return $\bigcup_{i=1}^{m}$ wsPairs$(u_i, v, \mathcal{T}, s)$



Konstruktion einer WSPD

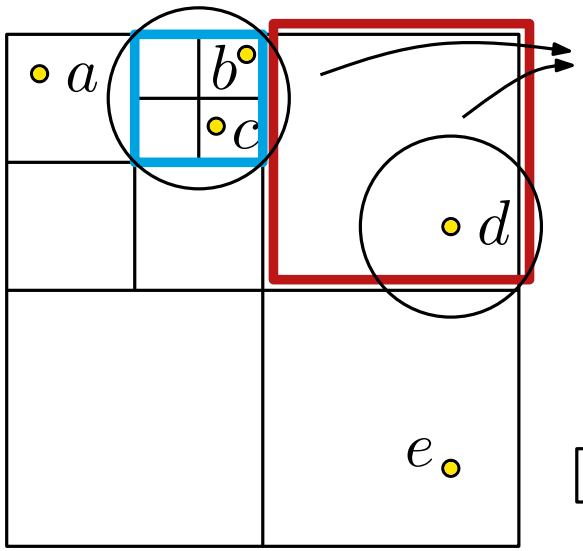
wsPairs$(u, v, \mathcal{T}, s)$

Input: Quadt

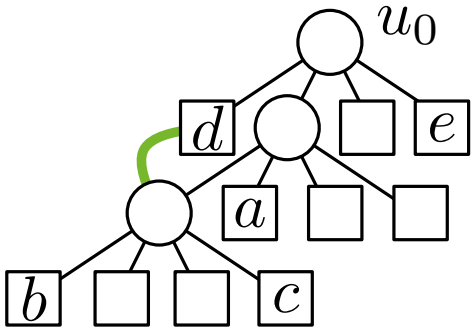
Output: WS

Kreise um Q_u und Q_v (bzw. Radius 0 für Punkt in Blatt)
vergrößere kleineren Kreis und prüfe Abstand $\geq sr$

if rep$(u) = \emptyset$ oder rep$(v) = \emptyset$ oder Blätter $u = v$ **then return** $\emptyset$
else if P_u und P_v s-well separated **then return** $\{\{u, v\}\}$
else
  if level$(u) >$ level(v) **then** tausche u und v
  $(u_1, \ldots, u_m) \leftarrow$ Kinder von u in $\mathcal{T}$
  return $\bigcup_{i=1}^{m}$ wsPairs$(u_i, v, \mathcal{T}, s)$



