

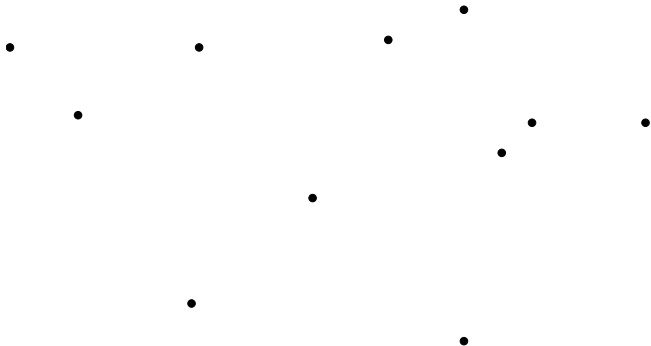
Worum geht es?

Gegeben: Eine Menge von Punkten,

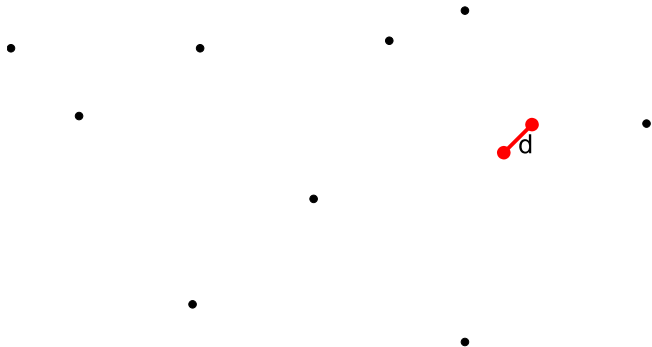
Worum geht es?

Gegeben: Eine Menge von Punkten, gesucht die beiden nächsten Punkte oder ihr Abstand.

Problem



Lösung



Idee

Klassischer Ansatz der dynamischen Programmierung

Idee

Klassischer Ansatz der dynamischen Programmierung

Ziel: Laufzeit $O(n \log(n))$

Vorgehen

1. Vorsortieren

Vorgehen

1. Vorsortieren
2. Divide

Vorgehen

1. Vorsortieren
2. Divide
3. Conquer

Vorgehen

1. Vorsortieren
2. Divide
3. Conquer
4. Combine

Vorsortieren

Zunächst müssen die Punkte nach jeweils x und y Koordinaten sortiert werden.

Vorsortieren

Zunächst müssen die Punkte nach jeweils x und y Koordinaten sortiert werden.

Sei im weiteren X die nach x sortierte Punktfolge

Vorsortieren

Zunächst müssen die Punkte nach jeweils x und y Koordinaten sortiert werden.

Sei im weiteren X die nach x sortierte Punktfolge

Sei im weiteren Y die nach y sortierte Punktfolge

Divide

Eingabe: X und Y

Divide

Eingabe: X und Y

Aufteilen einer großen Menge X in zwei kleine X_L und X_R :

Divide

Eingabe: X und Y

Aufteilen einer großen Menge X in zwei kleine X_L und X_R :

Median bilden, NICHT Mittelwert (könnte Ausarten)

Divide

Eingabe: X und Y

Aufteilen einer großen Menge X in zwei kleine X_L und X_R :

Median bilden, NICHT Mittelwert (könnte Ausarten)

Einfach, da Arrays zugrunde liegen

Divide

Eingabe: X und Y

Aufteilen einer großen Menge X in zwei kleine X_L und X_R :

Median bilden, NICHT Mittelwert (könnte Ausarten)

Einfach, da Arrays zugrunde liegen

In X_L kommen alle Punkte links vom Median,

in X_R kommen alle Punkte rechts vom Median,

Divide

Eingabe: X und Y

Aufteilen einer großen Menge X in zwei kleine X_L und X_R :

Median bilden, NICHT Mittelwert (könnte Ausarten)

Einfach, da Arrays zugrunde liegen

In X_L kommen alle Punkte links vom Median,

in X_R kommen alle Punkte rechts vom Median,

Sortierung in X bleibt erhalten.

Divide

Eingabe: X und Y

Aufteilen einer großen Menge X in zwei kleine XL und XR:

Median bilden, NICHT Mittelwert (könnte Ausarten)

Einfach, da Arrays zugrunde liegen

In XL kommen alle Punkte links vom Median,

in XR kommen alle Punkte rechts vom Median,

Sortierung in X bleibt erhalten.

Selbiges für YL und YR.

