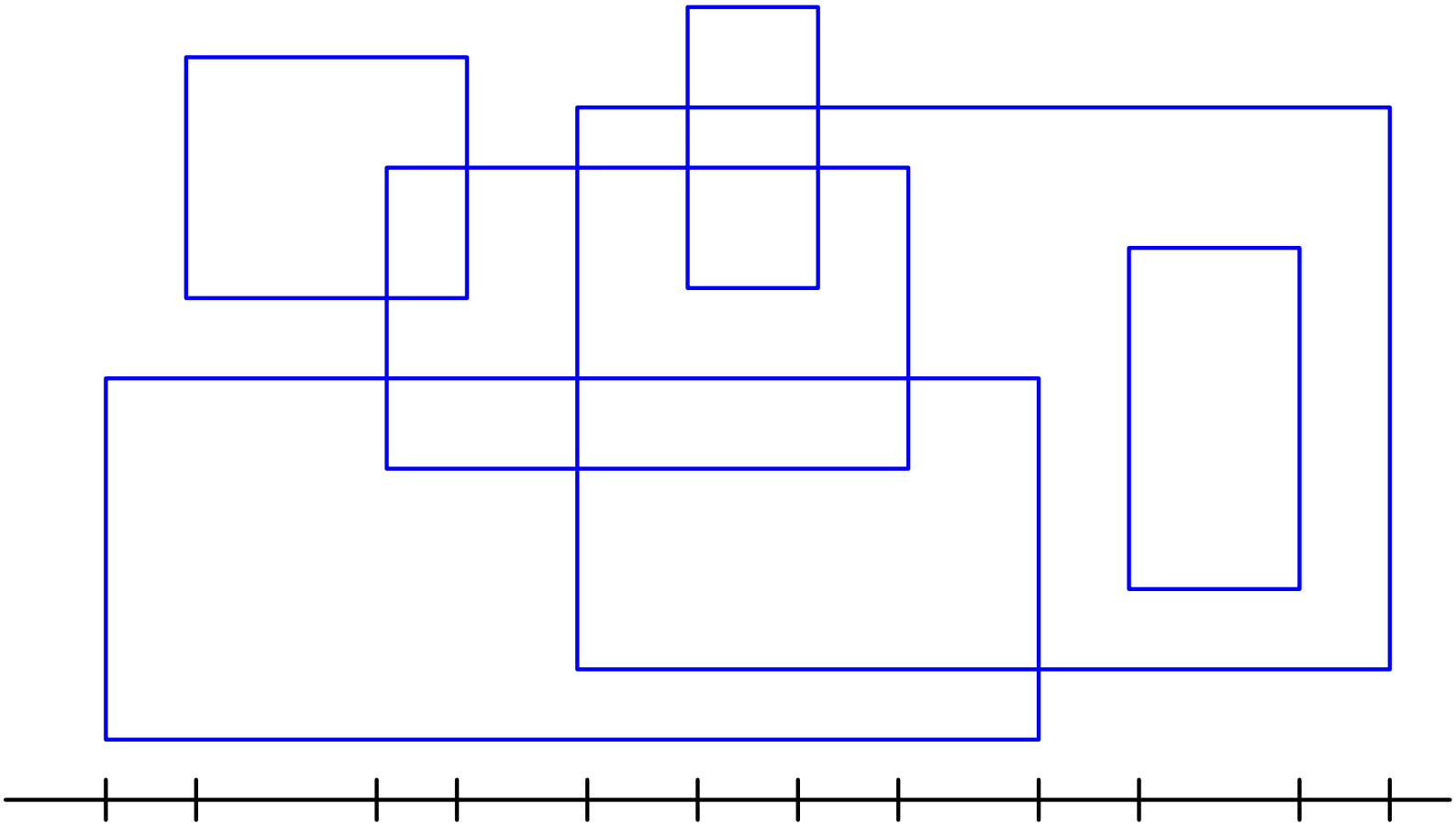
Geometrische Datenstrukturen

1. Rechteckschnitt
2. Segment Bäume
3. Intervall Bäume
4. Prioritätssuchbäume

1. Rechteckschnitt



- Schwenke horizontale Scan-Line von oben nach unten.