$\{\{b, c\}, \{d\}\}$

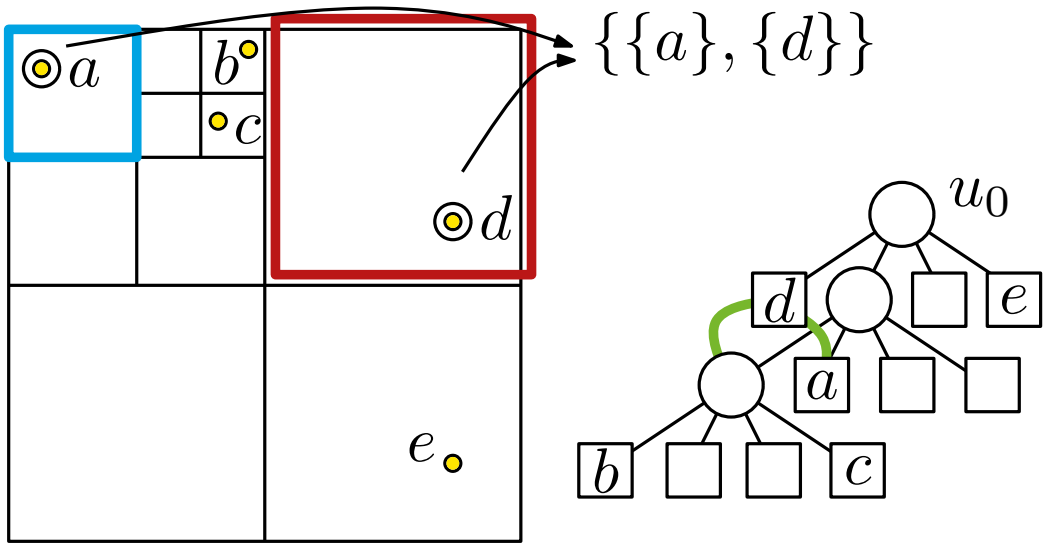


$\{\{b, c\}, \{d\}\}$

Konstruktion einer WSPD

wsPairs$(u, v, \mathcal{T}, s)$

Input: Quadtree-Knoten u, v, Quadtree $\mathcal{T}$, $s > 0$
Output: WSPD für $P_u \otimes P_v$
if rep$(u) = \emptyset$ oder rep$(v) = \emptyset$ oder Blätter $u = v$ **then return** $\emptyset$
else if P_u und P_v s-well separated **then return** $\{\{u, v\}\}$
else
  if level$(u) >$ level(v) **then** tausche u und v
  $(u_1, \ldots, u_m) \leftarrow$ Kinder von u in $\mathcal{T}$
  return $\bigcup_{i=1}^{m}$ wsPairs$(u_i, v, \mathcal{T}, s)$

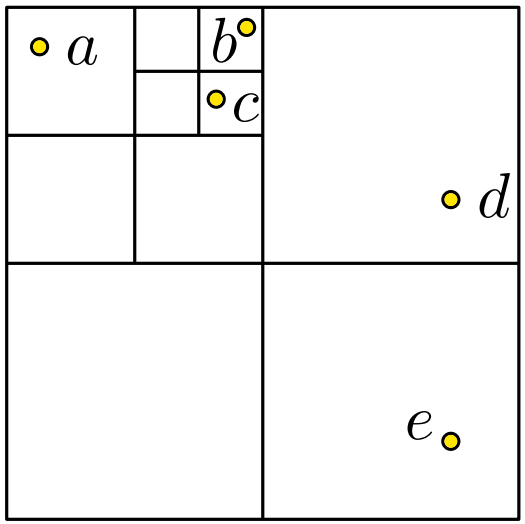


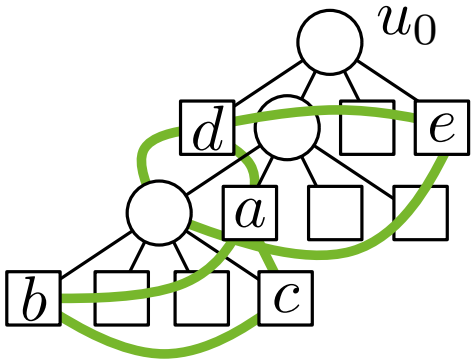
$\{\{b, c\}, \{d\}\}$
$\{\{a\}, \{d\}\}$

Konstruktion einer WSPD

wsPairs$(u, v, \mathcal{T}, s)$

Input: Quadtree-Knoten u, v, Quadtree $\mathcal{T}$, $s > 0$
Output: WSPD für $P_u \otimes P_v$
if rep$(u) = \emptyset$ oder rep$(v) = \emptyset$ oder Blätter $u = v$ **then return** $\emptyset$
else if P_u und P_v s-well separated **then return** $\{\{u, v\}\}$
else
  if level$(u) >$ level(v) **then** tausche u und v
  $(u_1, \ldots, u_m) \leftarrow$ Kinder von u in $\mathcal{T}$
  return $\bigcup_{i=1}^{m}$ wsPairs$(u_i, v, \mathcal{T}, s)$





$\{\{b, c\}, \{d\}\}$
$\{\{a\}, \{d\}\}$
$\{\{b, c\}, \{e\}\}$
$\{\{d\}, \{e\}\}$
$\{\{a\}, \{b\}\}$
$\{\{a\}, \{c\}\}$
$\{\{b\}, \{c\}\}$

Konstruktion einer WSPD

wsPairs$(u, v, \mathcal{T}, s)$

Input: Quadtree-Knoten u, v, Quadtree $\mathcal{T}$, $s > 0$
Output: WSPD für $P_u \otimes P_v$
if rep$(u) = \emptyset$ oder rep$(v) = \emptyset$ oder Blätter $u = v$ **then return** $\emptyset$
else if P_u und P_v s-well separated **then return** $\{\{u, v\}\}$
else
  if level$(u) >$ level(v) **then** tausche u und v
  $(u_1, \ldots, u_m) \leftarrow$ Kinder von u in $\mathcal{T}$
  return $\bigcup_{i=1}^{m}$ wsPairs$(u_i, v, \mathcal{T}, s)$

- initialer Aufruf wsPairs$(u_0, u_0, \mathcal{T}, s)$ Wie?
- Duplikate wsPairs$(u_i, u_j, \mathcal{T}, s)$ und wsPairs$(u_j, u_i, \mathcal{T}, s)$ vermeiden
- Blattpaare sind immer s-well separated, Algorithmus terminiert also
- Ausgabe sind Paare von Quadtree-Knoten Speicherplatz?

