Datenbankanwendung

Wintersemester 2014/15

Prof. Dr.-Ing. Sebastian Michel
TU Kaiserslautern

smichel@cs.uni-kl.de

Wiederholung: Min-Hashing

- Gegeben zwei Mengen A und B von Objekten.
- Ein oft benutztes Ähnlichkeitsmaß ist der Jaccard Koeffizient:

$$J(A,B) := \frac{|A \cap B|}{|A \cup B|}$$

Wiederholung: Idee hinter Min-Hashing

- Berechne für alle Elemente einer Menge A eine Hashfunktion, die $a \in A$ einem Integer-Wert zuweist.
- Hashfunktion hat hier nichts mit LSH zu tun, es ist eine "normale".
- Betrachte den kleinsten dieser Werte $h_{min}(A)$
- Wie groß ist die Wahrscheinlichkeit, dass zwei Mengen A und B dieser min-Wert identisch ist?

Paper: Andrei Broder. On the resemblance and containment of documents. 1997.

Wiederholung: Min-Hashing (2)

- Die Wahrscheinlichkeit $Pr[h_{min}(A) = h_{min}(B)]$ steht in direkter Verbindung zum Jaccard-Koeffizienten:

$$Pr[h_{min}(A) = h_{min}(B)] = \frac{|A \cap B|}{|A \cup B|}$$

Anwendung

- Suche nach ähnlichen Mengen: Betrachte nur Paare von Mengen, deren min-Wert identisch ist
- Bzw., nehme $Pr[h_{min}(A) = h_{min}(B)]$ als Näherung für den Jaccard-Koeffizienten
- Wie gut funktioniert das?

Wiederholung: Min-Hashing - Mehrere min-Werte bzw. Hashfunktionen

Mehrere Hash-Funktionen mit je einem min-Wert

- Betrachte k (unabhängige) Hashfunktionen, die jeweils einen min-Wert liefern.
- Approximiere $J(A, B)$ mit Anteil der übereinstimmenden min-Werte.

Mehrere min-Werte & eine Hashfunktionen

- Betrachte nur eine Hashfunktion, aber nehme von dieser die k kleinsten Werte.

Der Fehler ist bei beiden Schemata $O(1/\sqrt{k})$.

Indexstrukturen für mehrdimensionale Daten

Literatur und Verweis auf Demo-Seite

Umfassendes Buch zum Thema mehrdimensionale Indexstrukturen: Hanan Samet. Foundations of multidimensional and metric data structures, Elsevier/Morgan Kaufmann 1996.

Tolle Sammlung von Demos zur Veranschaulichung verschiedener Indexstrukturen, auf `http://donar.umiacs.umd.edu/quadtree/`.

Mehrdimensionale Daten

Problemstellung

- Gegeben eine Menge von Datensätzen
- Jeder Datensatz ist durch einen d-dimensionalen Schlüssel $A_1, A_2, \ldots A_d$ identifiziert.

Anforderungen an Indexstrukturen

- Effiziente Anfragevearbeitung für verschiedene Typen von Anfragen, wie
  - Punktanfragen
  - Bereichsanfragen
  - Partielle Anfragen
  - Nächste-Nachbar-Anfragen
- Operationen zum Einfügen, Löschen, Ändern (aka. Aktualisieren)

Beispieldaten

Beispiel Tabelle mit Städten und ihren X und Y Koordinaten

Name	**X**	**Y**
Chicago	35	42
Mobile	52	10
Toronto	62	77
Buffalo	82	65
Denver	5	45
Omaha	27	35
Atlanta	85	15
Miami	90	5

Dies ist ein Beispiel für 2-dimensionale Daten: X und Y beschreiben hier offensichtlich diese Dimensionen.

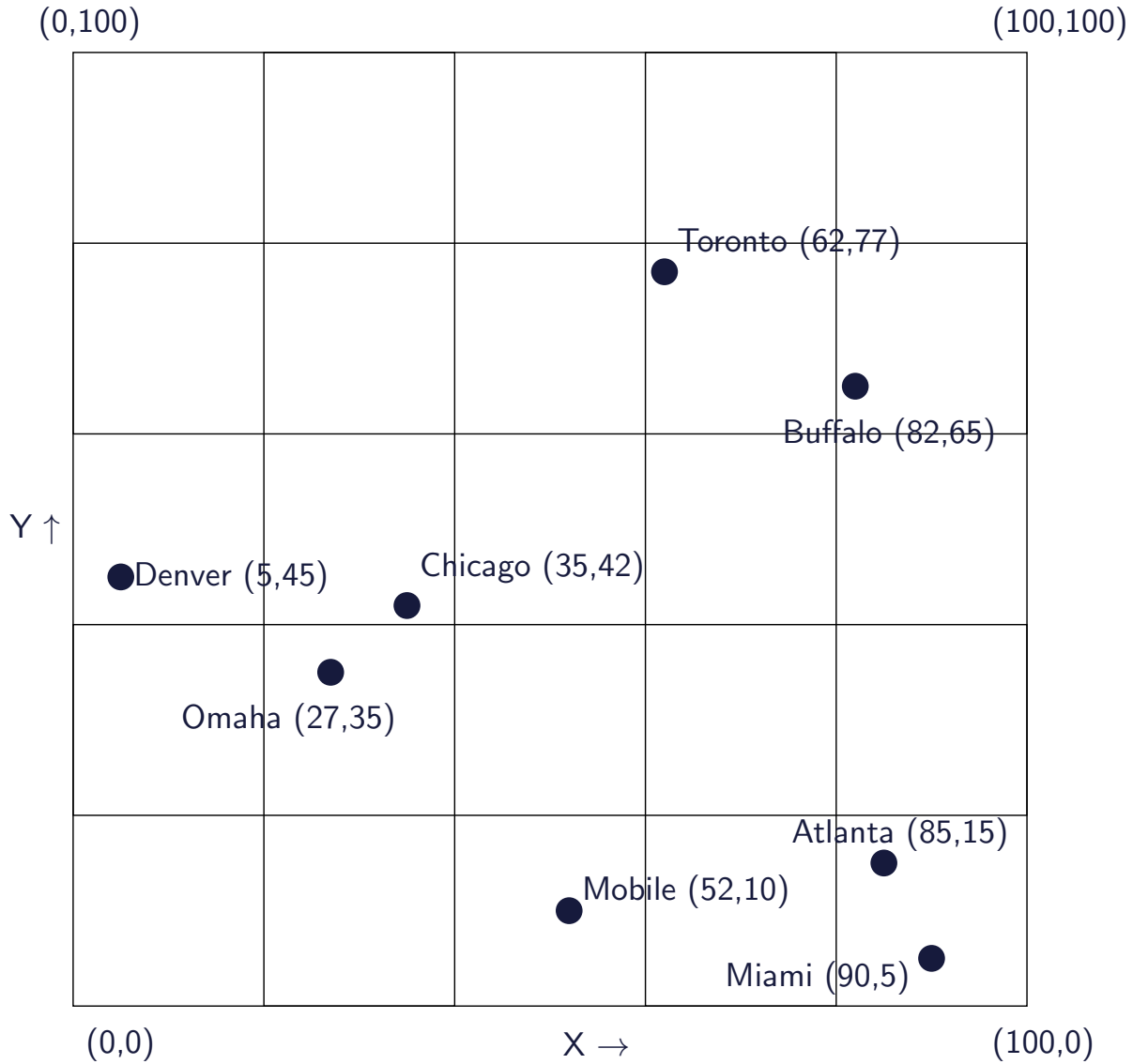
Mögliche Anfragen:

- Welche Städte liegen im Rechteck beschrieben durch Eckpunkte $(25, 25)$ und $(50, 50)$?
- Was sind die beiden nächsten Städte zu Chicago?

Partitionierung des Raumes

Wie können diese (mehrdimensionalen) Punkte indexiert werden?

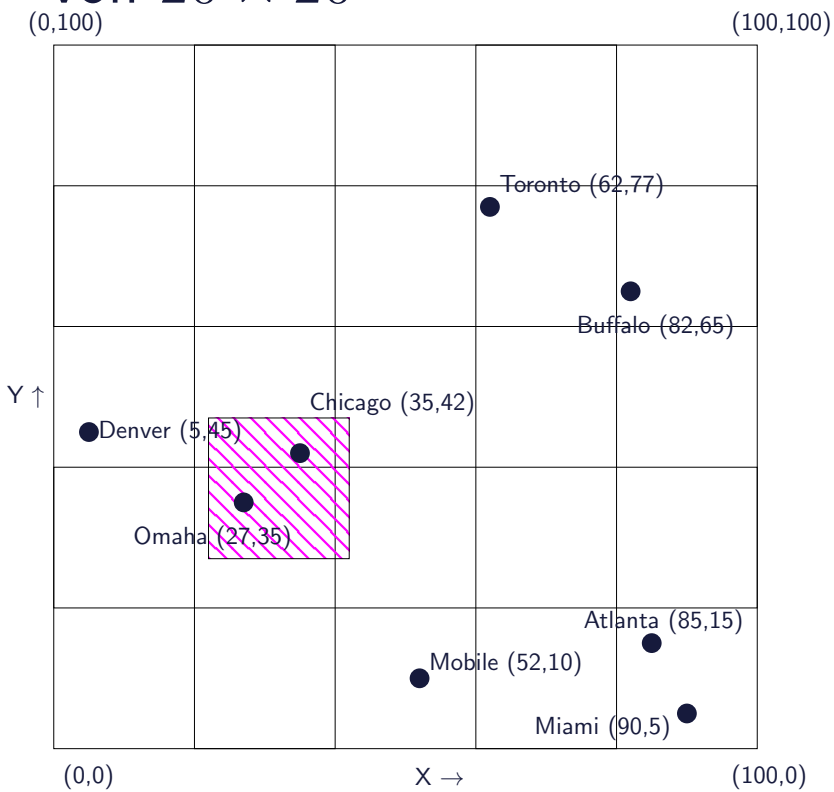
Grid mit Zellen gleicher u. fester Größe



Zugehörigkeit von Punkten: untere und links Grenze geschlossen, obere und rechte Grenze offen.

Grid mit Zellen gleicher u. fester Größe (2)

- Einteilung des Datenraums in Zellen gleicher Größe.
- hier 25 Zellen mit je einer Größe von 20×20



- Für Suche nach Punkten, die im dem 20×20 Rechteck mit Zentrum in $(32, 37)$ liegen müssten hier 4 Zellen angeschaut werden.
- Ergebnisse wären: Chicago und Omaha

In d Dimensionen: d-dimensionales Array

Überlegungen

zum Grid mit Zellen gleicher und fester Größe

Vorteile

- Sehr einfache Struktur
- Konstante Kosten für Zugriff auf Zelle für gegebenen Punkt

Nachteile

- Nur brauchbar bei statischen Daten
- Die darüberhinaus idealerweise auch noch gleichmäßig verteilt sind

Überlegungen (2)

Zellen unterschiedlicher Größe

- Auffinden der zu einem Punkt passenden Zelle nicht mehr so trivial wie zuvor
- Hat aber Vorteile, weil Zellen dem der Datenverteilung angepasst werden können, aber ...

Trotzdem: Probleme bei dynamischen Daten

- Manche Zellen werden zu voll, manche (viele) Zellen sind leer.
- Weiteres Teilen der vollen Zellen würde nur noch mehr leere Zellen erzeugen.

Adaptives "Grid"

Idee

- Vereinige (merge) benachtbarte leere Zellen in größere Zellen.
- Zu volle Zellen werden aufgespalten.

k-ary Trees

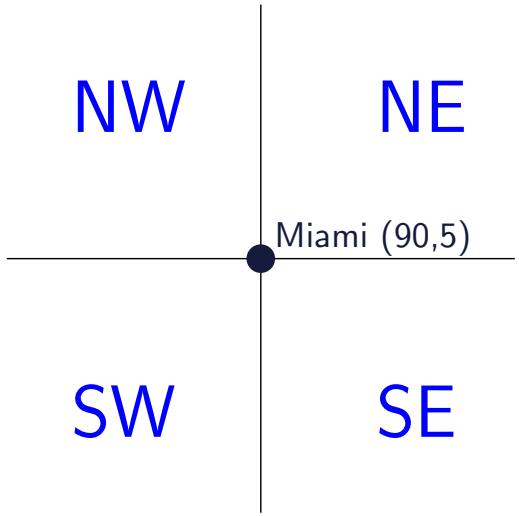
Idee

- Benutze Baum, um Zellen (Daten) zu indexieren
- k-ary, wobei k normalerweise 2^d

Generalisierung von binären Bäumen

Quadtree (Quadrantenbaum)

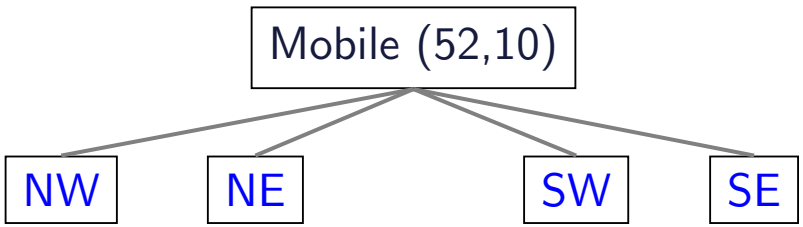
- Für 2-dimensionale Daten:
- Ein Knoten unterteilt Raum in vier "Quadranten"
- z.B. bezeichnet mit NW (North-West), NE (North-East), SW (South-West) und SE (South-East)
- 1 Datenpunkt pro Knoten (Bereich)



Konventionen

- Reihenfolge der Quadranten im Baum: NW, NE, SW, SE
- Zugehörigkeit von Punkten, die auf Linien liegen: Wie beim Grid: untere und links Grenze geschlossen, obere und rechte Grenze offen.