- Speichere die Schnitte der jeweils aktiven Rechtecke mit der Scan-Line in Statusstruktur L.

Operationen auf L:

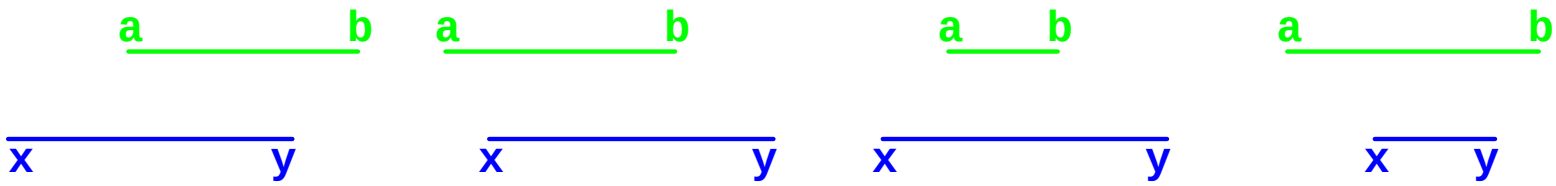
- **Einfügen eines Intervalls in L**

- **Entfernen eines Intervalls aus L**

- **Für ein gegebenes Intervall I:**
  Bestimmung aller Intervalle aus L, die sich mit I überlappen

L speichert Menge von Intervallen über einer diskreten, bekannten Menge von möglichen Endpunkten.

Reduktion der Überlappungsanfrage:

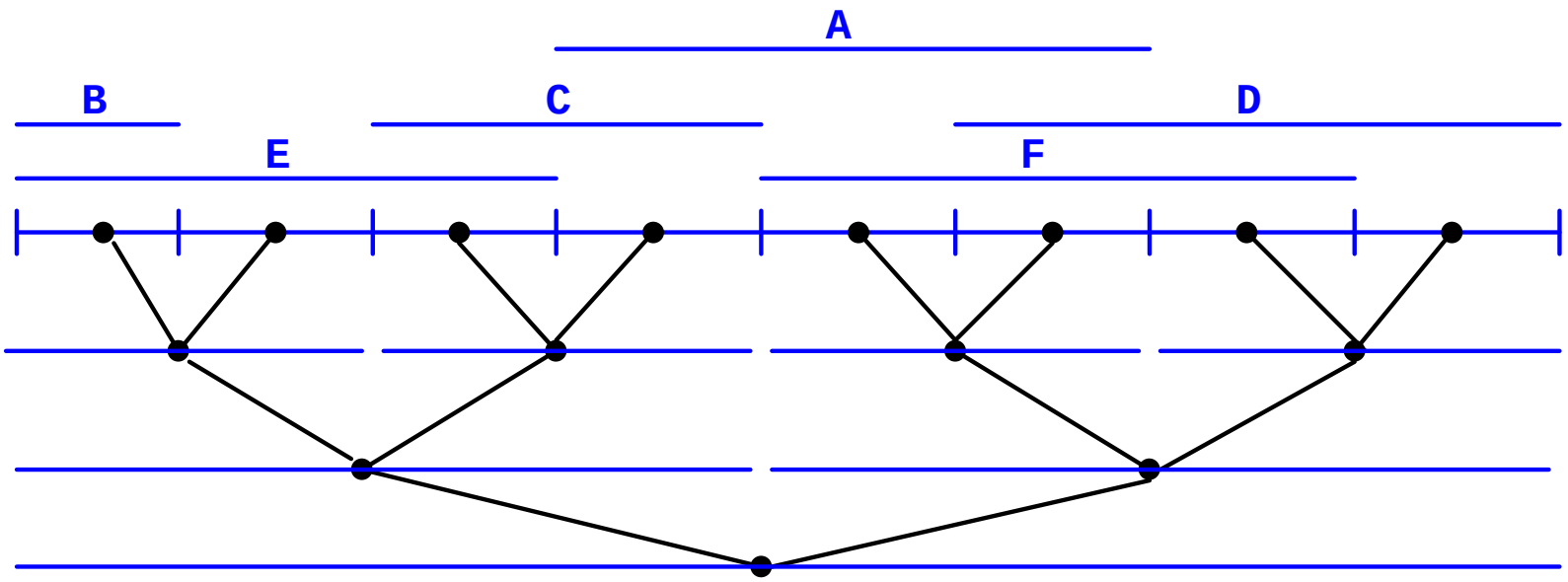


2. Segmentbäume

Segmentbäume sind eine Struktur zur Speicherung von Intervallen, die folgende Operationen unterstützen:

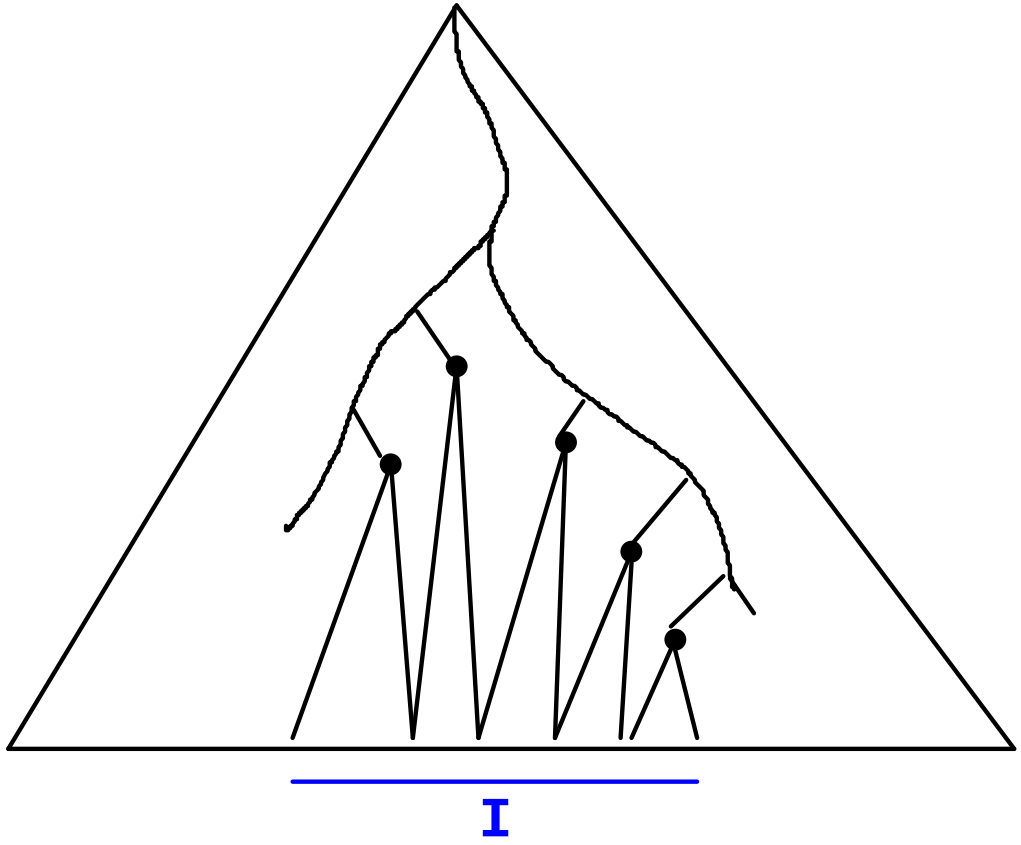
- **Einfügen** von Intervallen
- **Entfernen** von Intervallen
- **Aufspieß-Anfragen (stabbing queries):**
  Für einen gegebenen Punkt a, berichte alle Intervalle, die a enthalten (die a aufspießt).