Konstruktion einer WSPD

wsPairs$(u, v, \mathcal{T}, s)$

Input: Quadtree-Knoten u, v, Quadtree $\mathcal{T}$, $s > 0$
Output: WSPD für $P_u \otimes P_v$
if rep$(u) = \emptyset$ oder rep$(v) = \emptyset$ oder Blätter $u = v$ **then return** $\emptyset$
else if P_u und P_v s-well separated **then return** $\{\{u, v\}\}$
else
  if level$(u) >$ level(v) **then** tausche u und v
  $(u_1, \ldots, u_m) \leftarrow$ Kinder von u in $\mathcal{T}$
  return $\bigcup_{i=1}^{m}$ wsPairs$(u_i, v, \mathcal{T}, s)$

- initialer Aufruf wsPairs$(u_0, u_0, \mathcal{T}, s)$ Wie?
- Duplikate wsPairs$(u_i, u_j, \mathcal{T}, s)$ und wsPairs$(u_j, u_i, \mathcal{T}, s)$ vermeiden
- Blattpaare sind immer s-well separated, Algorithmus terminiert also
- Ausgabe sind Paare von Quadtree-Knoten Speicherplatz?

Frage: Wie viele Paare erzeugt der Algorithmus?

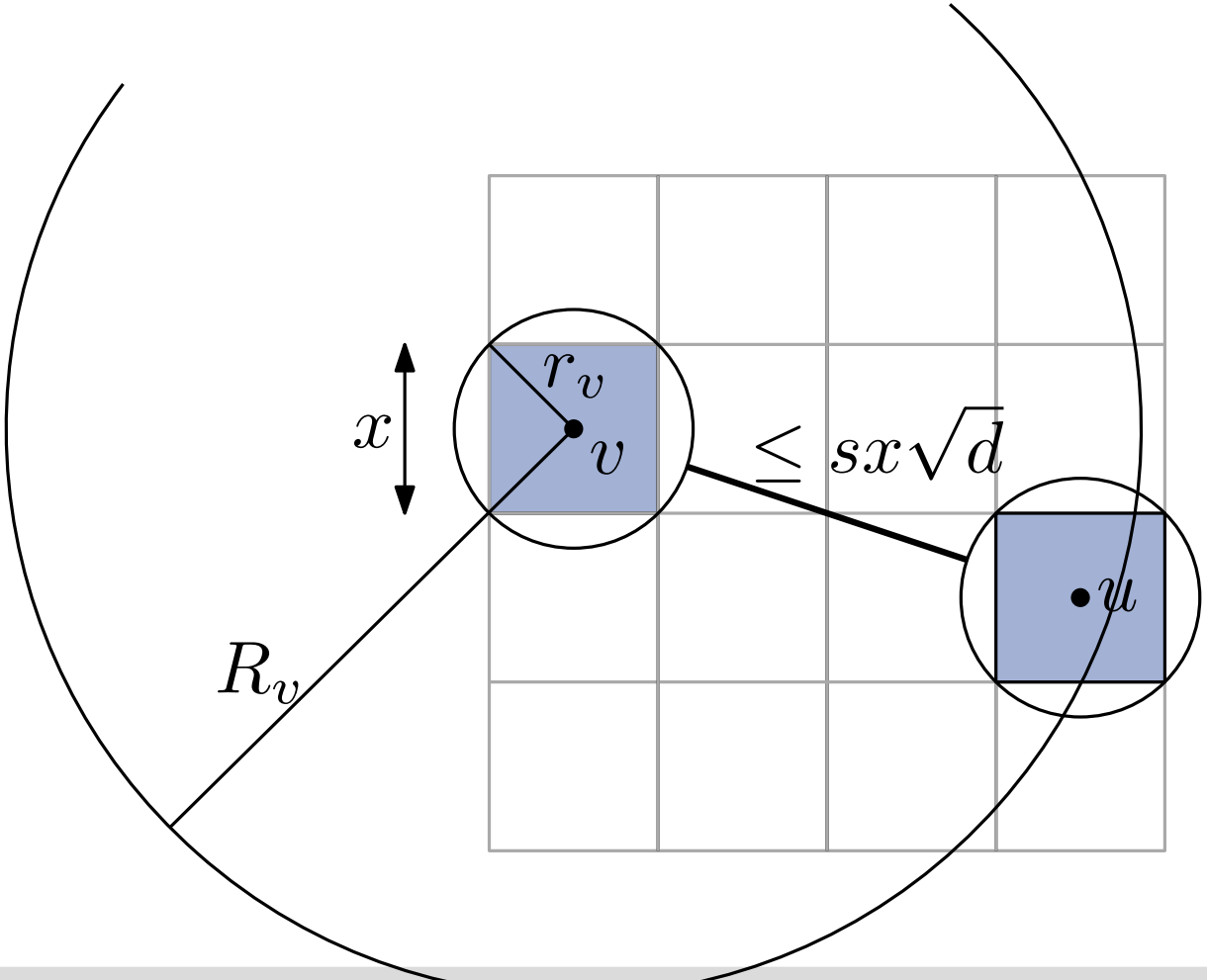
Analyse WSPD-Konstruktion

Satz 2: Gegeben eine Punktmenge P im $\mathbb{R}^d$ und $s \geq 1$ lässt sich eine s-WSPD mit $O(s^d n)$ Paaren in Zeit $O(n \log n + s^d n)$ konstruieren.

Analyse WSPD-Konstruktion

Satz 2: Gegeben eine Punktmenge P im $\mathbb{R}^d$ und $s \geq 1$ lässt sich eine s-WSPD mit $O(s^d n)$ Paaren in Zeit $O(n \log n + s^d n)$ konstruieren.