Quadtree - Knoten eines Baumes und Suche



- Knoten verweist auf die von ihm aus liegenden Quadranten
- Initial ist ein Knoten NIL

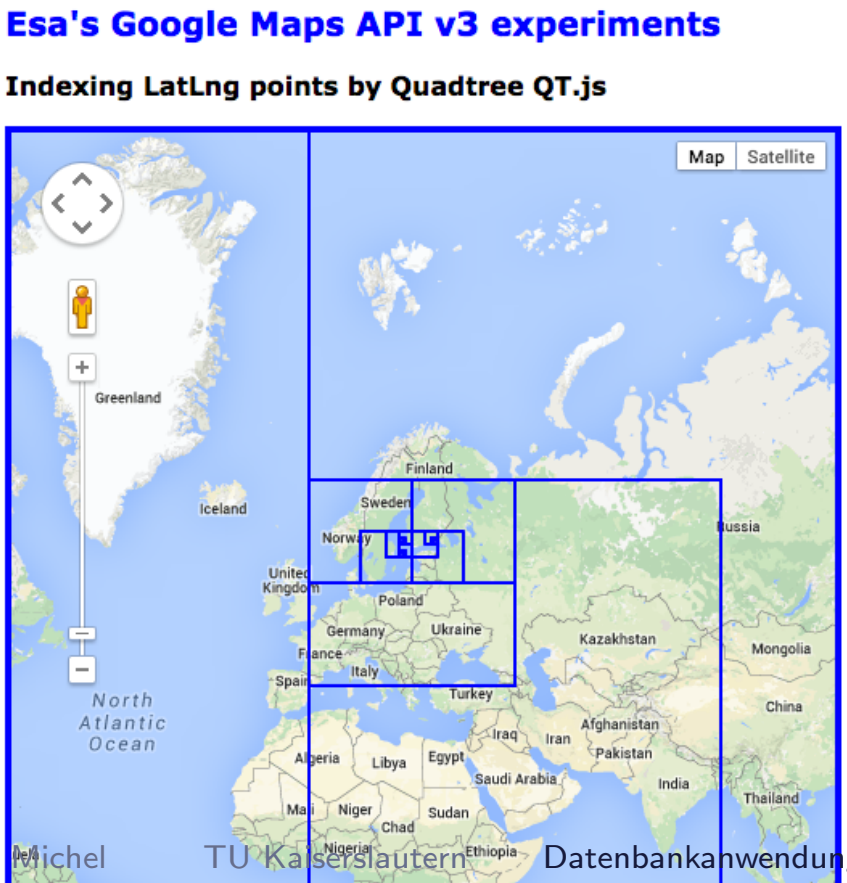
Suche nach Punkten

- Suche von Wurzel aus, geleitet durch Quadranten.
- Z.B. liegt Atlanta (85,15) in Quadrant NE (Nord-Ost) des Knotens Mobile

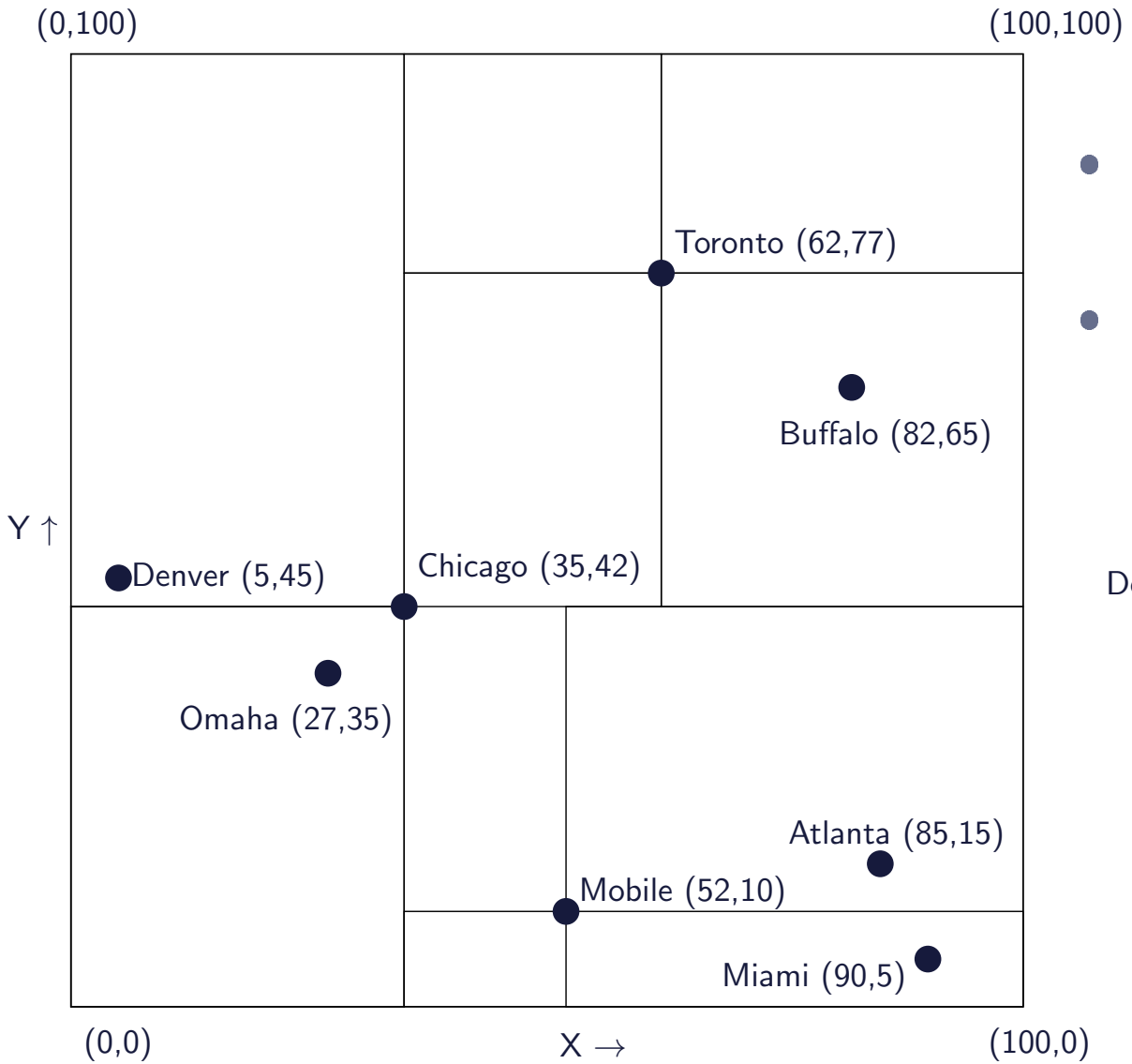
Quadtree - Google Maps Demo

Hübsche Demo zur Anwendung von Quadtrees. Zu Beachten ist hier die etwas andere Nummerierung der Quadranten (0,1,2,3, beginnend oben links in Form des Buchstabens Z). Gegend um Kaiserslautern: 12020323... (TU KL: 1202032312220)

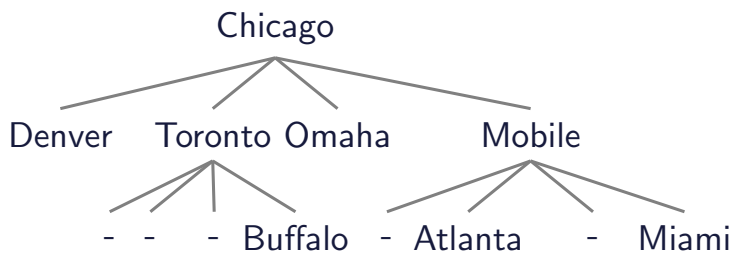
`http://koti.mbnet.fi/ojalesa/quadtree/quadtree_intro.htm`