Für die Lösung des Rechteckschnittproblems genügen semidynamische Segmentbäume.



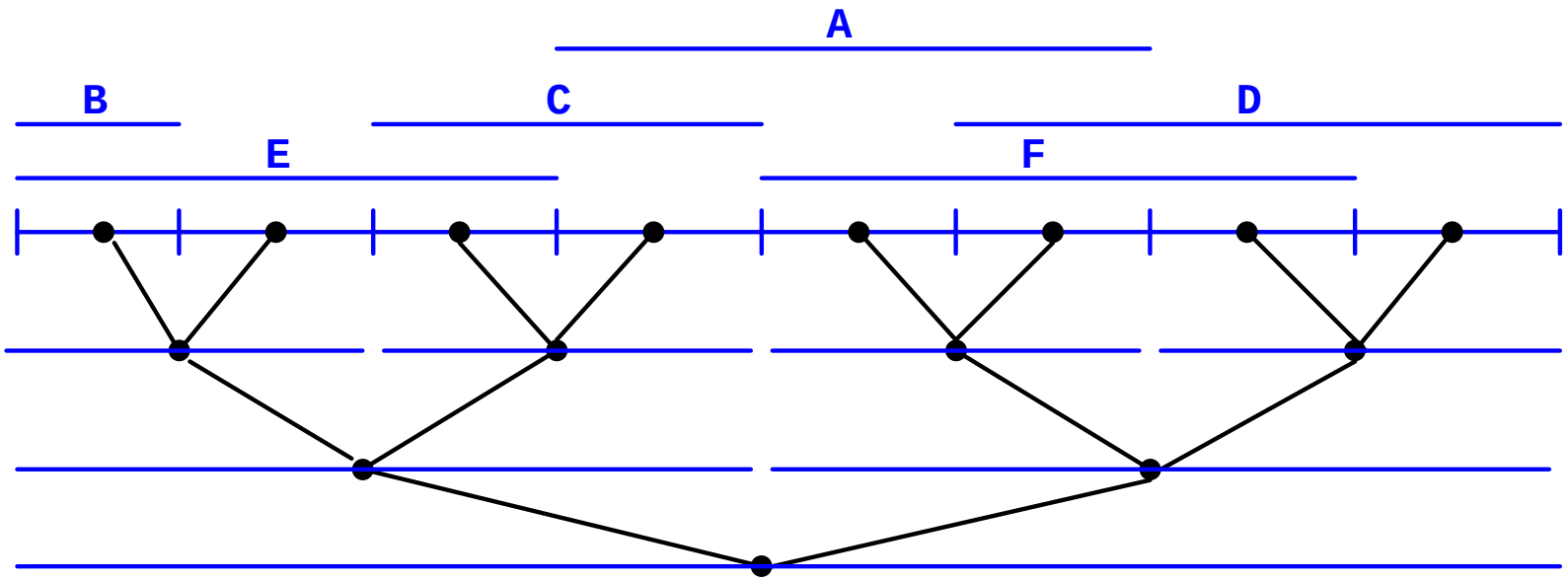
Ein Intervall I tritt in der Liste eines Knotens p auf gdw. p der von der Wurzel aus erste Knoten ist, sodaß das Intervall I(p) in I enthalten ist.

Einfügen eines Intervalls ist in O(log N) Schritten möglich.



Jedes Intervall I kommt in höchstens O(log N) Intervall Listen vor.

