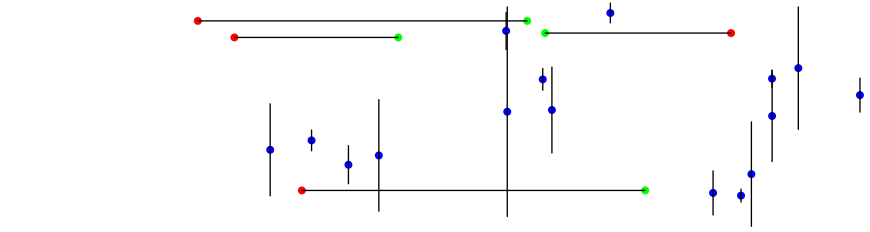
Übersicht

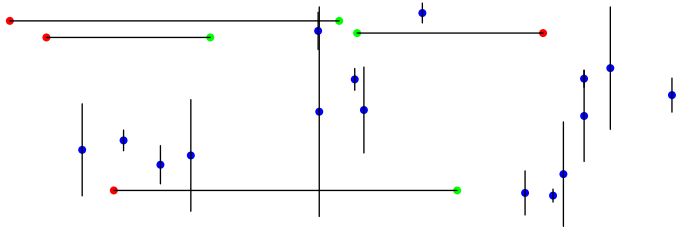
Sweepline-Algorithmen



- Interessante Punkte: Nach x-Koordinate sortieren
- Es gibt ein aktive Menge von Segmenten
- Eine imaginäre vertikale Linie bewegt sich von links nach rechts
- Roter Punkt: Segment in aktive Menge aufnehmen
- Grüner Punkt: Segment aus aktiver Menge entfernen
- Blauer Punkt: Segment mit aktiver Menge vergleichen

Suche nur Schnitte zwischen horizontalem und vertikalem Segment.

Sweepline-Algorithmen

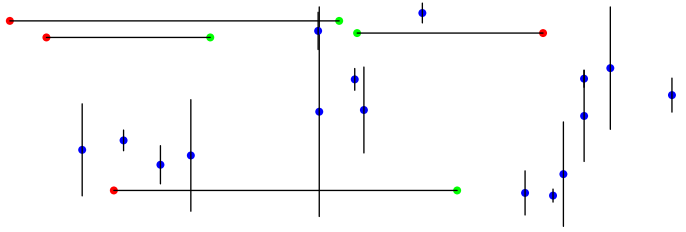


- Wie finden wir bei einem vertikalen Segment die geschnittenen horizontalen Segmente?
- Welche Datenstruktur für die aktive Menge?

Lösung: Speichere y-Koordinaten Y der aktiven Menge in balanziertem Suchbaum.
Gibt es einen Schnitt? $\rightarrow O(\log |Y|)$ Schritte

Alle Schnitte S ausgeben: $O(\log |Y| + |S|)$ Schritte
In aktive Menge einfügen oder löschen: $O(\log |Y|)$ Schritte

Sweepline-Algorithmen

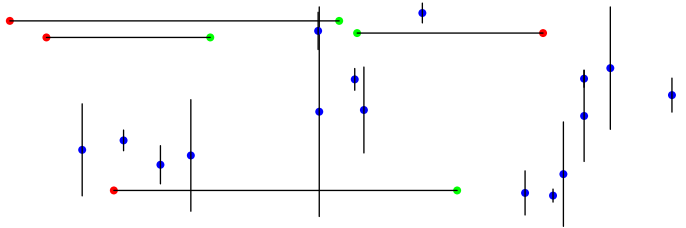


- Wie finden wir bei einem vertikalen Segment die geschnittenen horizontalen Segmente?
- Welche Datenstruktur für die aktive Menge?

Lösung: Speichere y-Koordinaten Y der aktiven Menge in balanziertem Suchbaum.
Gibt es einen Schnitt? $\rightarrow O(\log |Y|)$ Schritte

Alle Schnitte S ausgeben: $O(\log |Y| + |S|)$ Schritte
In aktive Menge einfügen oder löschen: $O(\log |Y|)$ Schritte