Quadtree - Beispiel

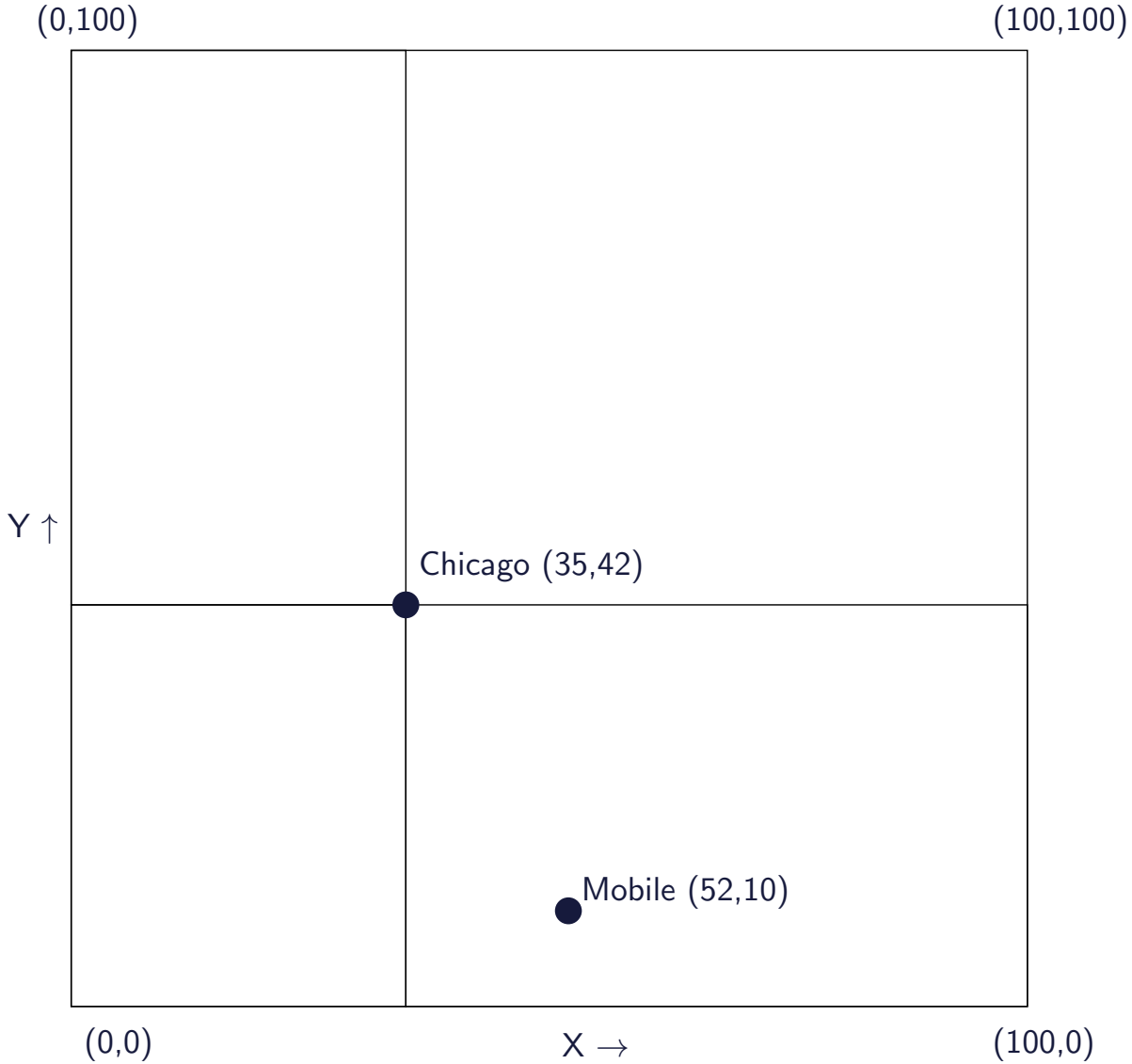


- Quadtree für Beispieldaten
- Reihenfolge des Einfügens wie in Daten-Tabelle



Quadtree - Beispielaufbau

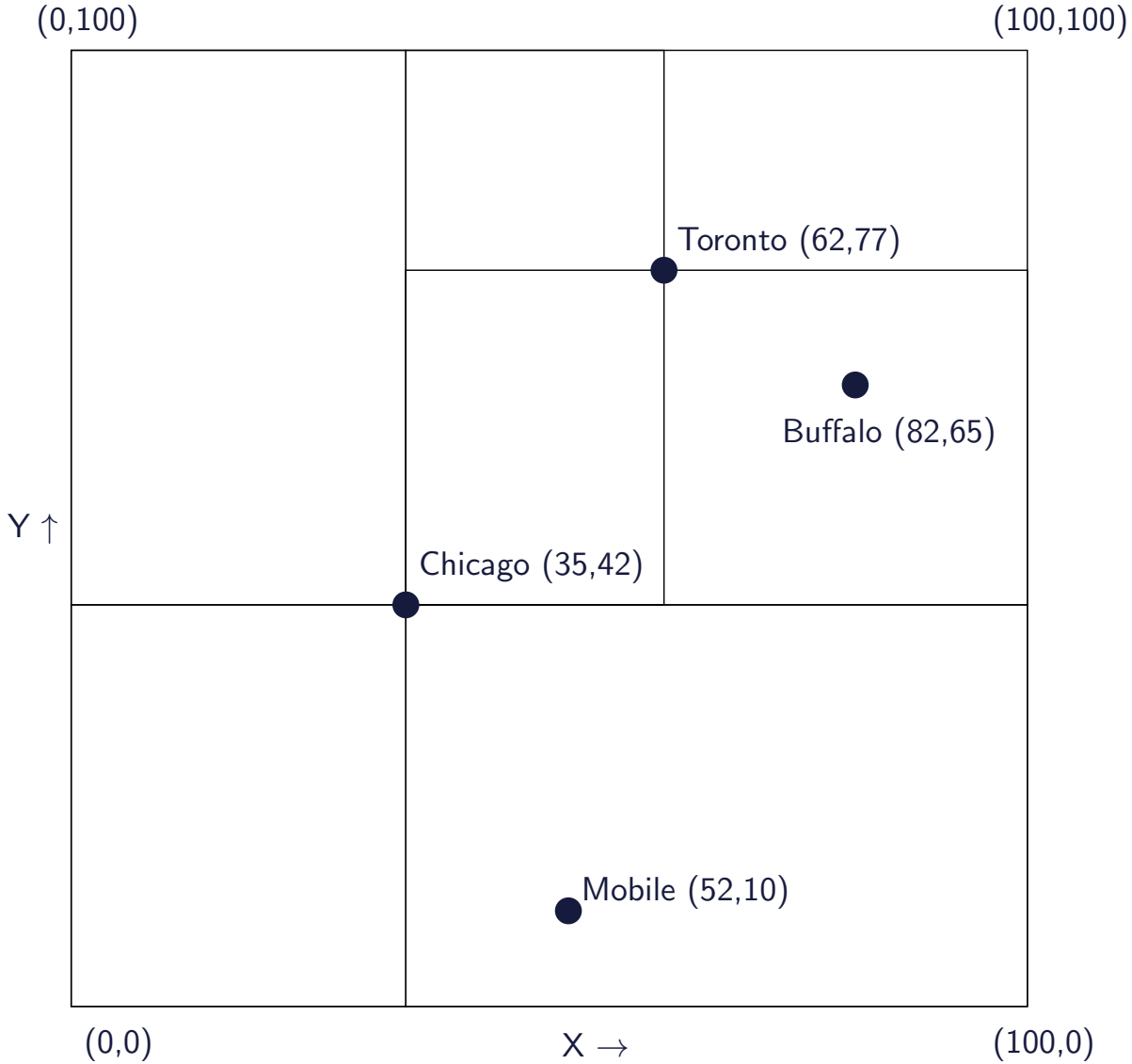
Einfügen von Chicago und Mobile



Name	X	Y
Chicago	35	42
Mobile	52	10
Toronto	62	77
Buffalo	82	65
Denver	5	45
Omaha	27	35
Atlanta	85	15
Miami	90	5

Quadtree - Beispielaufbau (2)

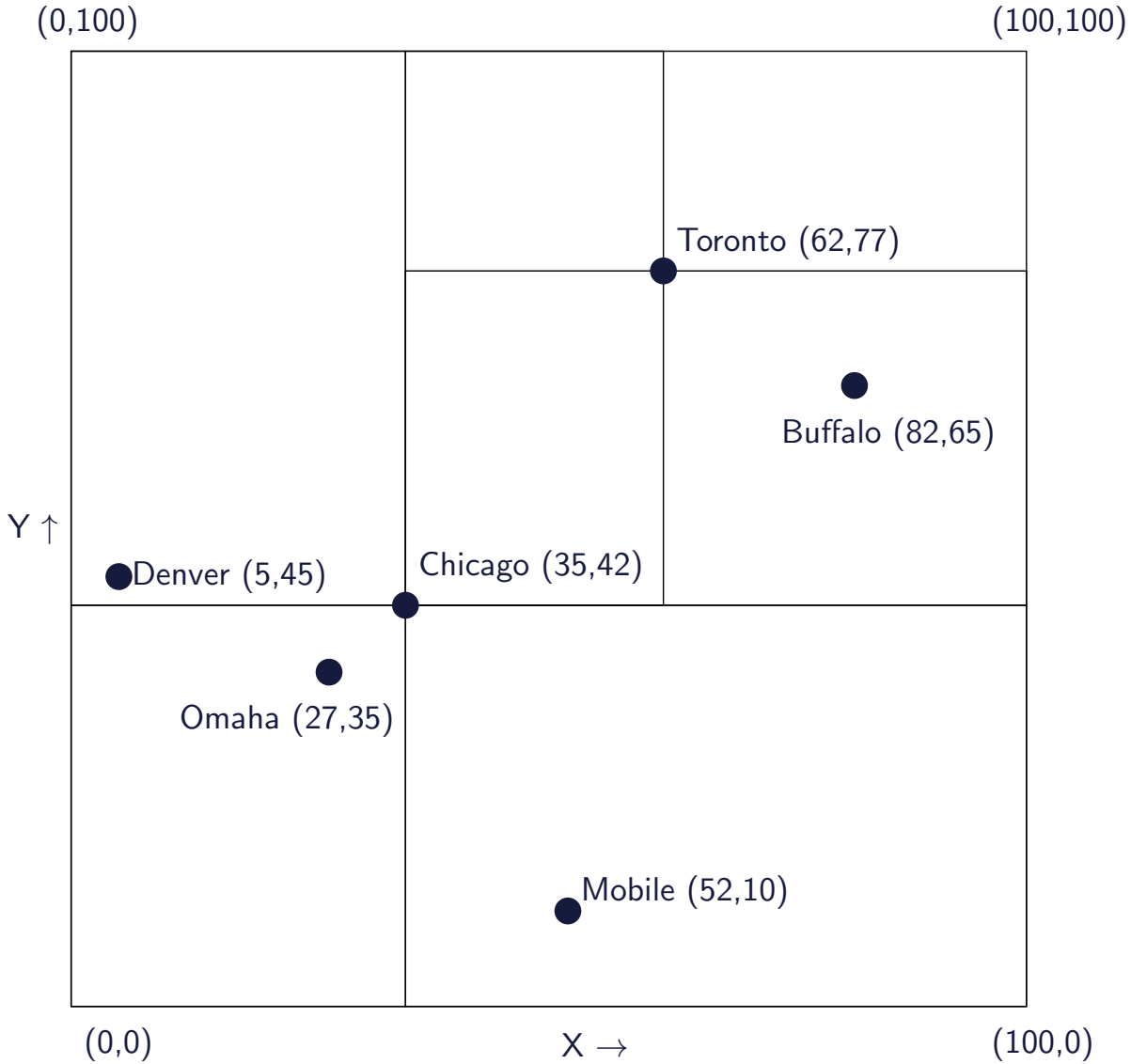
Einfügen von Toronto und Buffalo



Name	X	Y
Chicago	35	42
Mobile	52	10
Toronto	62	77
Buffalo	82	65
Denver	5	45
Omaha	27	35
Atlanta	85	15
Miami	90	5

Quadtree - Beispielaufbau (3)

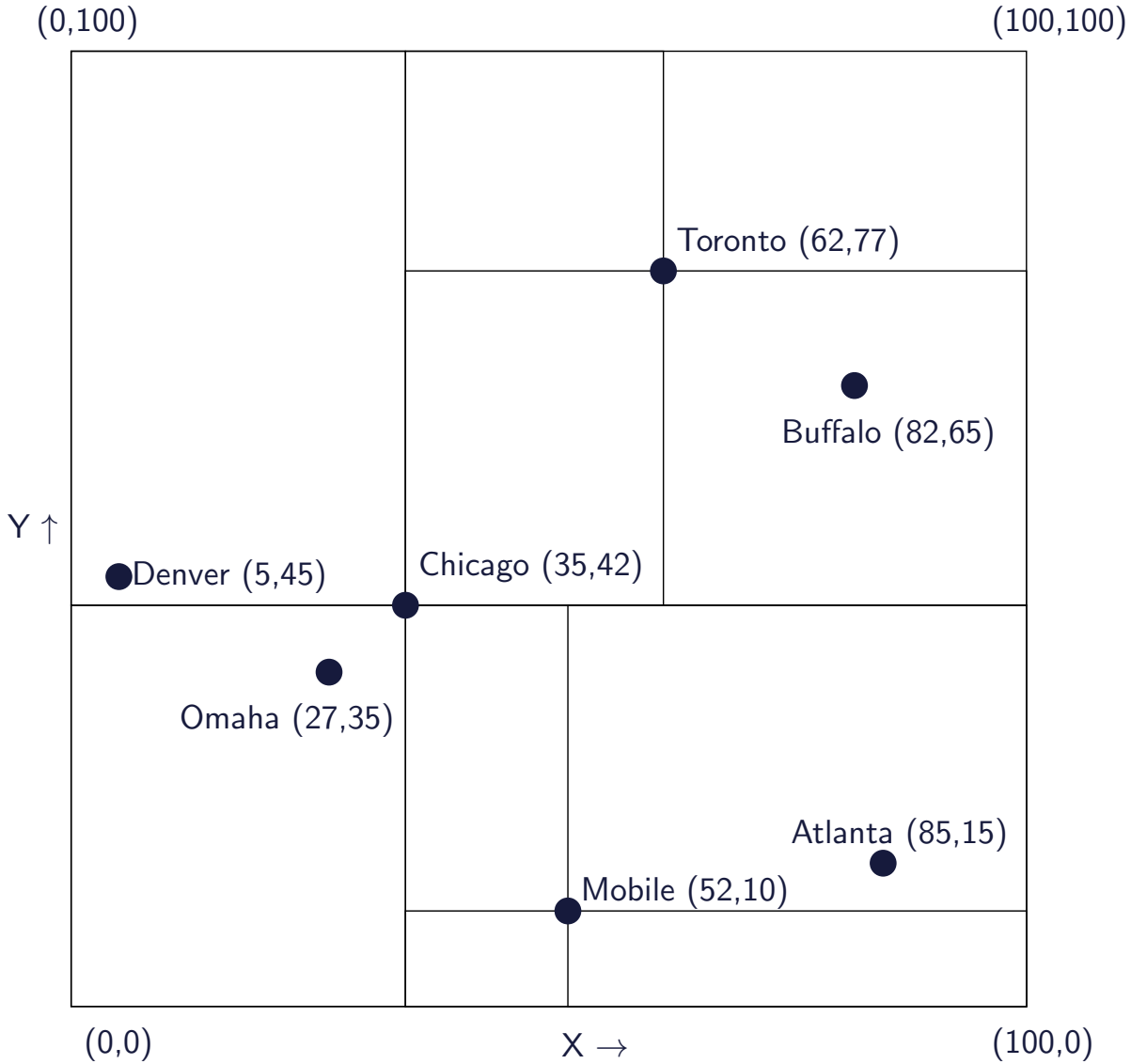
Einfügen von Denver und Omaha



Name	X	Y
Chicago	35	42
Mobile	52	10
Toronto	62	77
Buffalo	82	65
Denver	5	45
Omaha	27	35
Atlanta	85	15
Miami	90	5

Quadtree - Beispielaufbau (3)

Einfügen von Atlanta



Name	X	Y
Chicago	35	42
Mobile	52	10
Toronto	62	77
Buffalo	82	65
Denver	5	45
Omaha	27	35
Atlanta	85	15
Miami	90	5

Quadtree - Einfügen

Einfügen

Eingabe: ein Datensatz r mit Schlüssel (a, b)

- Ist der Baum leer?
  - Erzeuge neuen Knoten mit r und neuen Baum mit diesem Knoten
- Sonst:
  - Suche im Baum nach Schlüssel (a, b)
  - Existiert? Dann ersetze durch neuen Datensatz
  - Sonst:
    - Sind an einem NIL Knoten von Elternknoten n
    - Erzeuge Knoten mit r mit Schlüssel (a, b)
    - Hänge neuen Knoten unter n (in korrekter Quadranten!)

Quadtree - Löschen

Naiver Ansatz

- Entferne den Knoten aus Baum und füge alle Kinder-Knoten wieder ein.