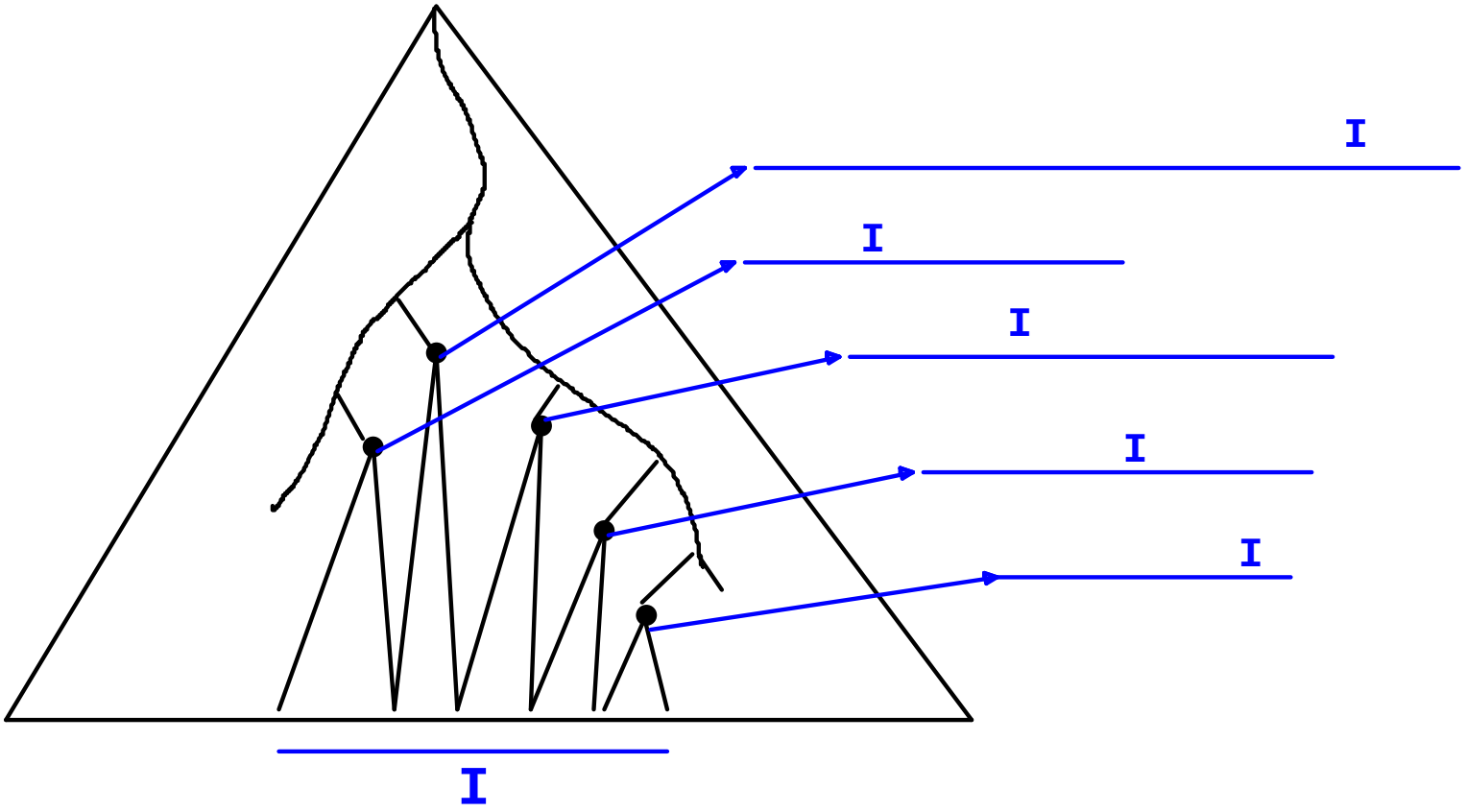
Aufbau eines Segmentbaumes mit N Intervallen ist in Zeit O(N log N) möglich.

