TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS18/19

Moritz Sichert, Lukas Vogel (gdb@in.tum.de)
https://db.in.tum.de/teaching/ws1819/grundlagen/

Blatt Nr. 10

Hausaufgabe 1

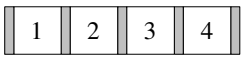
(a) Fügen Sie in einen anfänglich leeren B$^+$-Baum mit $k = 3$ und $k^* = 2$ die Zahlen eins bis fünfundzwanzig in aufsteigender Reihenfolge ein. In den Blattknoten werden TIDs verwendet. Was sind TIDs, wann lohnt sich ihre Verwendung, was ist die Alternative zu TIDs?

(b) Erläutern Sie die Vorgehensweise bei der Bearbeitung der folgenden Anfrage „Finde alle Datensätze mit einem Schlüsselwert zwischen 5 und 15.“

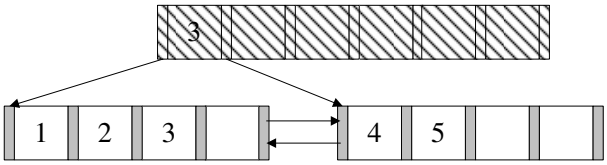
Lösung:

(a) Bei B$^+$-Bäumen unterscheiden sich die Kapazitäten von inneren Knoten und Blattknoten (angegeben durch k und k^*). Im Folgenden werden innere Knoten zur leichteren Unterscheidung schraffiert.

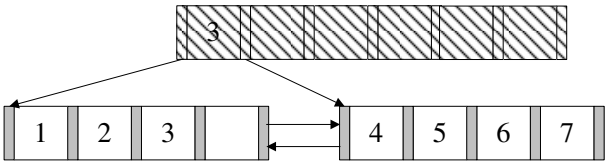
Nachdem man die Zahlen 1 bis 4 eingefügt hat, liegt folgender B-Baum vor (da die Wurzel in diesem Fall ein Blatt ist können höchstens 4 Einträge eingefügt werden):



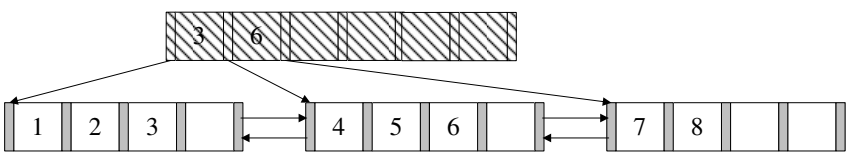
Beim Einfügen von 5 wird der Knoten gespalten. Der Referenzschlüssel 3 wandert in die neue Wurzel, deren Kapazität 6 ist. Die neuen Blattknoten haben eine Kapazität von 4 und sind untereinander verlinkt.



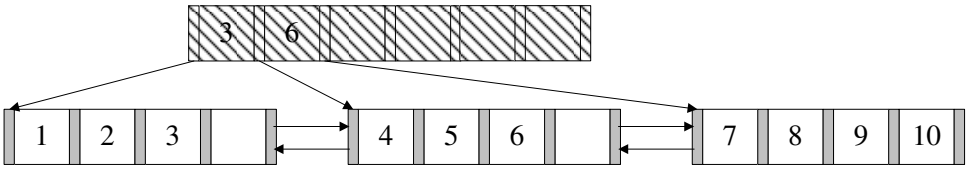
Die nächsten beiden Einträge lassen sich wieder ohne Probleme einfügen.