Divide

Eingabe: X und Y

Aufteilen einer großen Menge X in zwei kleine XL und XR:

Median bilden, NICHT Mittelwert (könnte Ausarten)

Einfach, da Arrays zugrunde liegen

In XL kommen alle Punkte links vom Median,

in XR kommen alle Punkte rechts vom Median,

Sortierung in X bleibt erhalten.

Selbiges für YL und YR.

Sonderfall: Array X enthält nur 3 Punkte

Divide

Eingabe: X und Y

Aufteilen einer großen Menge X in zwei kleine XL und XR:

Median bilden, NICHT Mittelwert (könnte Ausarten)

Einfach, da Arrays zugrunde liegen

In XL kommen alle Punkte links vom Median,

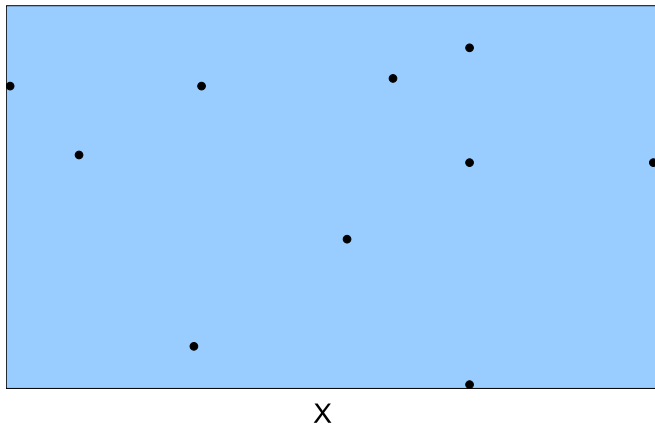
in XR kommen alle Punkte rechts vom Median,

Sortierung in X bleibt erhalten.

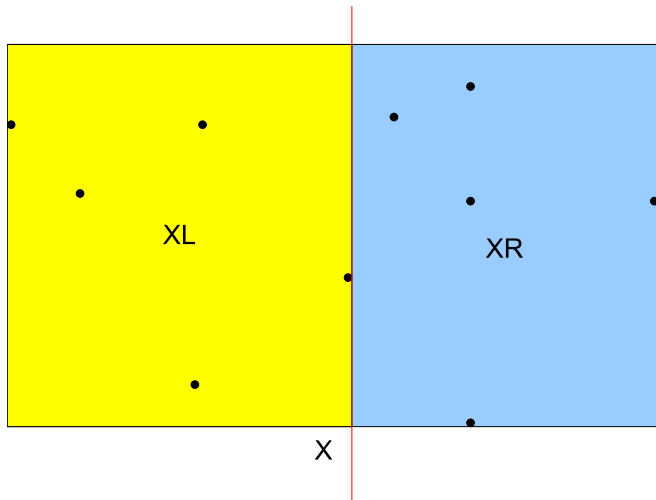
Selbiges für YL und YR.

Sonderfall: Array X enthält nur 3 Punkte → bruteforce Algorithmus.

Divide



Divide



Conquer

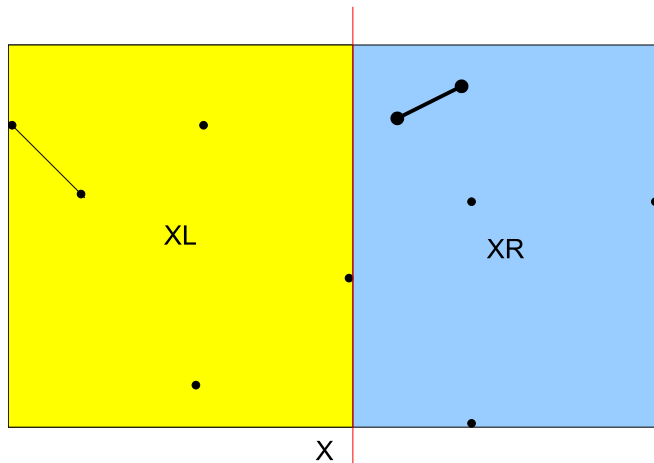
Rekursive divide Aufrufe liefern δ_L und δ_R als minimale Abstände.

Conquer

Rekursive divide Aufrufe liefern δ_L und δ_R als minimale Abstände.
Setze δ als kleineren der Beiden.

Conquer

Rekursive divide Aufrufe liefern δ_L und δ_R als minimale Abstände.
Setze δ als kleineren der Beiden.