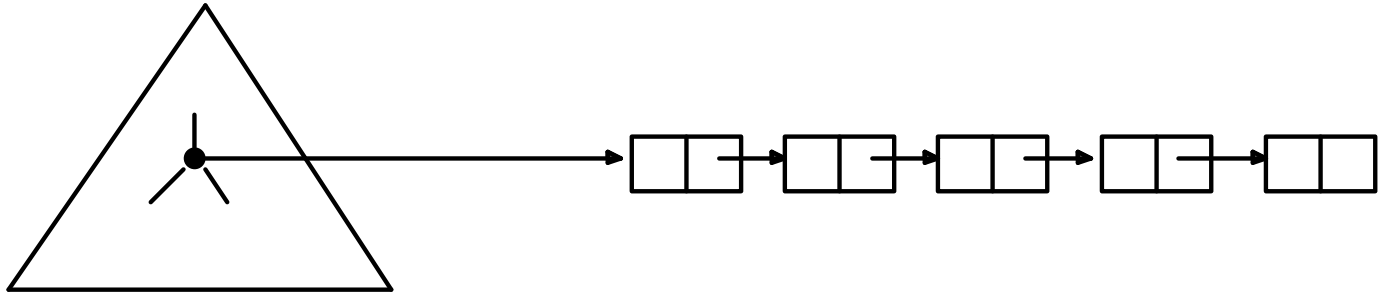



```
procedure report(p: Knoten; x: Punkt):
  gebe alle Intervalle der Liste von p aus;
  if p ist Blatt then fertig else
    { if (p hat linken Sohn pl & x in I(pl))
           then report(pl, x);
      if (p hat rechten Sohn pr & x in I(pr))
           then report(pr, x);
    }
```

Alle k Intervalle, die ein gegebener Punkt aufspießt, können in Zeit O(log N + k) berichtet werden.