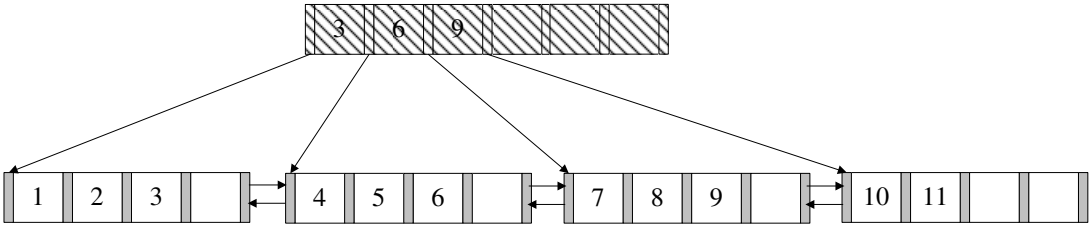
Beim Einfügen der 8 kommt es erneut zum Überlauf. Die 6 wandert in die Wurzel.



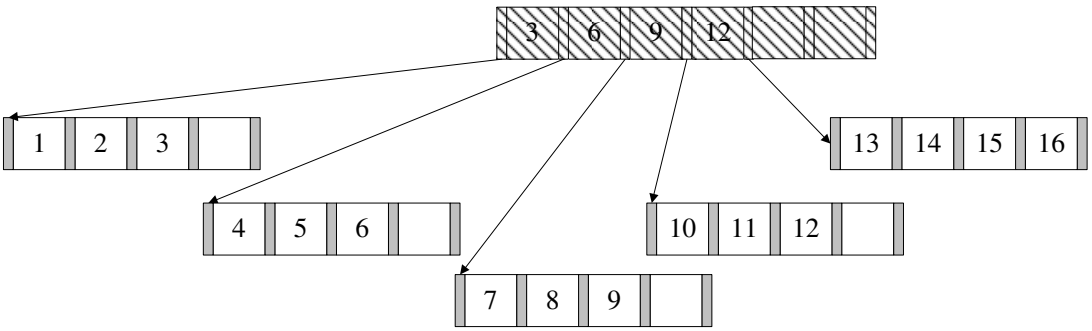
9 und 10 lassen sich wieder ohne Probleme einfügen. Bei 11 kommt es zum Überlauf.



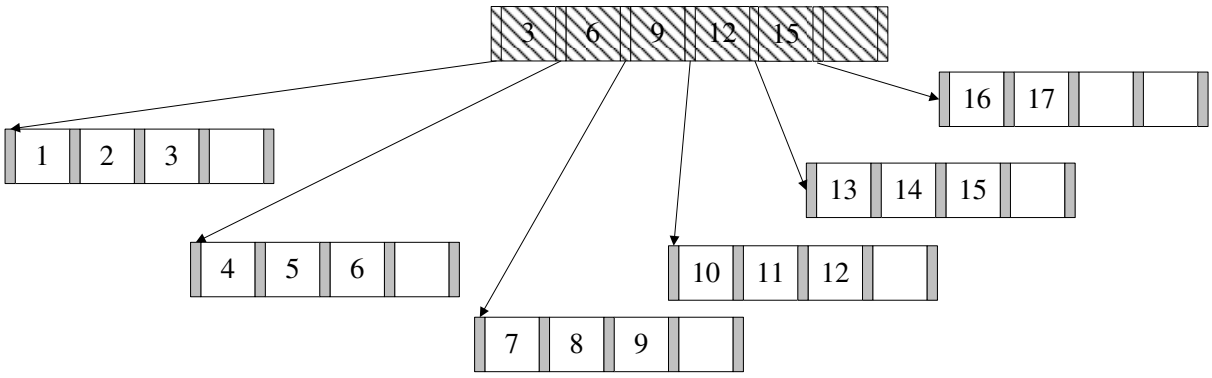
Nach dem Aufspalten erhält man dann:



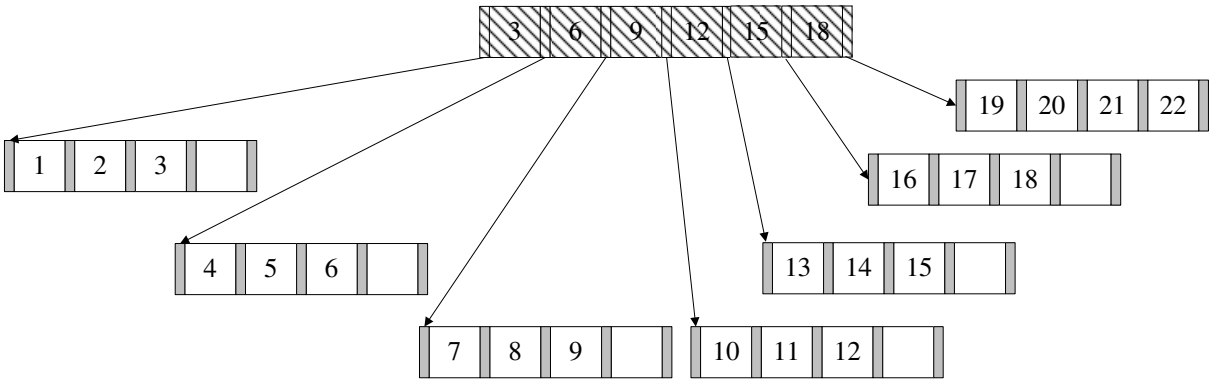
Es werden nun die nächsten Zahlen bis 16 analog eingefügt. (Die Pointer zwischen den Blattknoten existieren weiterhin, werden hier jedoch nicht mehr dargestellt.)



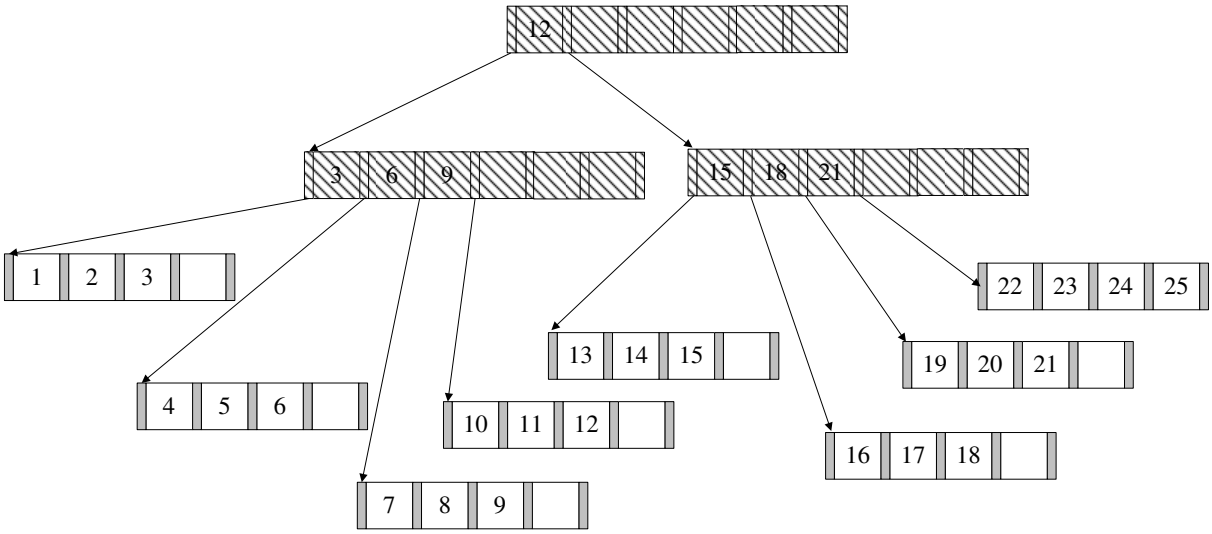
Bei 17 kommt es dann wieder zum Überlauf.