Beweisskizze:



t-Spanner

Karlsruhe Institute of Technology

Für eine Menge P von n Punkten in $\mathbb{R}^d$ ist der **Euklidische Graph** $\mathcal{EG}(P)$ der vollständige gewichtete Graph, dessen Knotenmenge P ist und dessen Kantengewichte dem Euklidischen Abstand der Kantenendpunkte entsprechen.

t-Spanner

Karlsruhe Institute of Technology

Für eine Menge P von n Punkten in $\mathbb{R}^d$ ist der **Euklidische Graph** $\mathcal{EG}(P)$ der vollständige gewichtete Graph, dessen Knotenmenge P ist und dessen Kantengewichte dem Euklidischen Abstand der Kantenendpunkte entsprechen.

Da $\mathcal{EG}(P)$ $\Theta(n^2)$ Kanten hat, wird oft ein dünner Graph mit $O(n)$ Kanten gesucht, dessen kürzeste Wege die Kantengewichte in $\mathcal{EG}(P)$ approximieren.

t-Spanner

Karlsruhe Institute of Technology

Für eine Menge P von n Punkten in $\mathbb{R}^d$ ist der **Euklidische Graph** $\mathcal{EG}(P)$ der vollständige gewichtete Graph, dessen Knotenmenge P ist und dessen Kantengewichte dem Euklidischen Abstand der Kantenendpunkte entsprechen.

Da $\mathcal{EG}(P)$ $\Theta(n^2)$ Kanten hat, wird oft ein dünner Graph mit $O(n)$ Kanten gesucht, dessen kürzeste Wege die Kantengewichte in $\mathcal{EG}(P)$ approximieren.