Tries und PR-Quadtree

Gleichmäßige Aufteilung in Quadranten

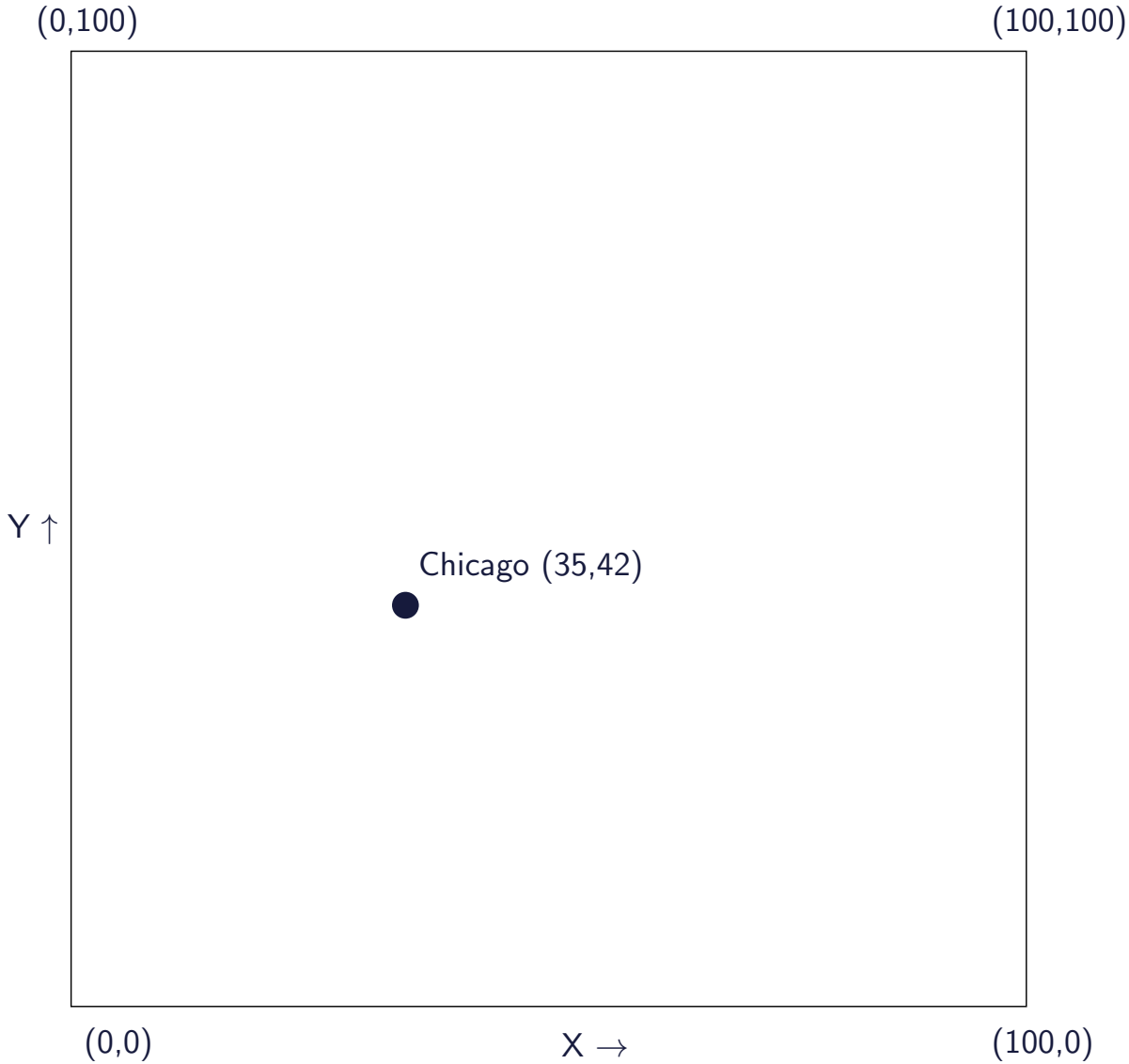
- Bei jedem Split wird der aufzuteilende Bereich in gleichmäßige Bereiche unterteilt
- Im Quadtree zuvor wurde anhand des eingefügten Datenpunktes aufgeteilt.
- Bäume, die diese feste Paritionierung haben werden auch tries genannt.

PR (=Point Region) Quadtree

- Müsste eigentlich PR Quadtrie heißen

PR-Quadtree - Beispiel

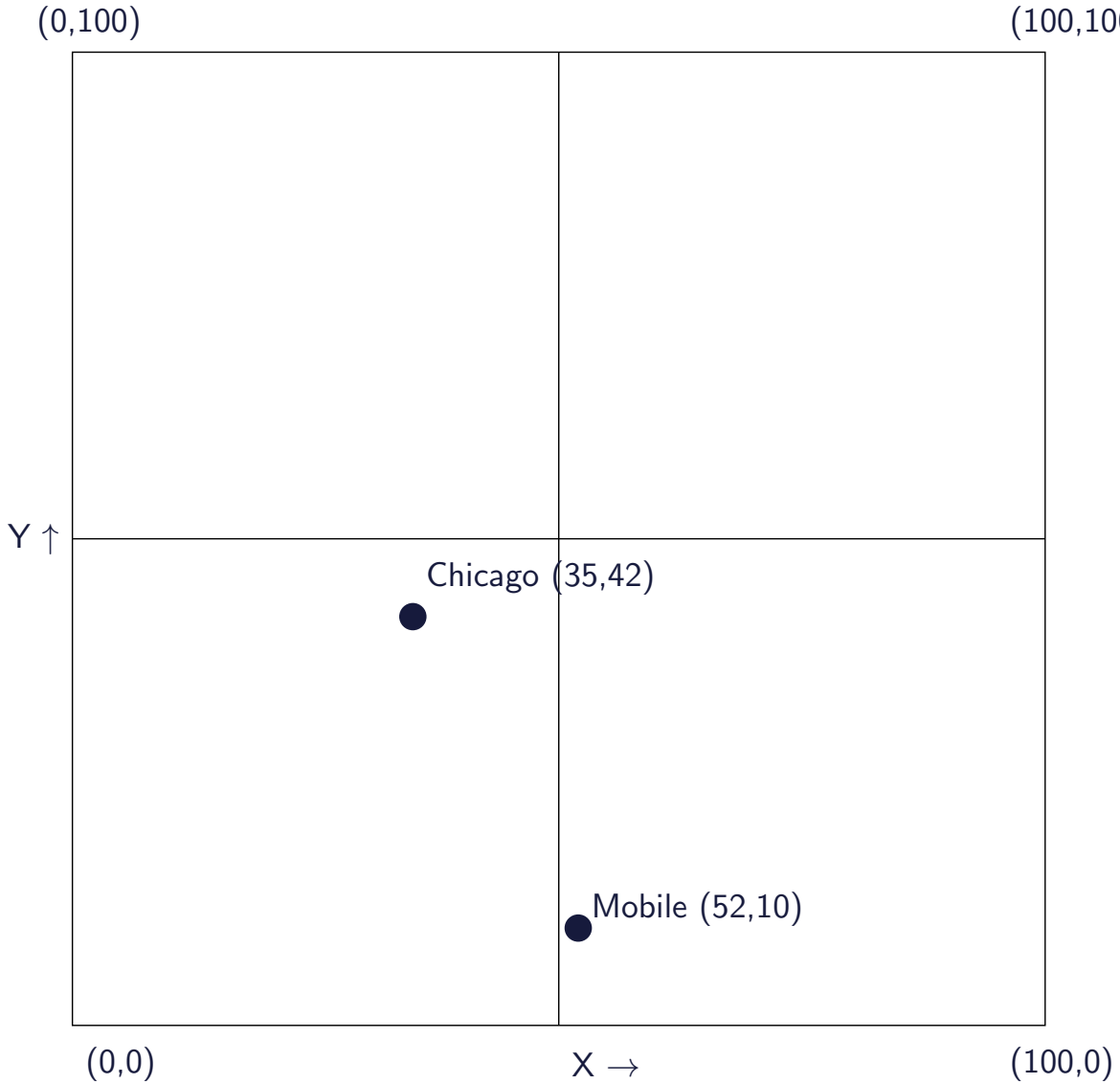
Einfügen von Chicago



Name	X	Y
Chicago	35	42
Mobile	52	10
Toronto	62	77
Buffalo	82	65
Denver	5	45
Omaha	27	35
Atlanta	85	15
Miami	90	5

PR-Quadtree - Beispiel

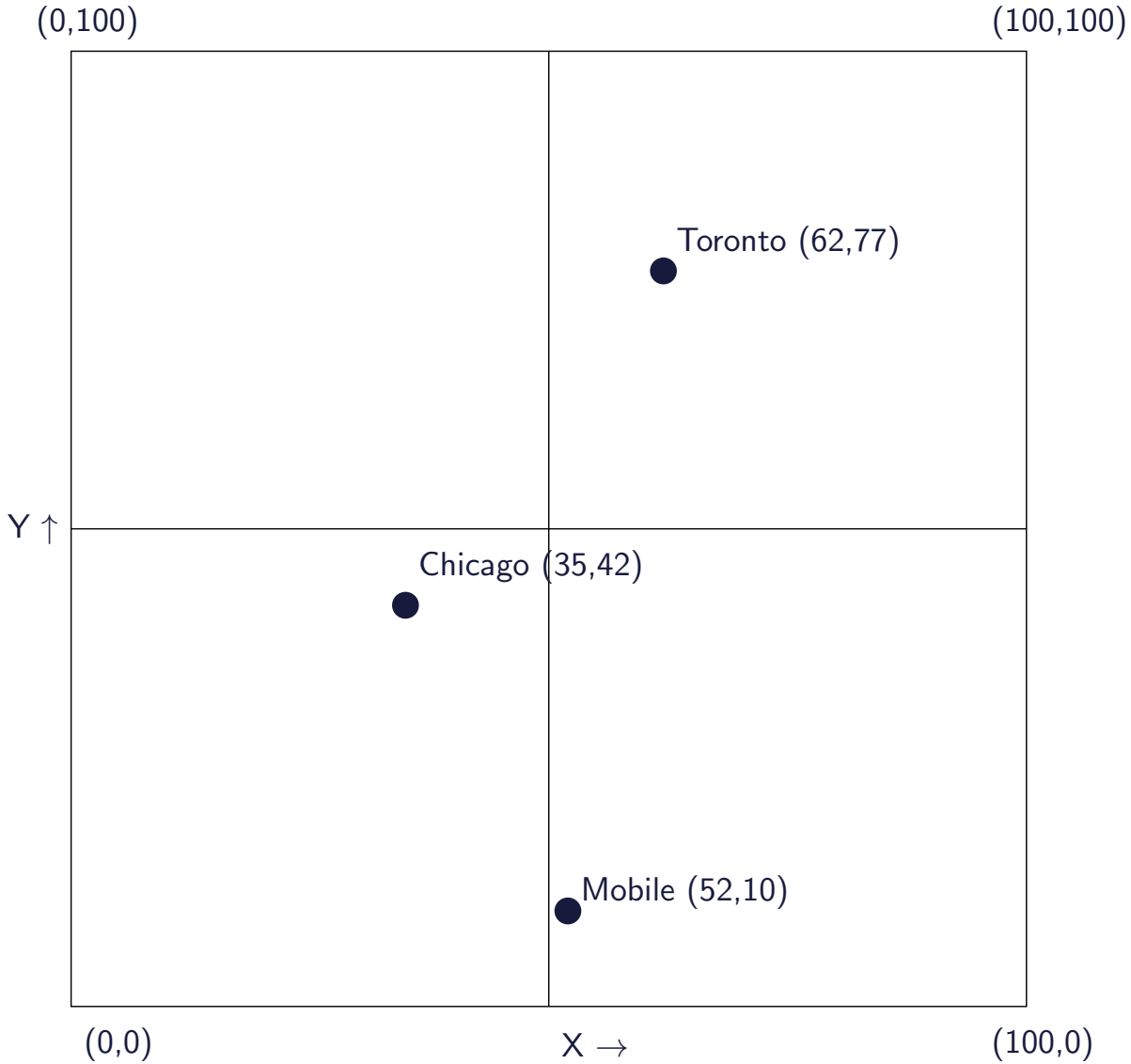
Einfügen von Mobile



Name	X	Y
Chicago	35	42
Mobile	52	10
Toronto	62	77
Buffalo	82	65
Denver	5	45
Omaha	27	35
Atlanta	85	15
Miami	90	5

PR-Quadtree - Beispiel

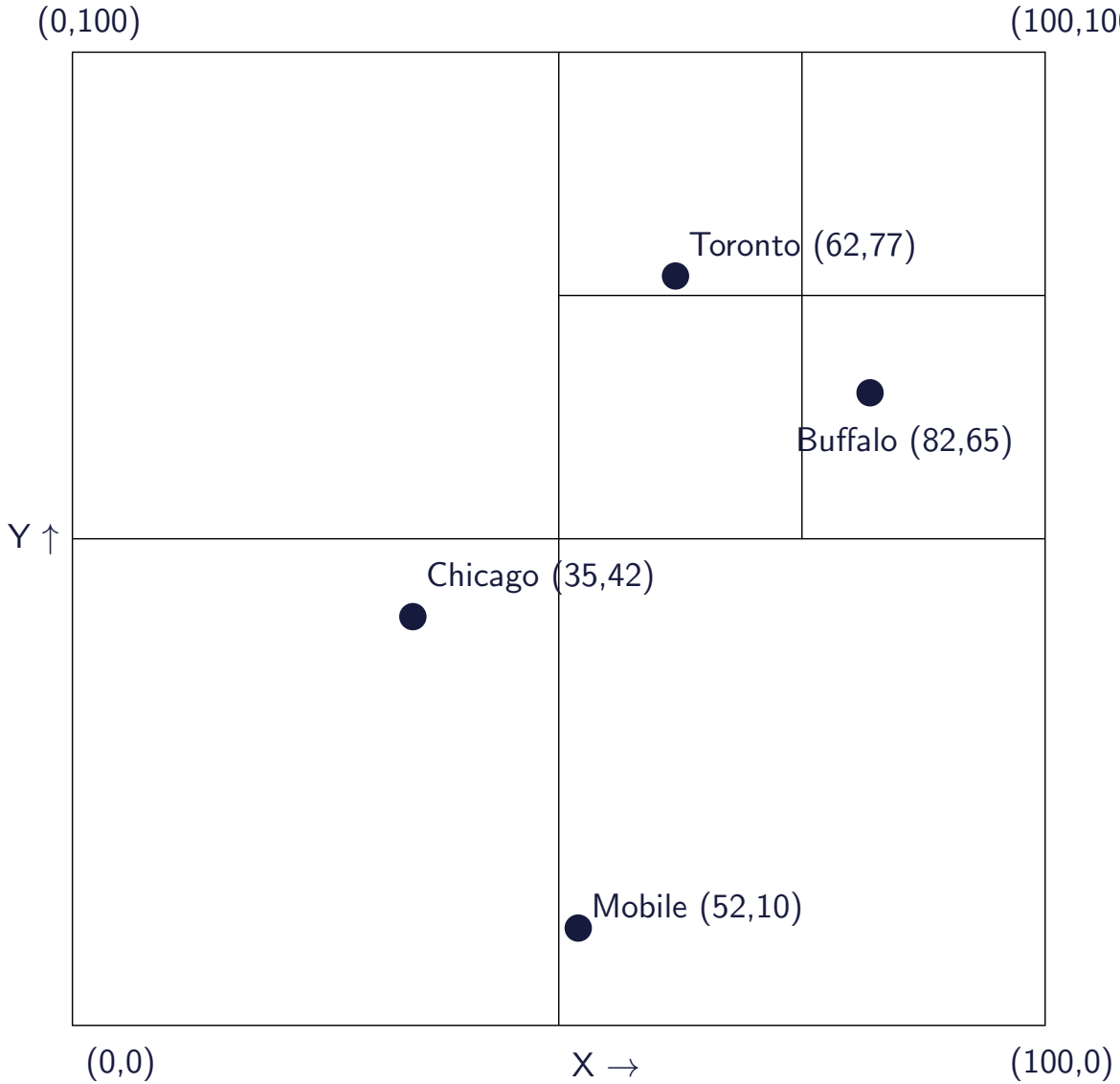
Einfügen von Toronto



Name	X	Y
Chicago	35	42
Mobile	52	10
Toronto	62	77
Buffalo	82	65
Denver	5	45
Omaha	27	35
Atlanta	85	15
Miami	90	5