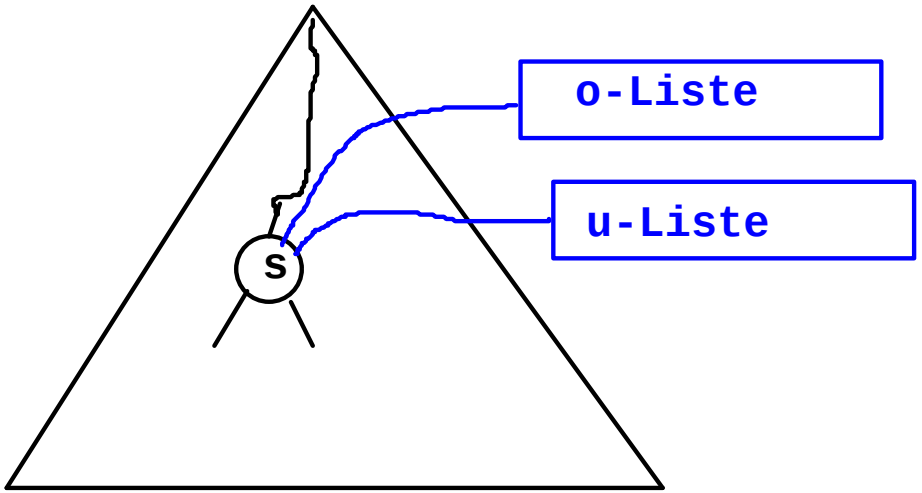
Entfernen von Intervallen





Wörterbuch für
alle Intervalle

3. Intervallbäume



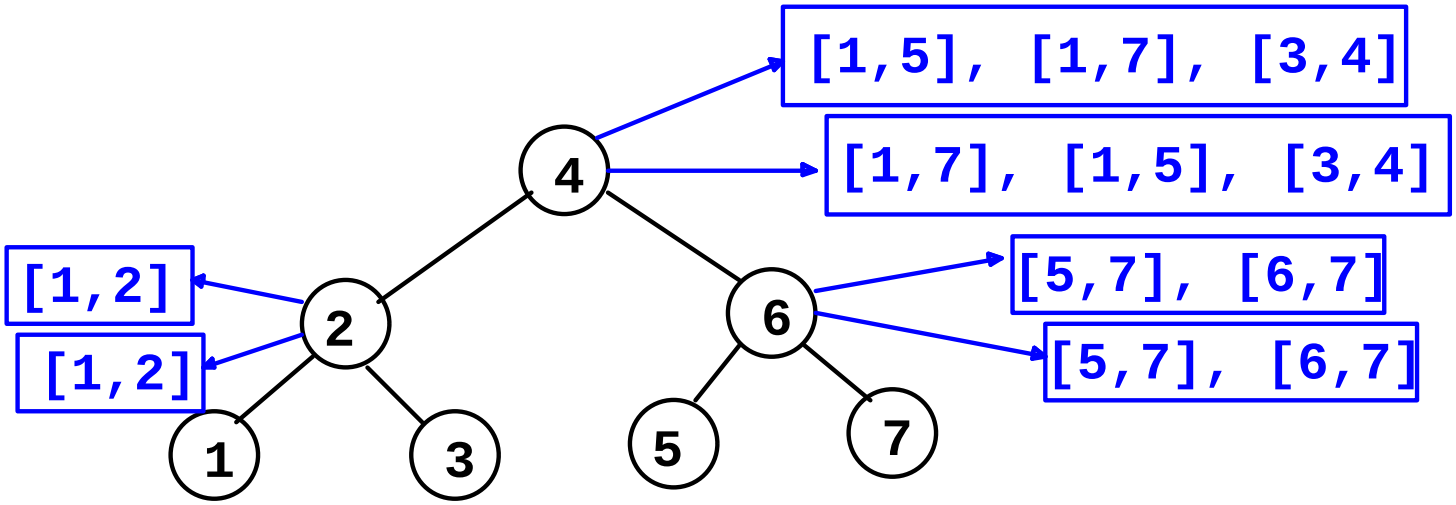
Skelett (vollständiger Suchbaum der Intervallgrenzen)

`o-Liste` **Nach absteigenden oberen Endpunkten sortiert**

`u-Liste` **Nach aufsteigenden unteren Endpunkten sortiert**

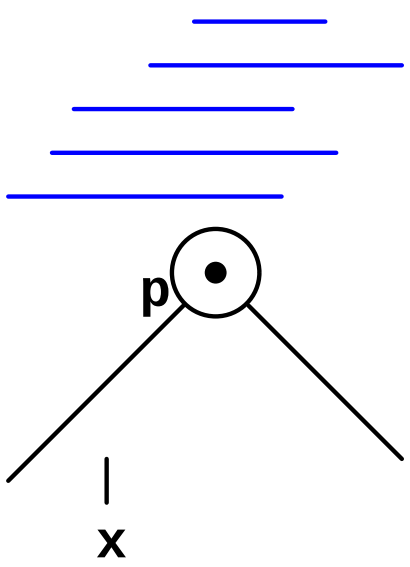
Intervall [l,r] kommt in u-/o-Liste des Knotens s vor gdw. s der Knoten minimaler Tiefe ist, sodass s in [l,r] liegt.

{[1,2], [1,5], [3,4], [5,7], [6,7], [1,7]}



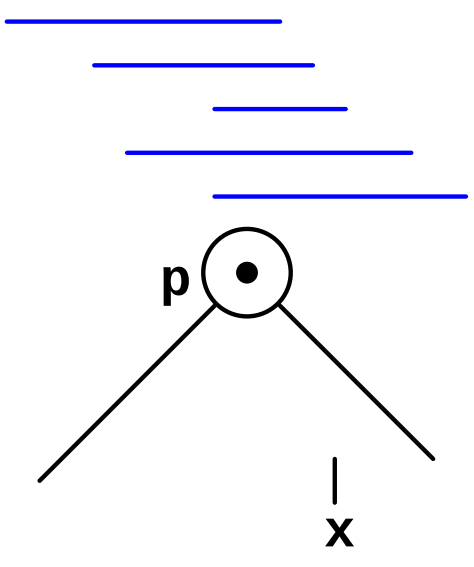
Einfügen und Entfernen von Intervallen in einem Intervallbaum mit Skelett der Größe O(N) und insgesamt O(N) Intervallen kann in Zeit O(log N) ausgeführt werden.