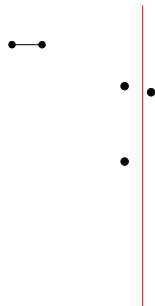


Combine

Der kleinste Abstand δ ist entweder schon gefunden, oder aber es gibt noch Punkte in dem 2δ breiten Streifen um die Linie, die einen kleineren Abstand haben.

Combine

Der kleinste Abstand δ ist entweder schon gefunden, oder aber es gibt noch Punkte in dem 2δ breiten Streifen um die Linie, die einen kleineren Abstand haben.



Combine

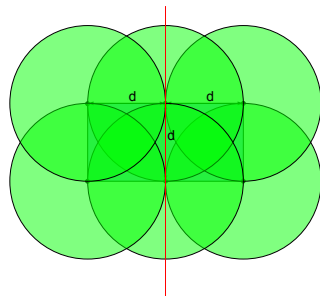
Alle Punkte aus dem Streifen kommen in Y' ,

Combine

Alle Punkte aus dem Streifen kommen in Y' , und jeder Punkt mit den 7(!) Nachfolgenden auf Abstand geprüft werden.

Combine

Alle Punkte aus dem Streifen kommen in Y' , und jeder Punkt mit den 7(!) Nachfolgenden auf Abstand geprüft werden.



Laufzeitanalyse

Vorsortieren: $O(n \log(n))$

Laufzeitanalyse

Vorsortieren: $O(n \log(n))$

$\log_2(n)$ Durchläufe

Laufzeitanalyse

Vorsortieren: $O(n \log(n))$

$\log_2(n)$ Durchläufe

Divide: $O(|X| + |Y|)$ pro Schritt, insgesamt also $O(n \log(n))$

Laufzeitanalyse

Vorsortieren: $O(n \log(n))$

$\log_2(n)$ Durchläufe

Divide: $O(|X| + |Y|)$ pro Schritt, insgesamt also $O(n \log(n))$

Conquer: $O(1)$

Laufzeitanalyse

Vorsortieren: $O(n \log(n))$

$\log_2(n)$ Durchläufe

Divide: $O(|X| + |Y|)$ pro Schritt, insgesamt also $O(n \log(n))$

Conquer: $O(1)$

Combine: $O(1)$ wegen den 7 Punkten

Laufzeitanalyse

Vorsortieren: $O(n \log(n))$

$\log_2(n)$ Durchläufe

Divide: $O(|X| + |Y|)$ pro Schritt, insgesamt also $O(n \log(n))$

Conquer: $O(1)$

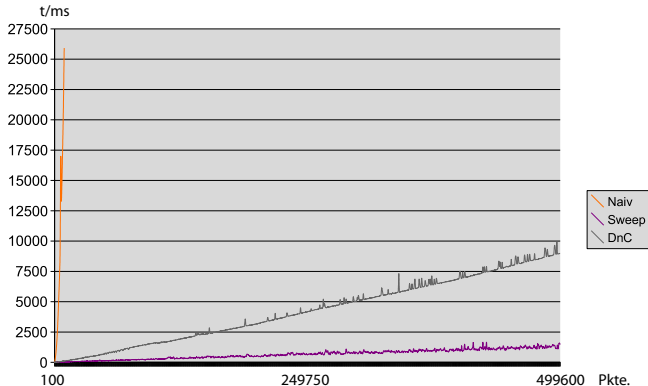
Combine: $O(1)$ wegen den 7 Punkten Gesamt: $O(n \log(n))$

Vergleich der Laufzeiten

Die Laufzeiten wurden mit dem am Ende angegebenen Quellcode ermittelt.

Vergleich der Laufzeiten

Die Laufzeiten wurden mit dem am Ende angegebenen Quellcode ermittelt.



Naiver Algorithmus nur bei sehr kleinen Mengen zu gebrauchen

Naiver Algorithmus nur bei sehr kleinen Mengen zu gebrauchen
Divide and Conquer in $O(n \log(n))$, aber immernoch langsamer als
Plane Sweep.

Naiver Algorithmus nur bei sehr kleinen Mengen zu gebrauchen
Divide and Conquer in $O(n \log(n))$, aber immernoch langsamer als
Plane Sweep. Plane Sweep sortiert im Baum, während Divide and
Conquer komplett 2 mal sortiert.



Introduction to Algorithms. MIT Press 2001.



Computational Geometry: An Introduction. Springer-Verlag, 1985.



Den Sourcecode der Algorithmen gibt es online unter
<http://zodiac.dnsalias.org/misc/ProSemDastru.tar.gz>