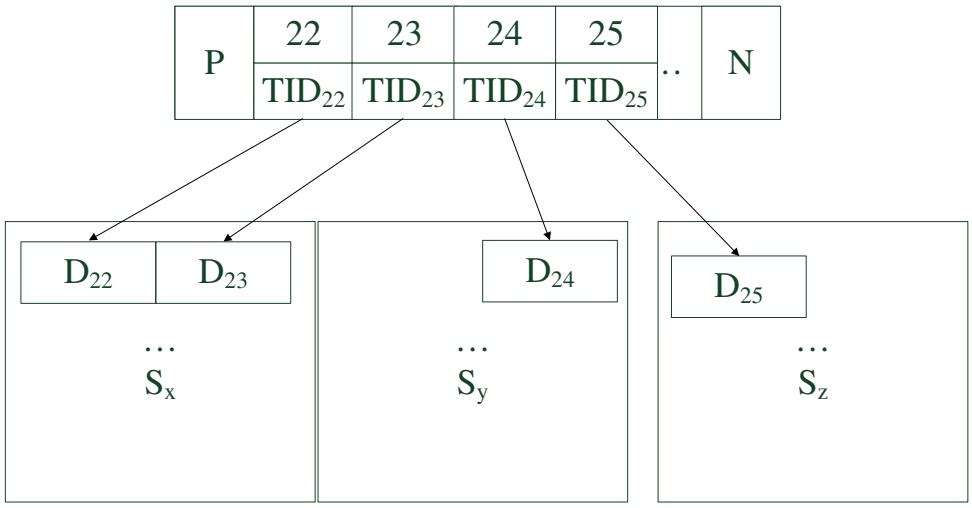
Die nächsten Zahlen werden wieder analog eingefügt. Bei 20 kommt es zum Überlauf.



Nun fügt man noch die Zahlen 21 und 22 ein. Bei 23 kommt es erneut zum Überlauf. Die 21 wird in den Wurzelknoten kopiert, wodurch auch hier ein Überlauf stattfindet, so dass der Baum in seiner Höhe wächst. Nach dem Einfügen von 24 und 25 sieht der Baum wie folgt aus:



In den Blättern können nun Datensätze oder TIDs gespeichert werden. TIDs sind „Zeiger“ auf Tupel. Werden TIDs vewendet, wird der Index kompakter und dadurch der Baum weniger hoch. Dafür ist eine weitere Indirektion zum Auffinden der Daten erforderlich, die bei der Suche nach einem Tupel verfolgt werden muss. Der letzte Knoten würde (im Detail) so aussehen:



S_x, S_y und S_z sind dabei beliebige Seiten im Speicher.

(b) Um eine Bereichsanfrage zu beantworten geht man wie folgt vor:

1. Zunächst sucht man nach der unteren Schranke der Anfrage, in diesem Fall nach der 5. Dies geschieht genauso wie beim B-Baum. Die Suche endet in einem Blattknoten.
2. Anschließend liest man alle sukzessiven Einträge bis zur oberen Schranke der Anfrage, in diesem Fall 15. Hierbei nutzt man die *Next*-Verlinkungen der Blattknoten untereinander.

Natürlich wäre es umgekehrt auch möglich, nach der oberen Schranke zu suchen und dann den *Previous*-Pointern zu folgen.

Hausaufgabe 2

Fügen Sie nacheinander die folgenden Einträge in eine anfangs leere erweiterbare Hashtabelle, welche 2 Einträge pro Bucket aufnehmen kann, ein. Es soll effizient nach der `KundenNr` gesucht werden können.

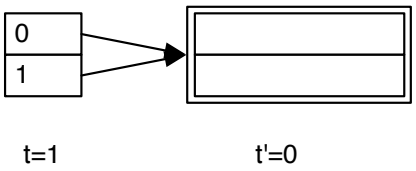
KundenNr	Name
10	Müller
25	Meier
30	Schmidt
18	Krause
40	Schulz
45	Kaufmann

Lösung:

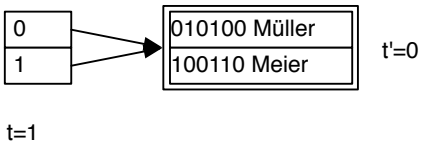
Werte mit binärer KundenNr sowie invers binärer Kundennummer, die für das Einfügen in den Hash genutzt wird:

KundenNr	Name	Binär	Umgekehrt Binär
10	Müller	001010	010100
25	Meier	011001	100110
30	Schmidt	011110	011110
18	Krause	010010	010010
40	Schulz	101000	000101
45	Kaufmann	101101	101101

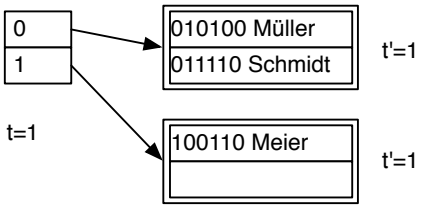
Zunächst eine leere erweiterbare Hashtabelle:



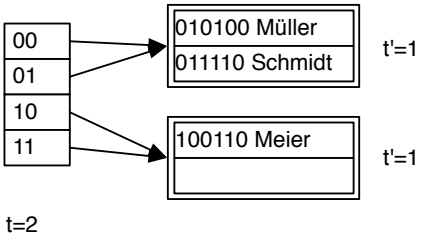
Wir fügen nun die ersten zwei Einträge ein, wonach die Hashtabelle wie folgt aussieht:



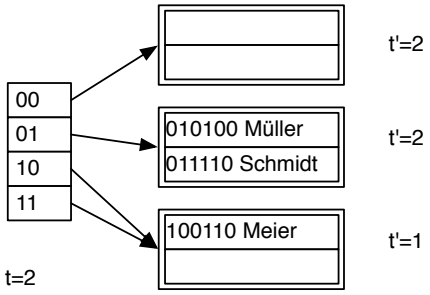
Der nächste Eintrag führt zu einem Überlauf. Da $t' < t$, können wir den Bucket teilen. Dies führt zur folgenden Hashtabelle:



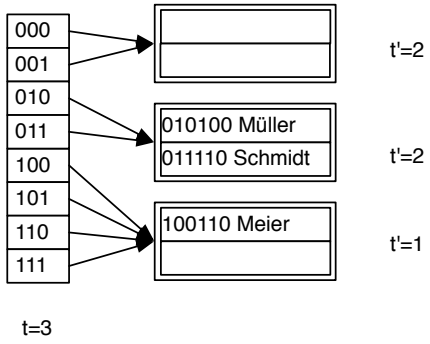
Das Einfügen von `Krause` führt erneut zu einem Überlauf. Da $t' = t$, können wir den Bucket aber nicht direkt teilen. Das Verzeichnis wird verdoppelt:



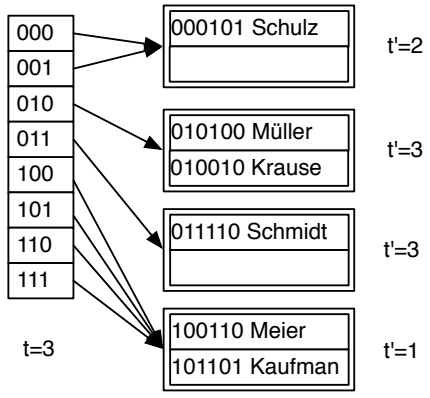
Nun kann der Bucket geteilt werden:



Das Einfügen ist leider immer noch nicht möglich und wieder gilt $t' = t$, weswegen das Verzeichnis erneut verdoppelt werden muss:



Nun kann der Bucket geteilt und alle Einträge eingefügt werden:



Hausaufgabe 3

Fügen Sie in einen anfangs leeren R-Baum mit Knotenkapazität 4 folgende Datenpunkte nacheinander ein:

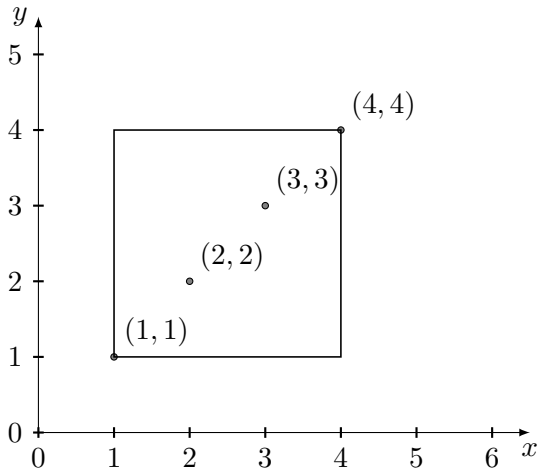
$$(1,1), (2,2), (3,3), (4,4), (5,3), (5,4), (6,5), (4,2)$$

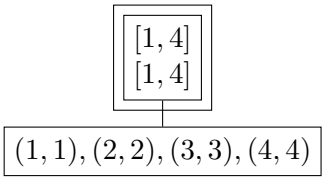
Splitten Sie die Knoten dabei so, dass die summierte Fläche der durch den Split enstandenen Boxen möglichst klein ist.

Illustrieren Sie die einzelnen Phasen im Aufbau des R-Baums. Zeichnen Sie hierzu den Baum und den Datenraum unmittelbar vor jedem Split und im Endzustand.

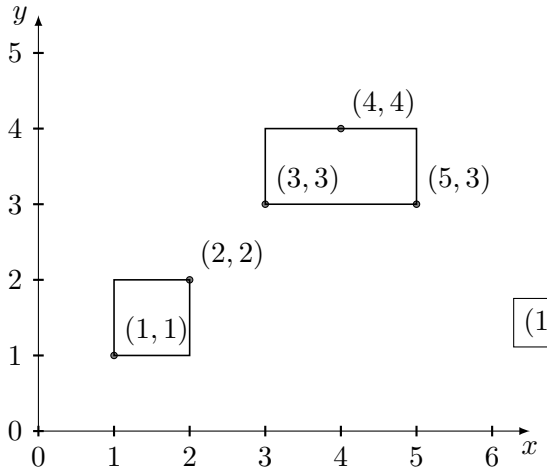
Lösung:

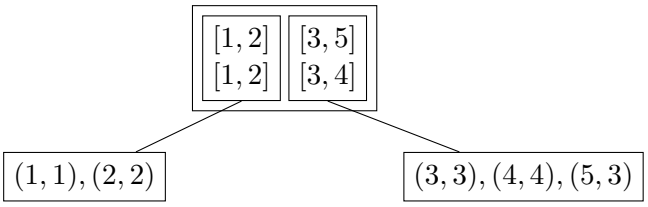
Wir fügen die Werte $(1,1), (2,2), (3,3)$ und $(4,4)$ in den ersten Knoten ein.