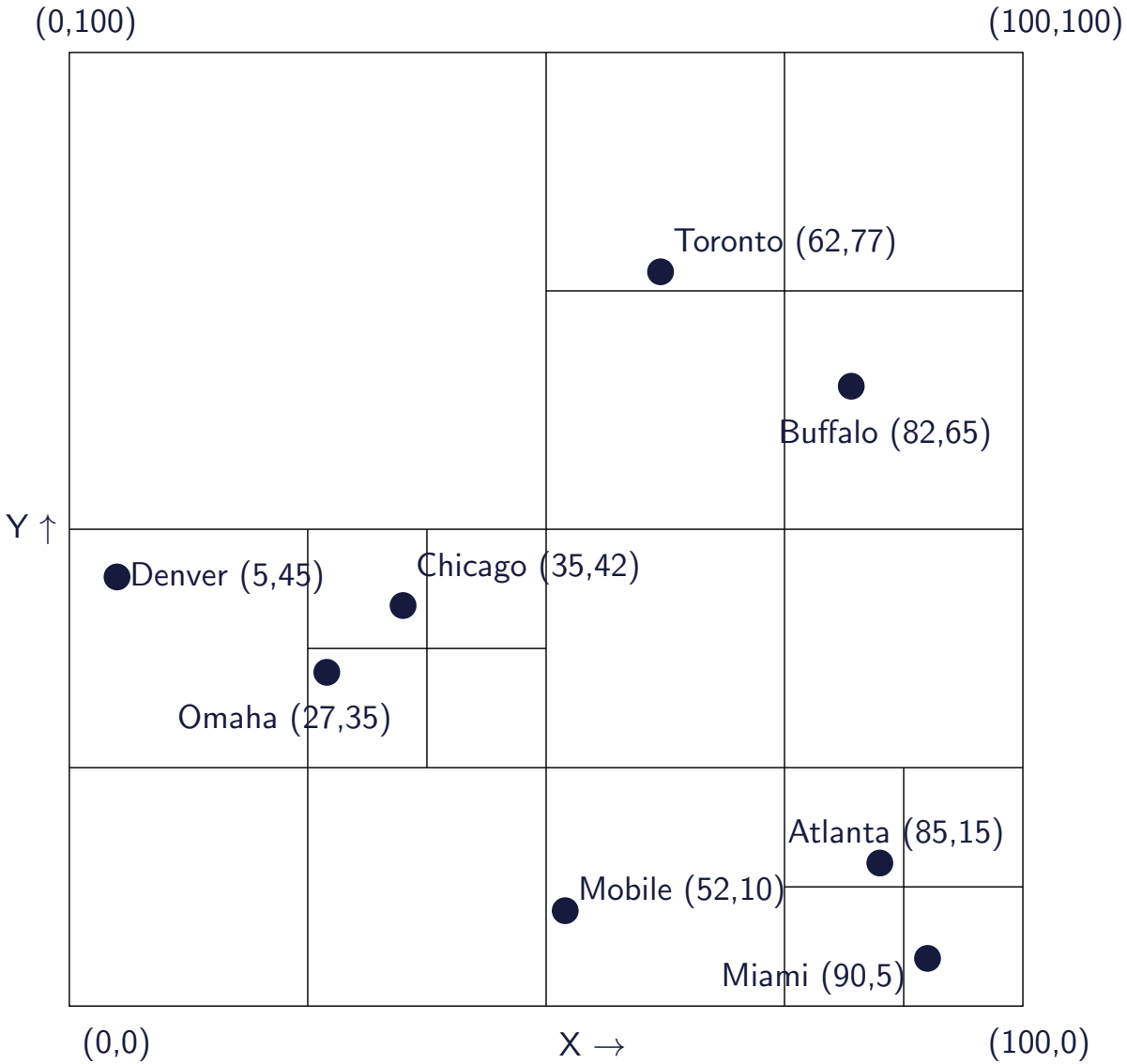
PR-Quadtree - Beispiel

Einfügen von Buffalo

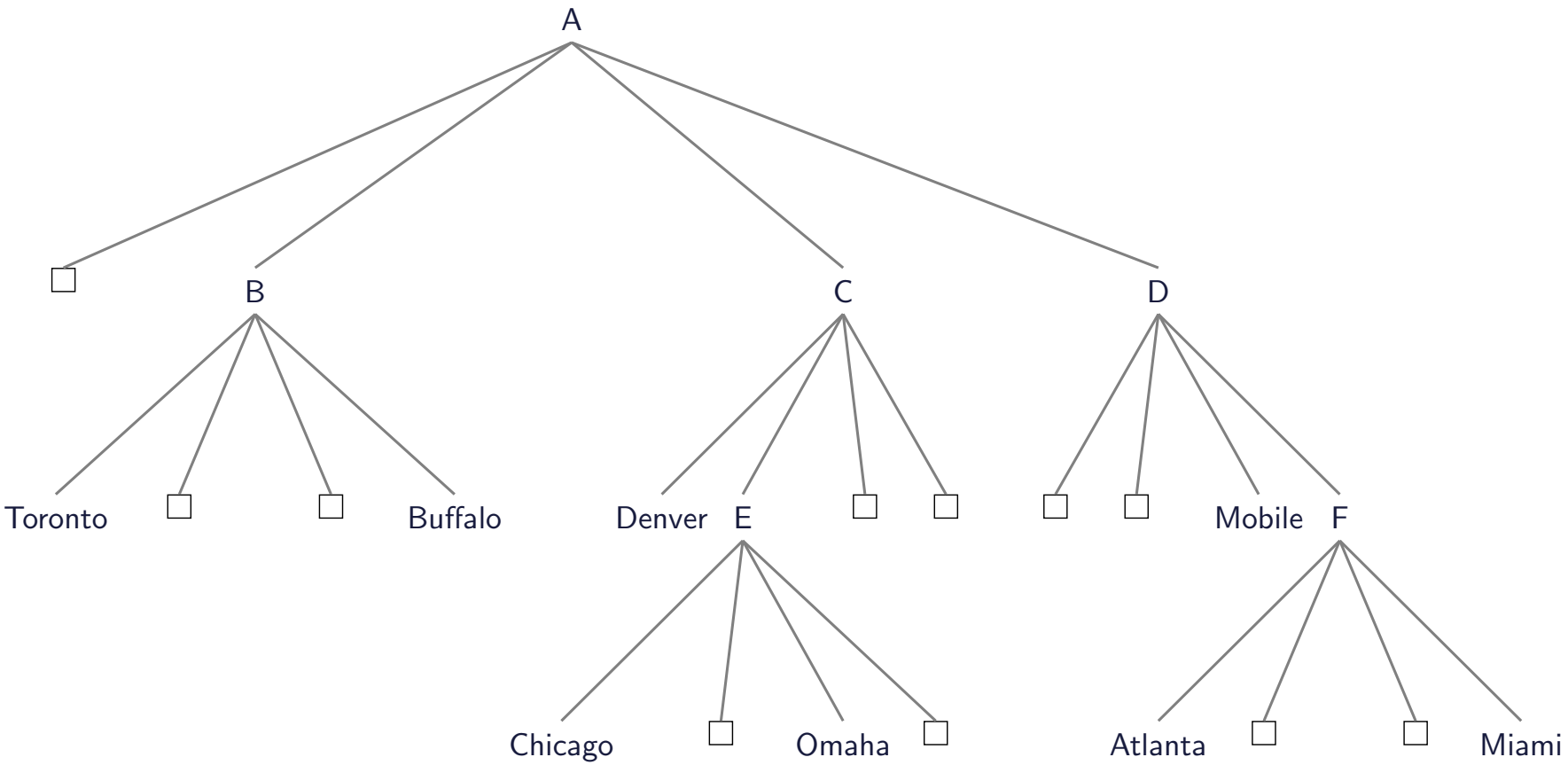


Name	X	Y
Chicago	35	42
Mobile	52	10
Toronto	62	77
Buffalo	82	65
Denver	5	45
Omaha	27	35
Atlanta	85	15
Miami	90	5

PR-Quadtree - Beispiel



PR Quadtree - Baum zum Beispiel



PR Quadtree - Einfügen

- Um einen Datenpunkt (a,b) in den Baum einzufügen wird nach diesem Punkt gesucht, bzw. nach dem Quadranten (Blatt), in dem er liegt
- Dies geschieht rekursiv von oben nach unten (anhand der Quadranten)
- Ist das gefundene Blatt schon belegt, wird neu aufgeteilt, bis sie in unterschiedlichen Teilen liegen
- Dieses rekursive Aufteilen kann sehr häufig auftreten für ein Einfügen

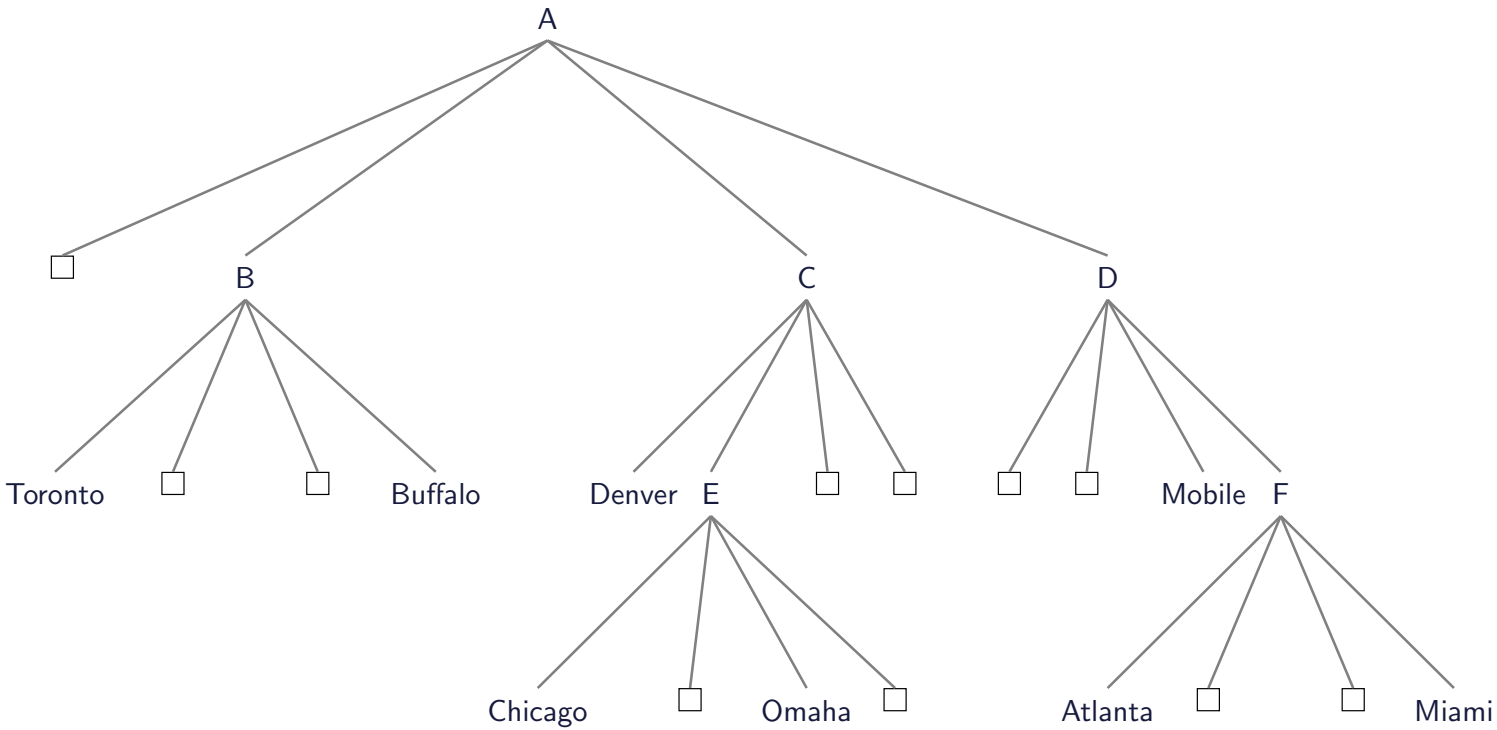
Anmerkung

- Die inneren Knoten brauchen keine Informationen zu speichern welchen Raum (Bereich) sie aufspannen
- Da dies einfach berechnet werden kann (da Raum im Gegensatz zum vorherigen Quadtree gleichmässig aufgespannt wird)