Def.: Ein gewichteter Graph G mit Knotenmenge P heißt t-**Spanner** für P und einen Dehnungsfaktor $t \geq 1$, falls für alle Paare $x, y \in P$ gilt

$$||xy|| \leq \delta_G(x, y) \leq t \cdot ||xy||,$$

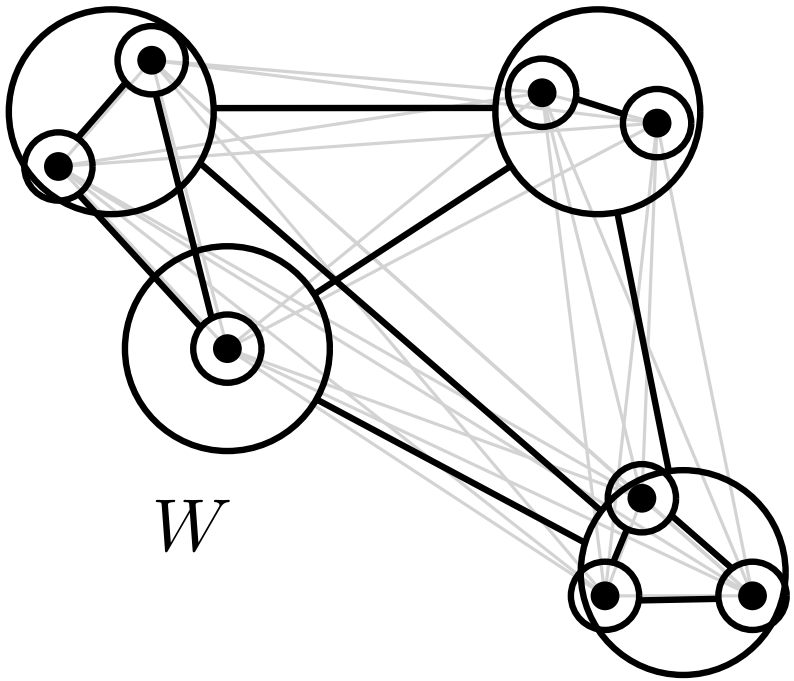
wobei $\delta_G(x, y)$ die Länge eines kürzesten x-y-Weges in G ist.

WSPD und t-Spanner

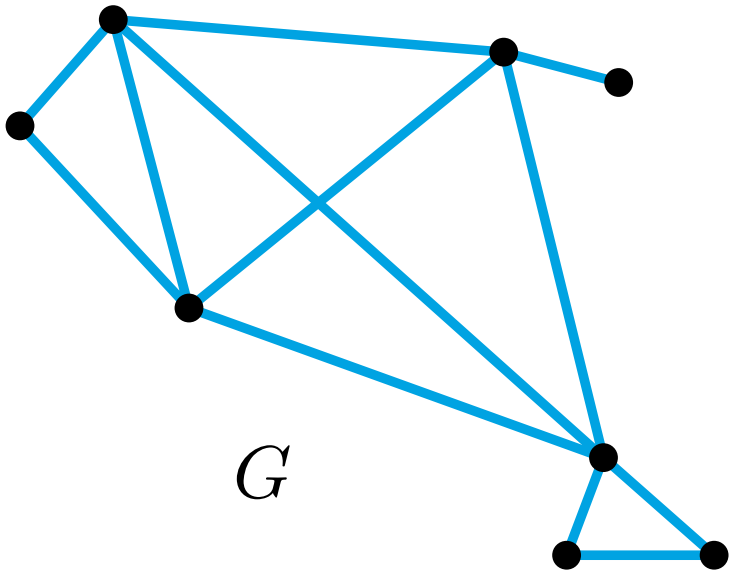
Def.: Für eine Menge P von n Punkten in $\mathbb{R}^d$ und eine WSPD W von P definiere einen Graphen $G = (P, E)$ mit $E = \{\{x, y\} \mid \{u, v\} \in W \text{ und } \mathsf{rep}(u) = x, \mathsf{rep}(v) = y\}$.

WSPD und t-Spanner

Def.: Für eine Menge P von n Punkten in $\mathbb{R}^d$ und eine WSPD W von P definiere einen Graphen $G = (P, E)$ mit $E = \{\{x, y\} \mid \{u, v\} \in W \text{ und } \mathsf{rep}(u) = x, \mathsf{rep}(v) = y\}$.