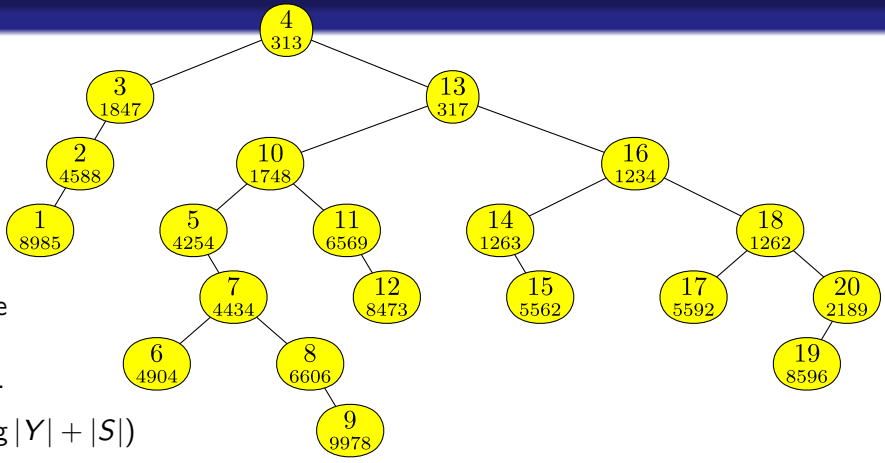
Sweepline-Algorithmen



- Wie finden wir bei einem vertikalen Segment die geschnittenen horizontalen Segmente?
- Welche Datenstruktur für die aktive Menge?

Lösung: Speichere y-Koordinaten Y der aktiven Menge in balanziertem Suchbaum.
Gibt es einen Schnitt? $\rightarrow O(\log |Y|)$ Schritte

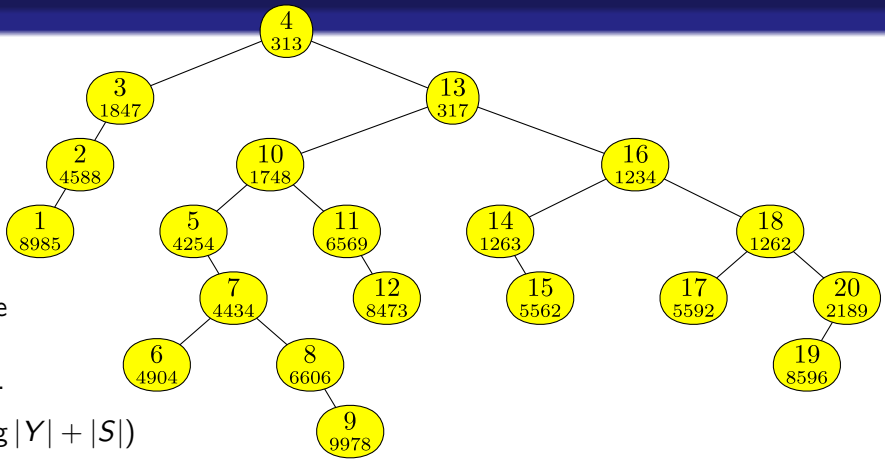
Alle Schnitte S ausgeben: $O(\log |Y| + |S|)$ Schritte
In aktive Menge einfügen oder löschen: $O(\log |Y|)$ Schritte



Aufgabe: Finde y-Koordinaten mit $a \leq y \leq b$.

Laufzeit: $O(\log |Y| + |S|)$

1. Finde größtes Element, daß kleiner oder gleich a ist.
2. Gehe von dort aus Element aufsteigend durch
3. Beende, wenn b überschritten.



Aufgabe: Finde y-Koordinaten mit $a \leq y \leq b$.

Laufzeit: $O(\log |Y| + |S|)$

1. Finde größtes Element, daß kleiner oder gleich a ist.
2. Gehe von dort aus Element aufsteigend durch
3. Beende, wenn b überschritten.

Range-Search

Operationen:

1. Einfügen einer Zahl x
2. Löschen einer Zahl x
3. Ausgabe aller gespeicherter Zahlen in $[a, b]$

Welche Datenstrukturen sind geeignet?

- Arrays?
- Listen?
- AVL-Bäume?
- Splay-Bäume?
- $(2, 3)$-Bäume?
- Treaps?
- Skiplists?
- Hashtabellen?

Range-Search

Operationen:

1. Einfügen einer Zahl x
2. Löschen einer Zahl x
3. Ausgabe aller gespeicherter Zahlen in $[a, b]$

Welche Datenstrukturen sind geeignet?

- Arrays?
- Listen?
- AVL-Bäume?
- Splay-Bäume?
- $(2, 3)$-Bäume?
- Treaps?
- Skiplists?
- Hashtabellen?

Range-Search

Operationen:

1. Einfügen einer Zahl x
2. Löschen einer Zahl x
3. Ausgabe aller gespeicherter Zahlen in $[a, b]$

Welche Datenstrukturen sind geeignet?