Diskussion: Quadtree vs. PR-Quadtree

- Aussehen des Quadtrees ist stark von der Reihenfolge des Einfügens der einzelnen Daten abhängig
- Im Gegensatz dazu ist das Aussehen des PR-Quadtrees unabhängig von der Reihenfolge des Einfügens
- Im PR-Quadtree werden Daten nur noch in den Blättern gespeichert und innere Knoten enthalten nur Verweise
- Das macht das Löschen einfacher

PR-Quadtree - Löschen



- Daten nur in Blättern: Löschen von Mobile kein Problem, einfach SW Eintrag von D auf NIL setzen
- Aber, Löschen von Toronto macht Probleme: Setzten des NW Eintrags im Elternknoten B verletzt Eigenschaft, dass jeder innere Knoten mindestens zwei Blattknoten mit Daten umfassen muss.

kd-tree - Motivation

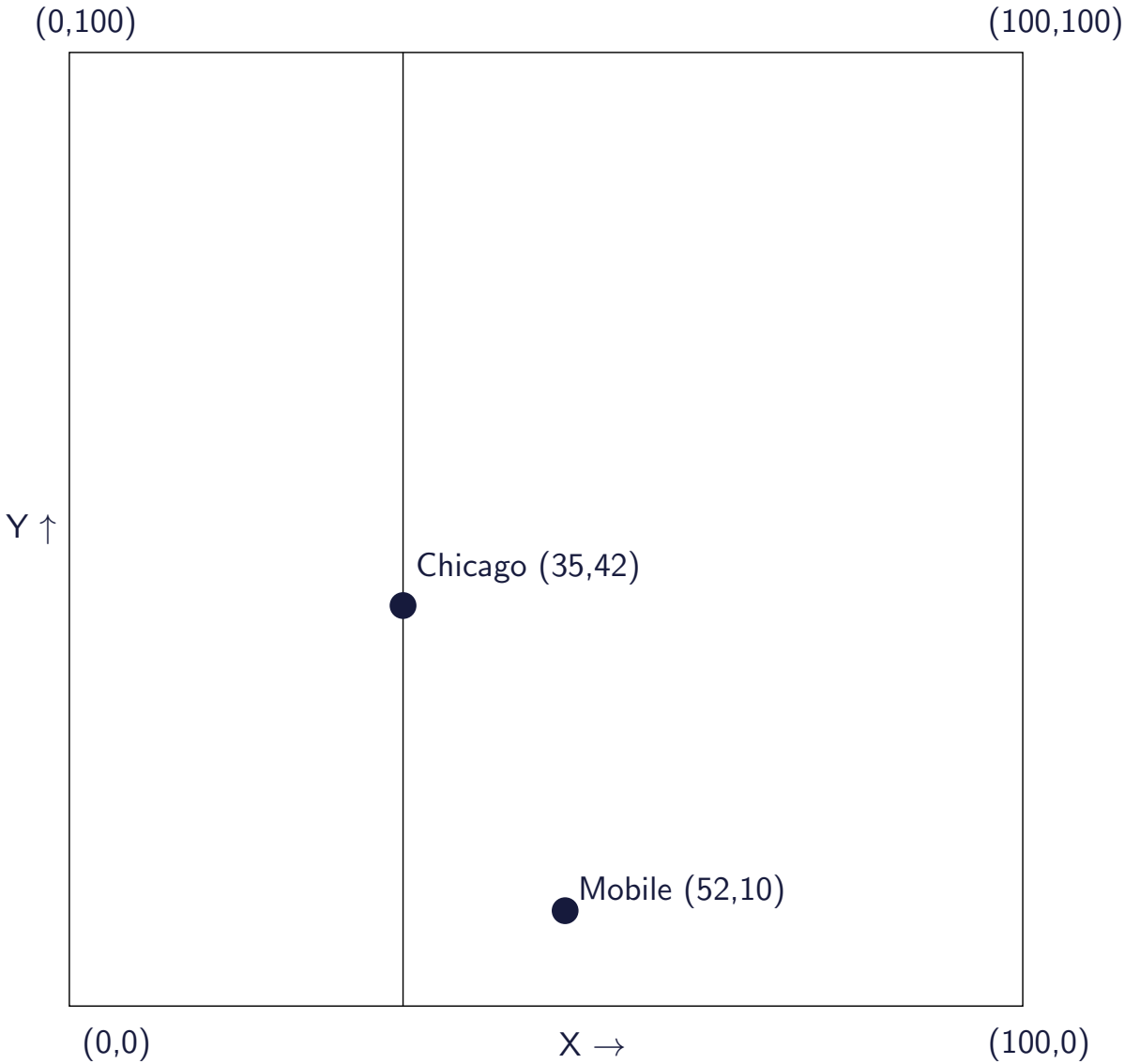
Beobachtungen

- Für höhere Dimensionen d wird der Raum in jedem Knoten des Baums in sehr viele ($=2^d$) Zellen aufgeteilt.

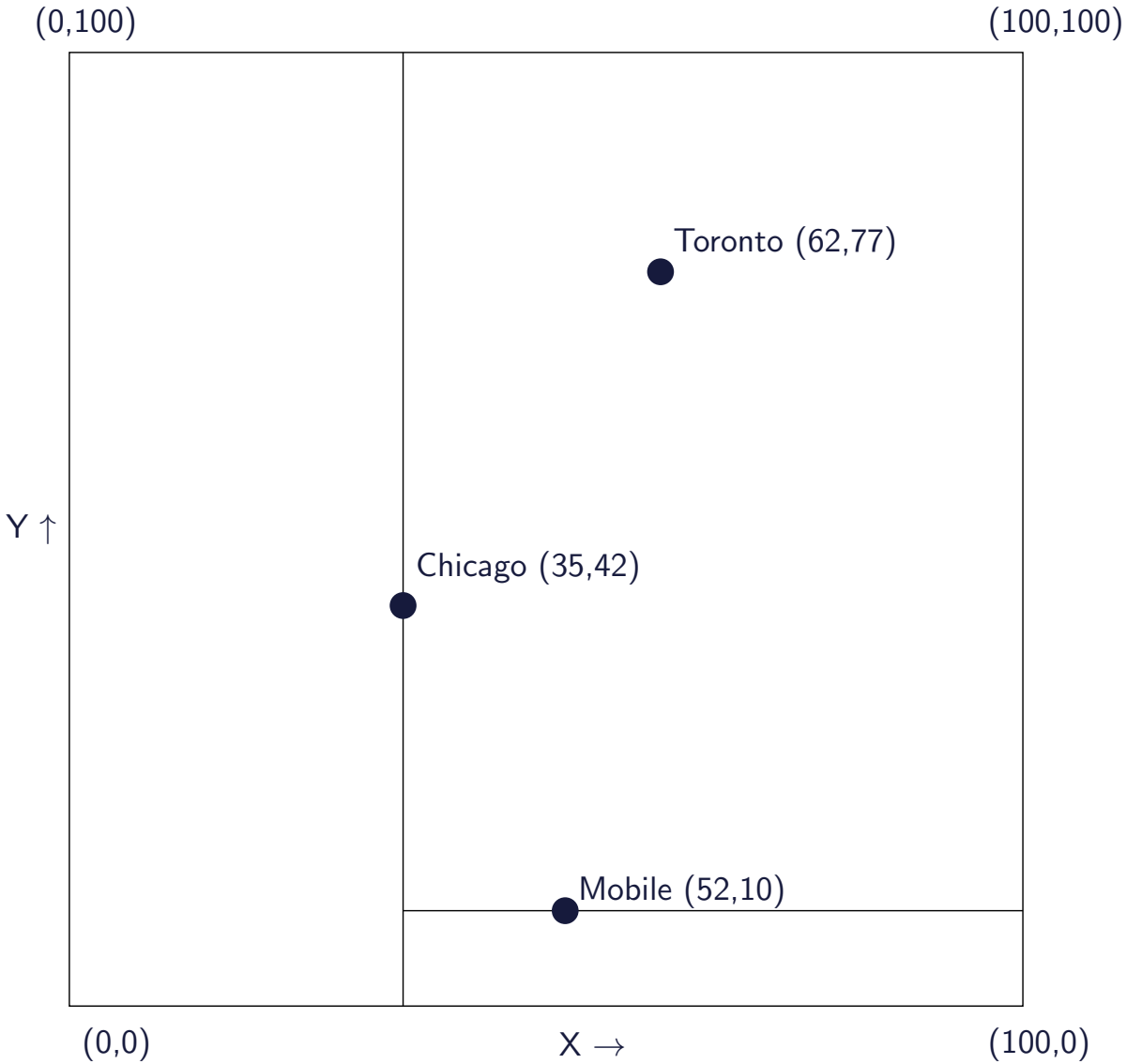
Idee hinter kd-tree

- Aufteilung in jedem Schritt nur nach einer Dimension
- Dimensionen wechseln sich ab