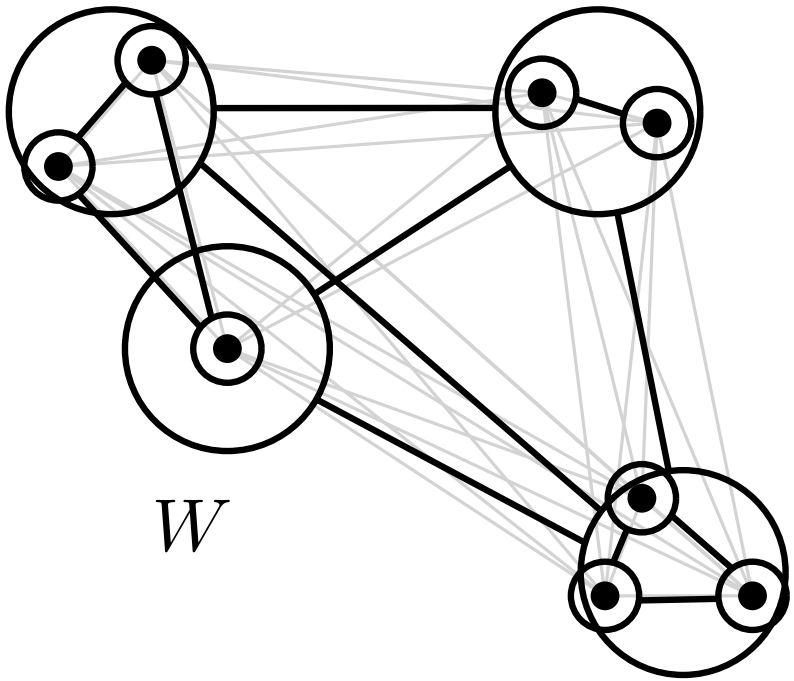
W



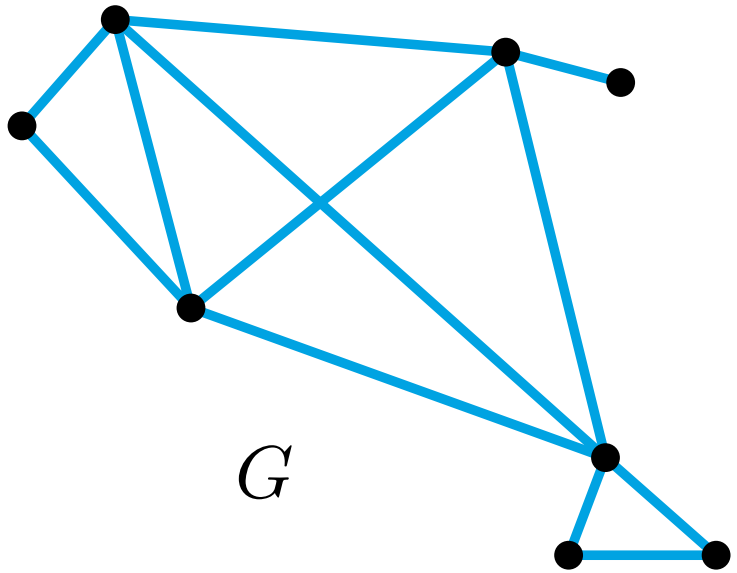
G

WSPD und t-Spanner

Def.: Für eine Menge P von n Punkten in $\mathbb{R}^d$ und eine WSPD W von P definiere einen Graphen $G = (P, E)$ mit $E = \{\{x, y\} \mid \{u, v\} \in W \text{ und } \mathsf{rep}(u) = x, \mathsf{rep}(v) = y\}$.