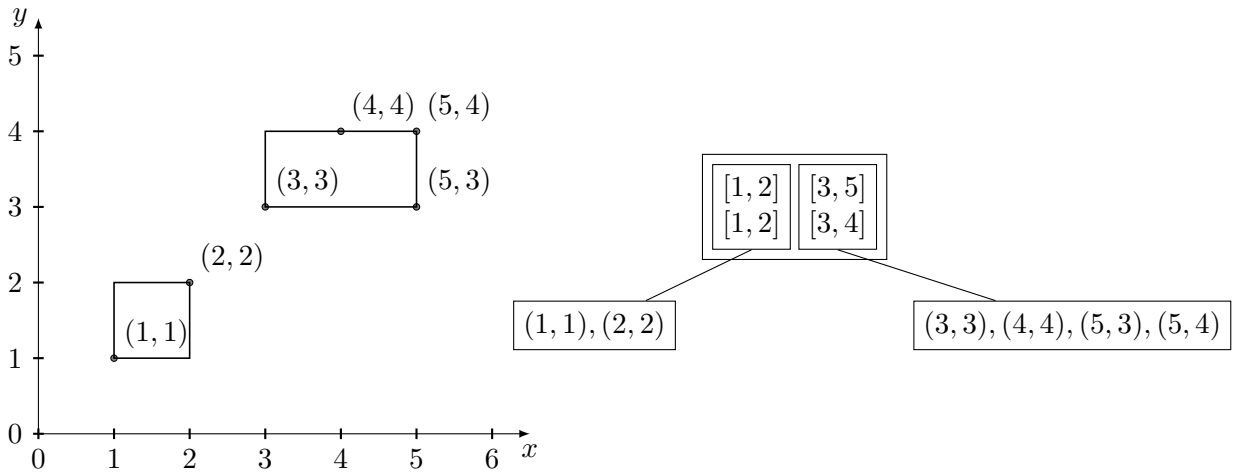


Da der Knoten nun voll ist, müssen wir ihn für den Wert $(5,3)$ aufspalten. Um die Gesamtfläche gering zu halten, fügen wir den neuen Wert dann in das zweite Blatt ein:

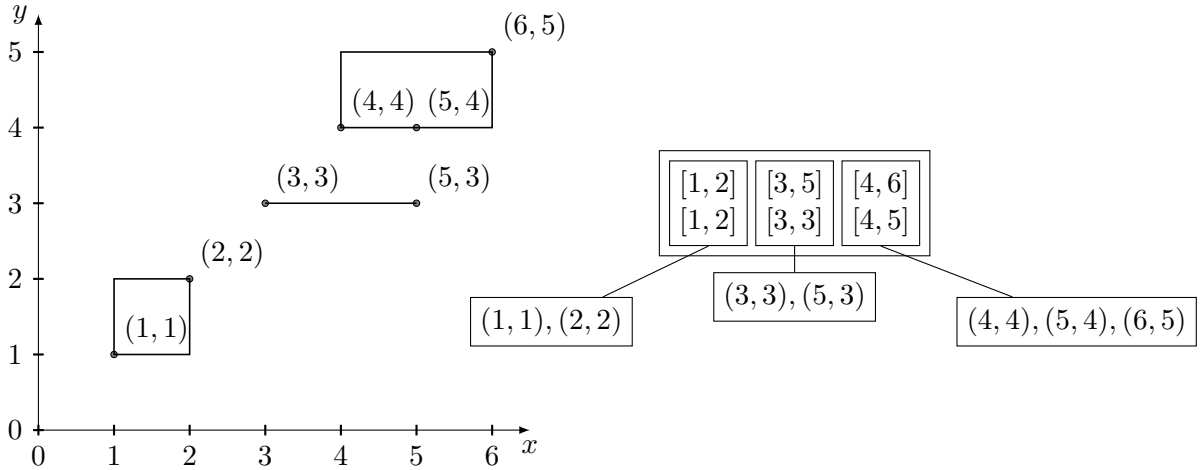




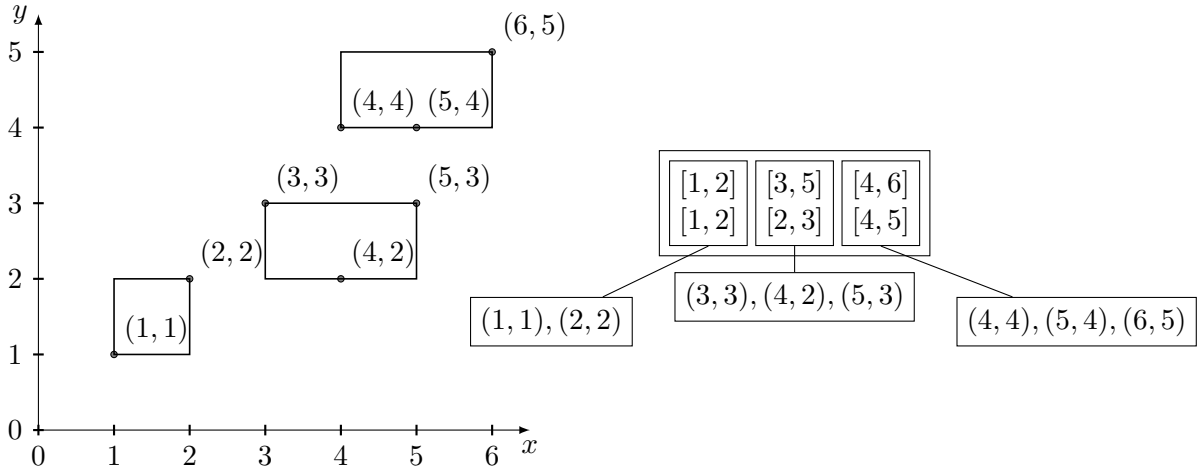
In dieses Blatt passt auch noch der Wert $(5,4)$:



Vor Einfügen des Wertes $(6, 5)$ müssen wir nun den zweiten Knoten splitten. Um die Größe nach dem Split minimal zu halten, ordnen wir die Werte $(3, 3)$ und $(5, 3)$, sowie $(4, 4)$ und $(5, 4)$ jeweils einer Box mit der Fläche 0 zu und fügen schließlich $(6, 5)$ in die obere Box ein.



Für den Wert $(4, 2)$ gibt es zwei Möglichkeiten: Wir können den Wert entweder in die zweite Box mit der Größe 0, oder in die erste Box mit der Größe 1 einfügen. In beiden Fällen vergrößert sich die Gesamtfläche um 2. Wir entscheiden uns hier für die erste Option, die Alternative ist aber genauso richtig.



NB: Bei R-Bäumen ist es nicht immer eindeutig, wo neue Werte einfügt werden müssen und wie man Knoten aufspaltet!

Bonusaufgabe 4

Klausuraufgabe aus dem WiSe 2016/17:

Gegeben sei das bekannte Uni-Schema. Formulieren Sie in SQL-92: Finden Sie alle Vorlesungen **(VorlNr und Titel duplikatfrei ausgeben)**, die nicht vor dem dritten Semester gehört werden sollten. – Ein Beispiel hierfür ist die Vorlesung Bioethik (5216), da diese die Vorlesung Ethik (5041) voraussetzt, welche wiederum die Vorlesung Grundzüge (5001) als Voraussetzung hat.

Lösung:

Ohne Rekursion:

```
SELECT DISTINCT v.VorlNr, v.Titel
FROM Vorlesungen v, voraussetzen v1, voraussetzen v2
WHERE v.VorlNr = v1.Nachfolger
  AND v1.Vorgaenger = v2.Nachfolger
```

Mit Rekursion:

```
WITH recursive vor as (
  SELECT *, 1 as cnt FROM voraussetzen
  UNION
  SELECT v1.vorgaenger, v2.nachfolger, v1.cnt + 1
  FROM vor v1, voraussetzen v2
  WHERE v1.nachfolger = v2.vorgaenger
  )
SELECT DISTINCT nachfolger, v.Titel
FROM vor, vorlesungen v
WHERE vor.nachfolger = v.VorlNr AND vor.cnt >= 2
```

Bonusaufgabe 5

Implementieren Sie in der Programmiersprache Ihrer Wahl einen B+ Baum *oder* eine Extendible Hashtable. Es sollten mindestens die Funktionen *insert* und *lookup* unterstützt werden. Zur Vereinfachung können Sie annehmen, dass lediglich Schlüssel-Wert-Paare bestehend aus Integern eingefügt werden.

Diese Aufgabe ist für diejenigen gedacht, die sich über den Vorlesungsstoff hinaus mit dem Thema Datenbanken beschäftigen wollen. Sie wird nicht in der Übung besprochen und ist nicht klausurrelevant. Falls Sie Feedback wünschen, können Sie Ihre Lösung gerne an `gdb@in.tum.de` *senden.*

Der Lehrstuhl für Datenbanksysteme wünscht frohe Weihnachten und schöne Feiertage!