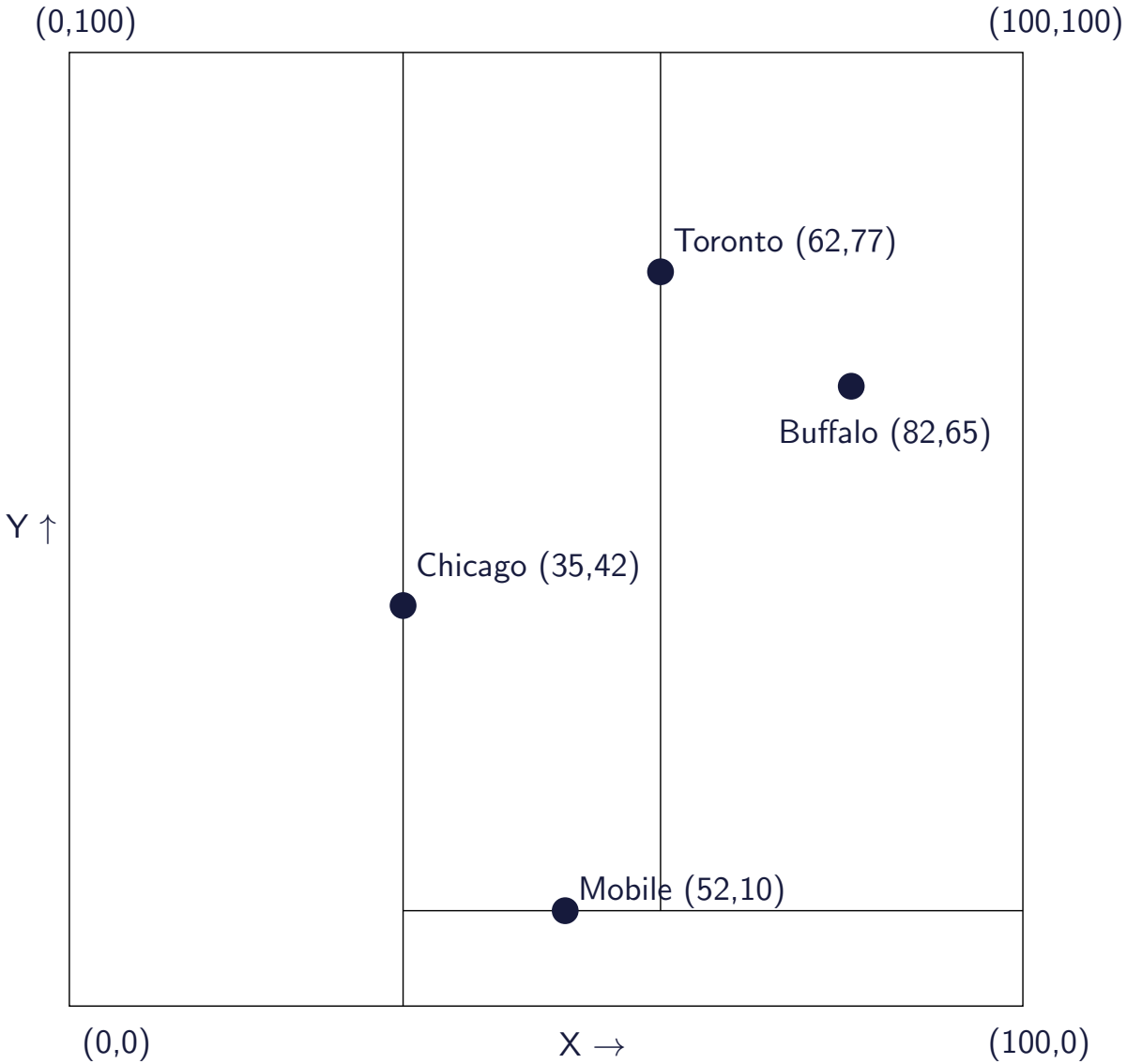
kd-tree - Beispiel



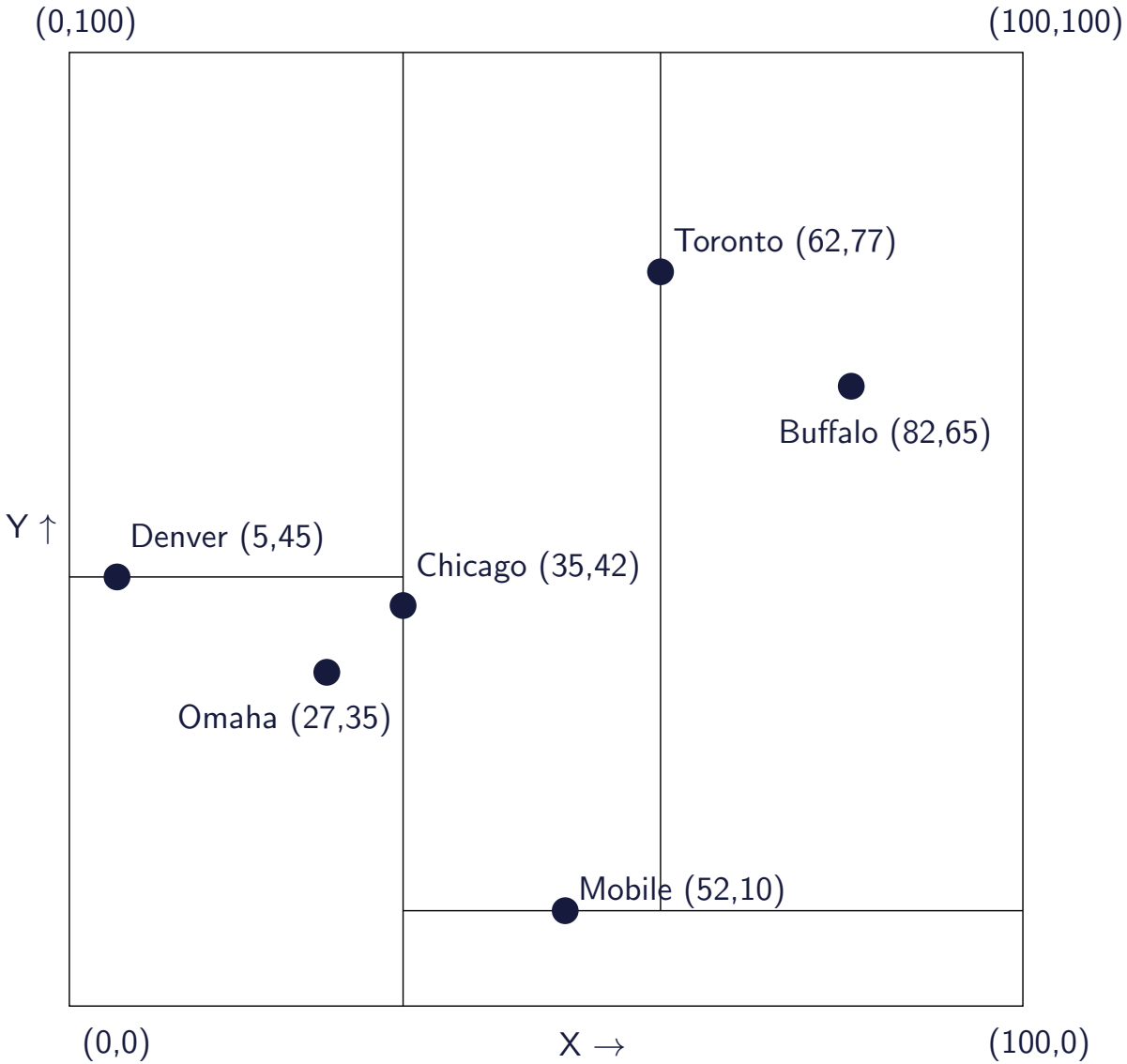
kd-tree - Beispiel (2)



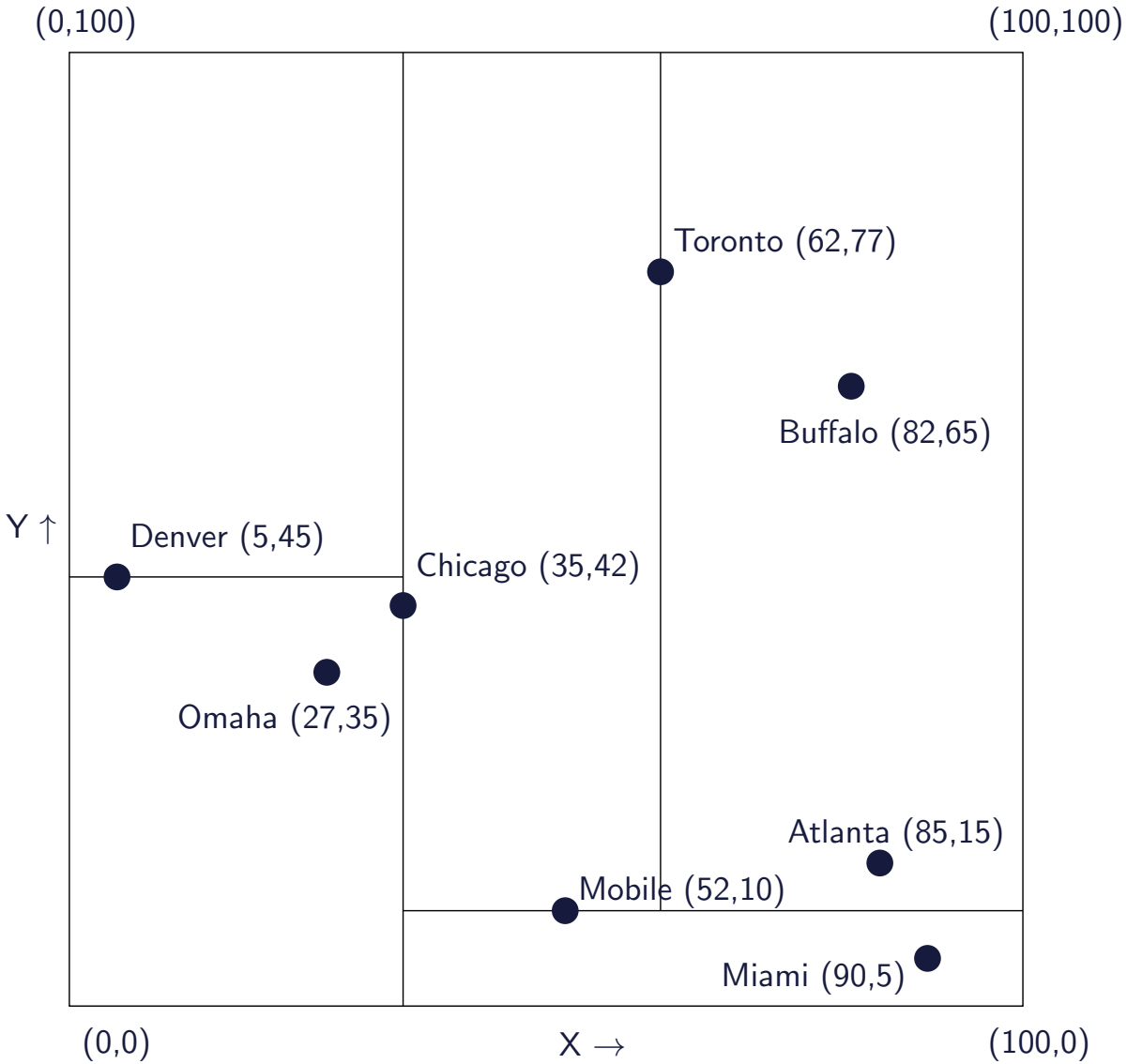
kd-tree - Beispiel (3)



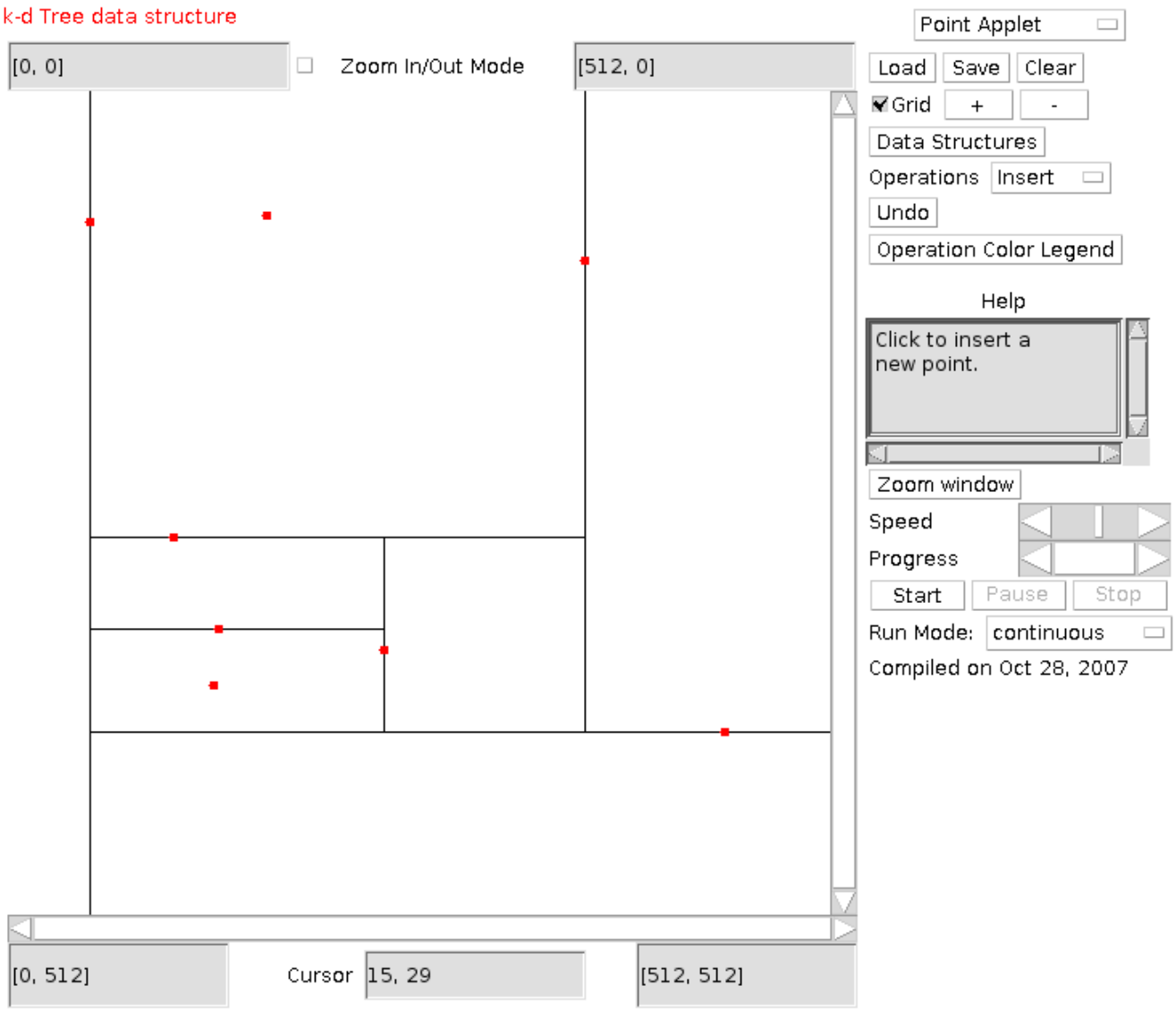
kd-tree - Beispiel (4)



kd-tree - Beispiel (5)



kd-tree - Demo Seite



http://donar.umiacs.umd.edu/quadtree/points/kdtree.html

Octtree - "Quadtree" in im Falle von 3 Dimensionen

Ein Quadtree für $d = 3$ Dimensionen funktioniert ähnlich zu dem betrachteten Quadtree in 2 Dimensionen:

- Anstelle von $2^2 = 4$ Quadranten werden im 3-dimensionalen Raum $2^3 = 8$ Unterteilungen (des, hier, Würfels) benötigt.