W



G

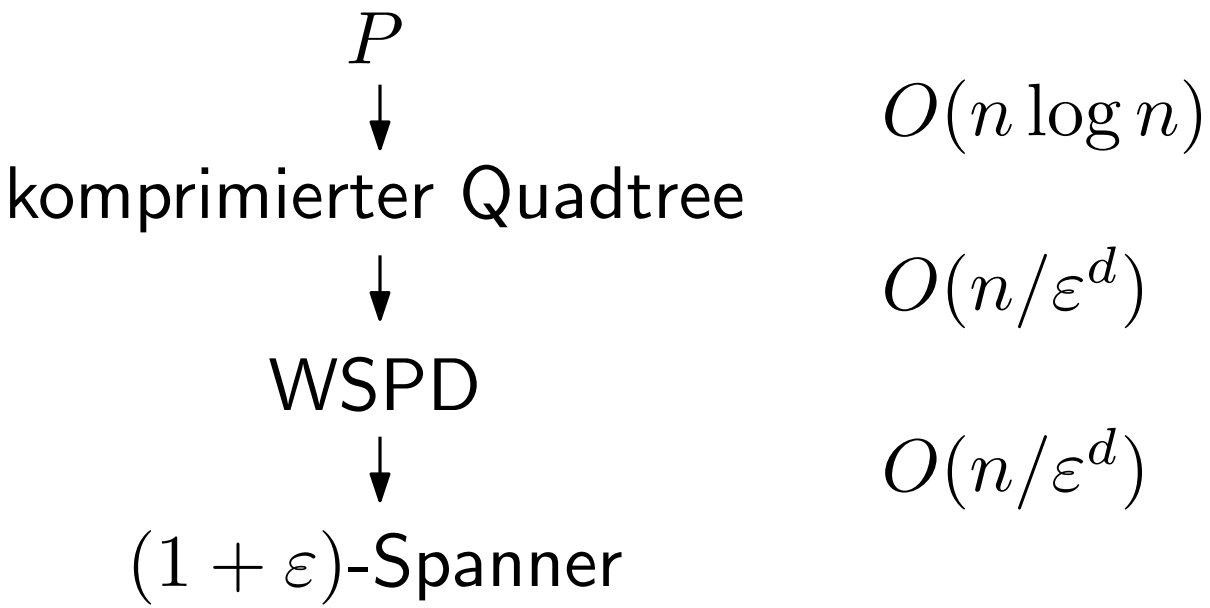
Beh.: Ist W eine s-WSPD für ein geeignetes $s = s(t)$ so ist G ein t-Spanner für P mit $O(s^d n)$ Kanten.

Zusammenfassung

Satz 3: Für eine Menge P von n Punkten in $\mathbb{R}^d$ und ein $\varepsilon \in (0, 1]$ kann ein $(1 + \varepsilon)$-Spanner für P mit $O(n/\varepsilon^d)$ Kanten in $O(n \log n + n/\varepsilon^d)$ Zeit berechnet werden.

Zusammenfassung

Satz 3: Für eine Menge P von n Punkten in $\mathbb{R}^d$ und ein $\varepsilon \in (0, 1]$ kann ein $(1+\varepsilon)$-Spanner für P mit $O(n/\varepsilon^d)$ Kanten in $O(n \log n + n/\varepsilon^d)$ Zeit berechnet werden.



Diskussion

Karlsruhe Institute of Technology

Welche weiteren Anwendungen hat die WSPD?

Diskussion

Welche weiteren Anwendungen hat die WSPD?

WSPD bietet sich immer dann an, wenn man auf die $\Theta(n^2)$ exakten Distanzen in einer Punktmenge verzichten kann und diese stattdessen approximiert:

- Berechnung der paarweisen Abstoßungskräfte von n Objekten, z.B. für kräftebasierte Algorithmen im Graphenzeichnen
- Berechnung von approx. Durchmesser und nähestem Paar (exakt!) einer Punktemenge
- schnelle Berechnung von EMST

Diskussion

Welche weiteren Anwendungen hat die WSPD?

WSPD bietet sich immer dann an, wenn man auf die $\Theta(n^2)$ exakten Distanzen in einer Punktmenge verzichten kann und diese stattdessen approximiert:

- Berechnung der paarweisen Abstoßungskräfte von n Objekten, z.B. für kräftebasierte Algorithmen im Graphenzeichnen
- Berechnung von approx. Durchmesser und nähestem Paar (exakt!) einer Punktemenge
- schnelle Berechnung von EMST

Wozu geometrisch approximieren?

Diskussion

Welche weiteren Anwendungen hat die WSPD?

WSPD bietet sich immer dann an, wenn man auf die $\Theta(n^2)$ exakten Distanzen in einer Punktmenge verzichten kann und diese stattdessen approximiert:

- Berechnung der paarweisen Abstoßungskräfte von n Objekten, z.B. für kräftebasierte Algorithmen im Graphenzeichnen
- Berechnung von approx. Durchmesser und nähestem Paar (exakt!) einer Punktemenge
- schnelle Berechnung von EMST

Wozu geometrisch approximieren?

Einerseits ersetzt man dadurch langsame Berechnungen durch schnellere (aber ungenauere), andererseits sind oft auch die Eingabedaten schon mit einer gewissen Ungenauigkeit behaftet, so dass approximative Lösungen je nach Anwendung ausreichend sind.