- Arrays?
- Listen?
- AVL-Bäume?
- Splay-Bäume?
- $(2, 3)$-Bäume?
- Treaps?
- Skiplists?
- Hashtabellen?

Range-Search

Operationen:

1. Einfügen einer Zahl x
2. Löschen einer Zahl x
3. Ausgabe aller gespeicherter Zahlen in $[a, b]$

Welche Datenstrukturen sind geeignet?

- Arrays?
- Listen?
- AVL-Bäume?
- Splay-Bäume?
- $(2, 3)$-Bäume?
- Treaps?
- Skiplists?
- Hashtabellen?

Range-Search

Operationen:

1. Einfügen einer Zahl x
2. Löschen einer Zahl x
3. Ausgabe aller gespeicherter Zahlen in $[a, b]$

Welche Datenstrukturen sind geeignet?

- Arrays?
- Listen?
- AVL-Bäume?
- Splay-Bäume?
- $(2, 3)$-Bäume?
- Treaps?
- Skiplists?
- Hashtabellen?

Range-Search

Operationen:

1. Einfügen einer Zahl x
2. Löschen einer Zahl x
3. Ausgabe aller gespeicherter Zahlen in $[a, b]$

Welche Datenstrukturen sind geeignet?

- Arrays?
- Listen?
- AVL-Bäume?
- Splay-Bäume?
- $(2, 3)$-Bäume?
- Treaps?
- Skiplists?
- Hashtabellen?

Range-Search

Operationen:

1. Einfügen einer Zahl x
2. Löschen einer Zahl x
3. Ausgabe aller gespeicherter Zahlen in $[a, b]$

Welche Datenstrukturen sind geeignet?

- Arrays?
- Listen?
- AVL-Bäume?
- Splay-Bäume?
- $(2, 3)$-Bäume?
- Treaps?
- Skiplists?
- Hashtabellen?

Range-Search

Operationen:

1. Einfügen einer Zahl x
2. Löschen einer Zahl x
3. Ausgabe aller gespeicherter Zahlen in $[a, b]$

Welche Datenstrukturen sind geeignet?

- Arrays?
- Listen?
- AVL-Bäume?
- Splay-Bäume?
- $(2, 3)$-Bäume?
- Treaps?
- Skiplists?
- Hashtabellen?

Range-Search

Operationen:

1. Einfügen einer Zahl x
2. Löschen einer Zahl x
3. Ausgabe aller gespeicherter Zahlen in $[a, b]$

Welche Datenstrukturen sind geeignet?

- Arrays?
- Listen?
- AVL-Bäume?
- Splay-Bäume?
- $(2, 3)$-Bäume?
- Treaps?
- Skiplists?
- Hashtabellen?

Range-Search

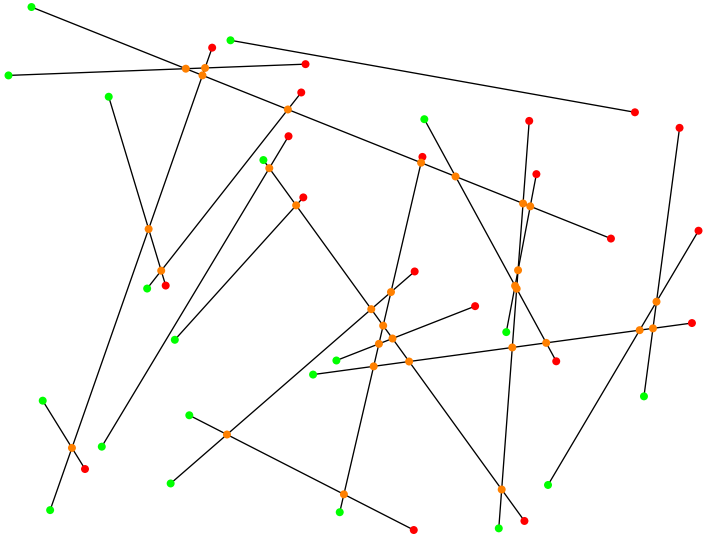
Operationen:

1. Einfügen einer Zahl x
2. Löschen einer Zahl x
3. Ausgabe aller gespeicherter Zahlen in $[a, b]$

Welche Datenstrukturen sind geeignet?