Raumfüllende Kurven (Space Filling Curves)

Aka. Peano-Kurven

Idee

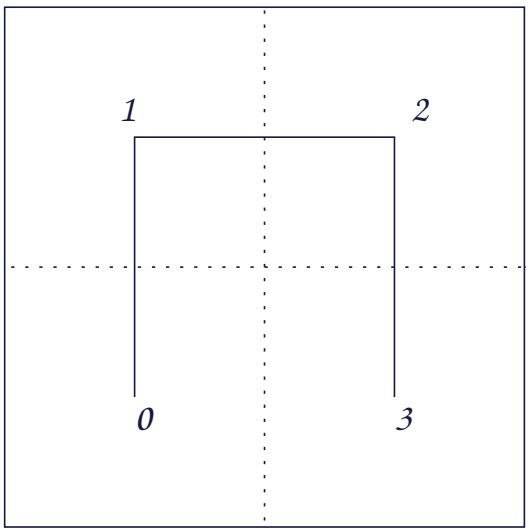
- Bilde d-dimensionalen Raum ab auf 1-dimensionalen Raum
- Dazu wird eine raumfüllende Kurve betrachtet
- Ein Punkt wird dann durch Länge der Kurve bis zu diesem Punkt beschrieben

2D Fall

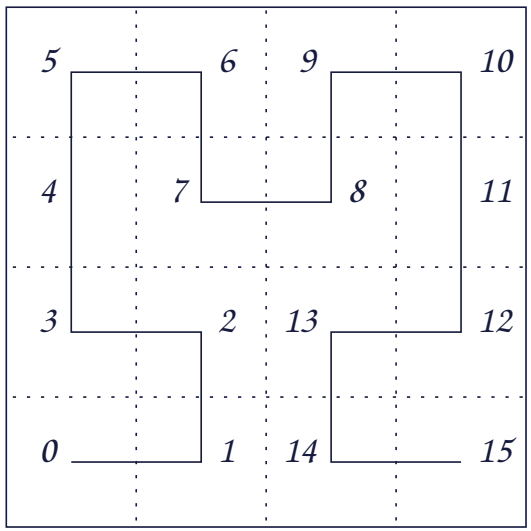
- Gegeben ein Punkt $P_1 = (x_1, y_1)$.
- Die Länge der Kurve bis sie P_1 erreicht sei m
- Dann sollte ein Punkt $P_2 = (x_2, y_2)$ der nah an Punkt P_1 liegt auch einen ähnlichen Wert für m haben

Hilbert-Kurve

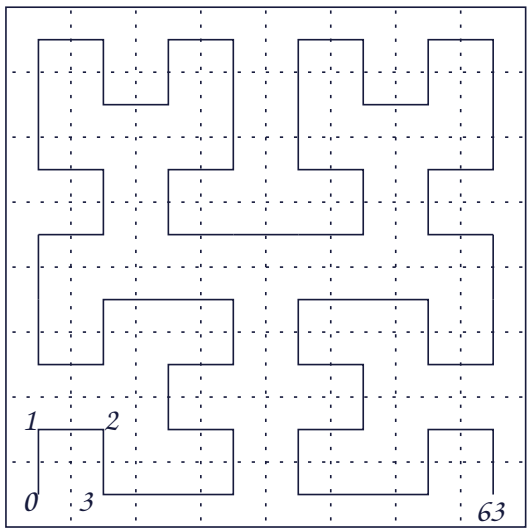
Hilbert-Kurven der Ordnung $k = 1$ (a), $k = 2$ (b) und $k = 3$ (c). Anzahl der nummerierten "Quadrate" ist 2^{k*d} (d =Anzahl Dimensionen)



(a)



(b)



(c)

D. Hilbert: Über die stetige Abbildung einer Linie auf ein Flächenstück. Mathematische Annalen 38 (1891).

Abbildung und Implementierungs-Details aus/in: J. K. Lawder. Calculation of Mappings Between One and n-dimensional Values Using the Hilbert Space-Filling Curve.

Z-Kurve (aka. Morton-Kurve)

Z-Kurven der Ordnungen 1, 2, 3 und 4.

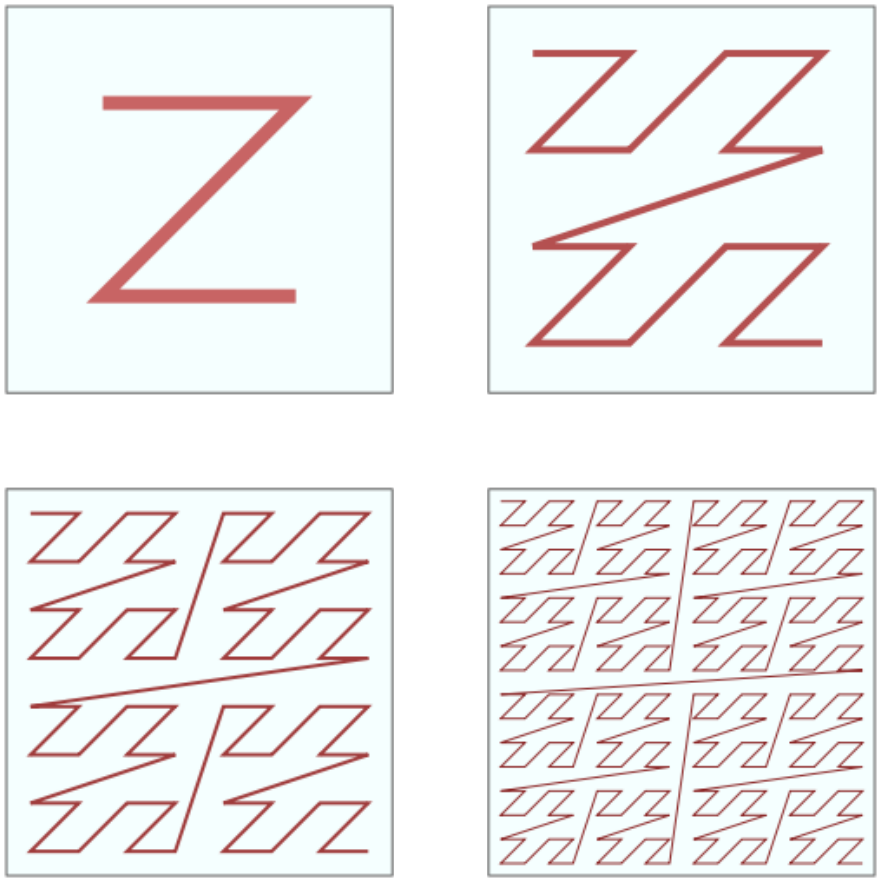


Abbildung: Wikipedia

Z-Kurve - Beispiel Aufzählung

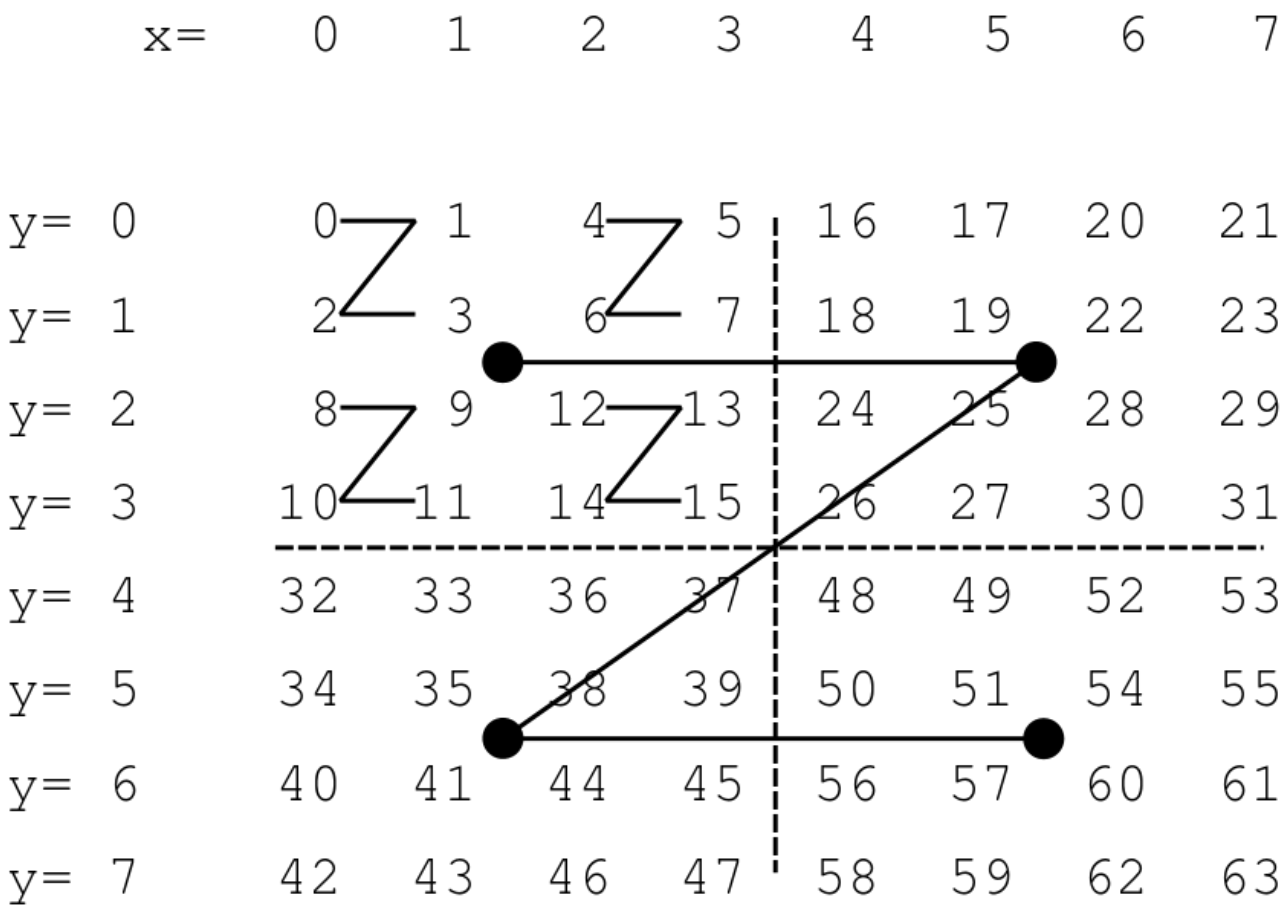


Abbildung: Wikipedia

Z-Kurve - Beispiel Berechnung

Berechnung der Z-Werte durch bitweise Verschränkung (Bit-Interleaving)

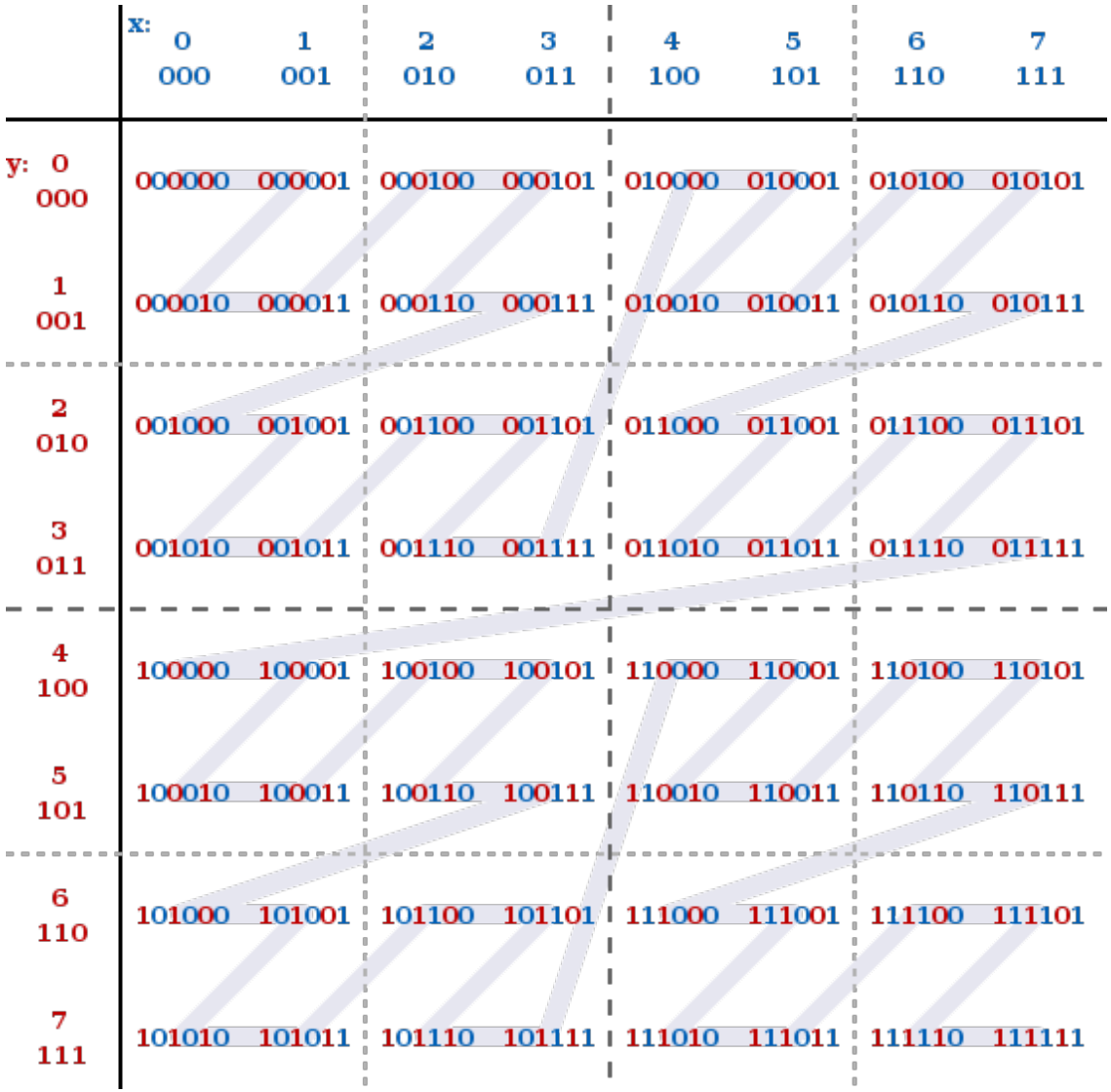


Abbildung: Wikipedia