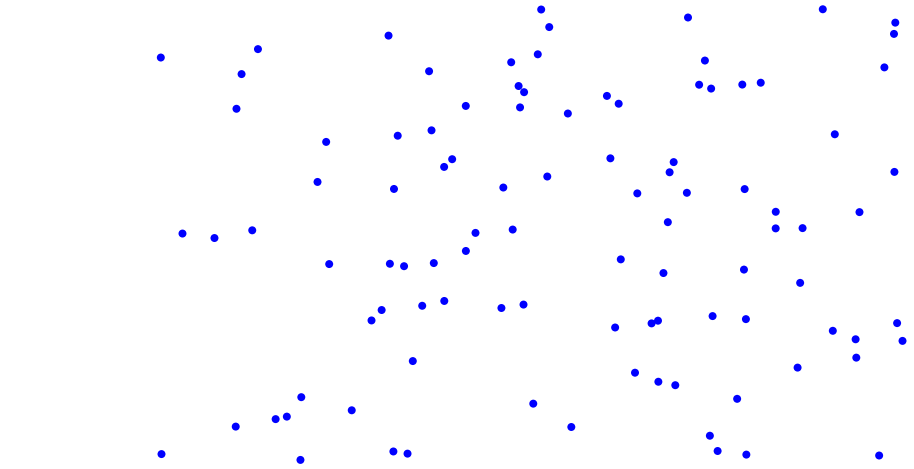
- Arrays?
- Listen?
- AVL-Bäume?
- Splay-Bäume?
- $(2, 3)$-Bäume?
- Treaps?
- Skiplists?
- Hashtabellen?

Das allgemeine Segmentschnittproblem

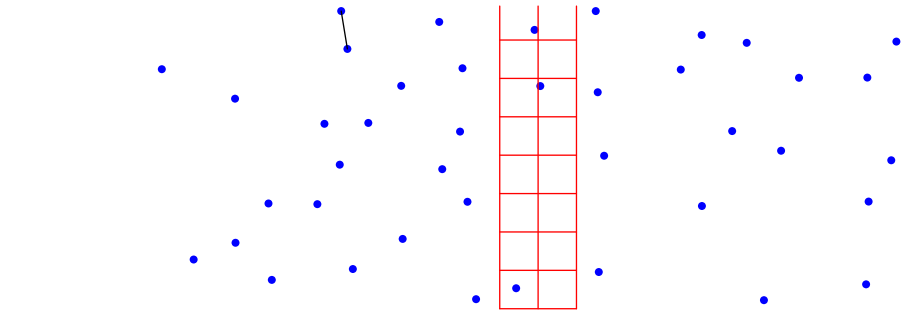


Übersicht

Nächste Nachbarn



Nächste Nachbarn – Divide-and-Conquer



1. Sortiere nach x-Koordinate
2. Teile in linke und rechte Hälfte
3. Finde rekursiv nächste Nachbarn links und rechts (Abstand d)
4. Finde nächste Nachbarn in einem Streifen der Breite $2d$ um die Trennlinie

Nächste Nachbarn – Analyse

Es sei $T(n)$ die Zeit, um n Punkte zu bearbeiten, nachdem sie nach x-Koordinate sortiert wurden.

Nehmen wir an, n sei eine Zweierpotenz.

Wir müssen zwei Probleme der Größe $n/2$ lösen und anschließend die Punkte im Mittelstreifen behandeln.

Es ergibt sich diese Rekursionsgleichung:

$$T(n) = 2\,T(n/2) + O(n)$$

Dies ist die selbe Rekursionsgleichung wie für Mergesort.

Laufzeit: $O(n \log n)$

Nächste Nachbarn – Analyse

Es sei $T(n)$ die Zeit, um n Punkte zu bearbeiten, nachdem sie nach x-Koordinate sortiert wurden.

Nehmen wir an, n sei eine Zweierpotenz.

Wir müssen zwei Probleme der Größe $n/2$ lösen und anschließend die Punkte im Mittelstreifen behandeln.

Es ergibt sich diese Rekursionsgleichung:

$$T(n) = 2T(n/2) + O(n)$$

Dies ist die selbe Rekursionsgleichung wie für Mergesort.

Laufzeit: $O(n \log n)$

Nächste Nachbarn – Analyse

Es sei $T(n)$ die Zeit, um n Punkte zu bearbeiten, nachdem sie nach x-Koordinate sortiert wurden.

Nehmen wir an, n sei eine Zweierpotenz.

Wir müssen zwei Probleme der Größe $n/2$ lösen und anschließend die Punkte im Mittelstreifen behandeln.

Es ergibt sich diese Rekursionsgleichung:

$$T(n) = 2T(n/2) + O(n)$$

Dies ist die selbe Rekursionsgleichung wie für Mergesort.

Laufzeit: $O(n \log n)$