u-Liste



x<p.key

o-Liste



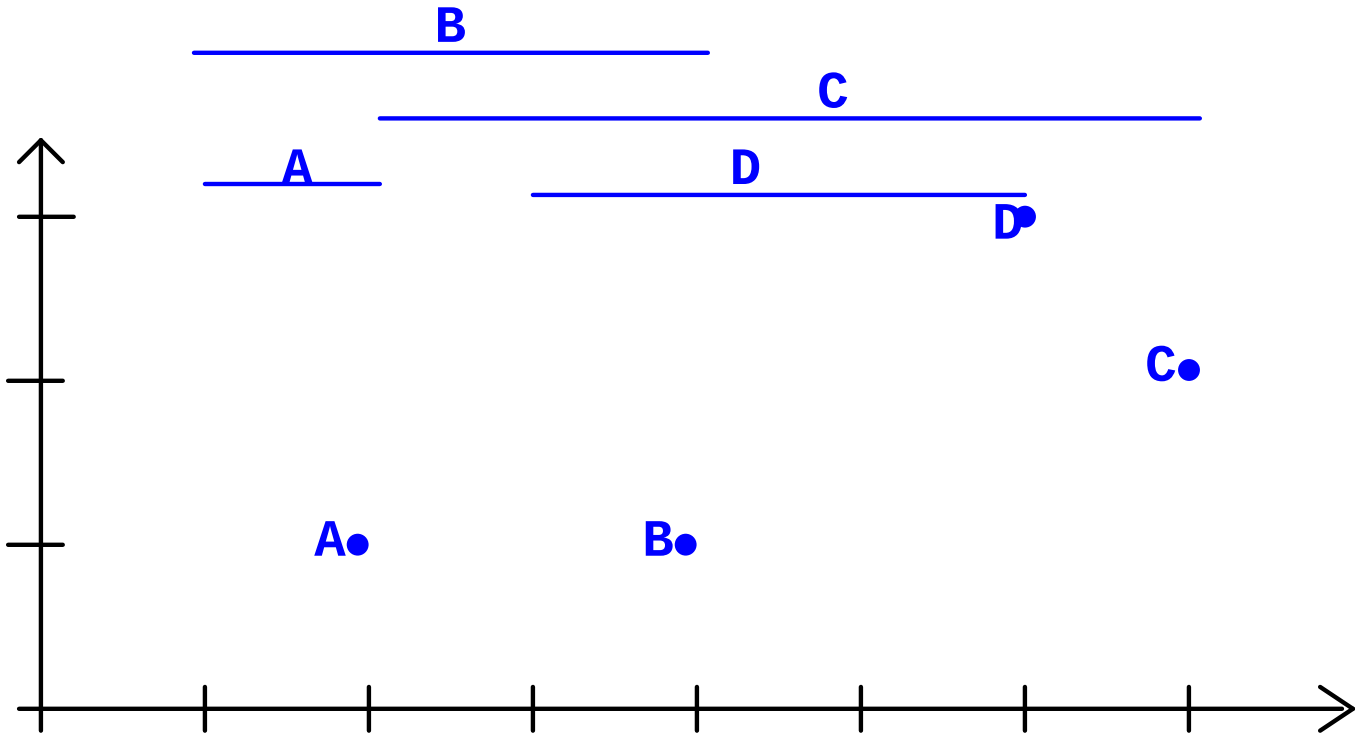
x>p.key

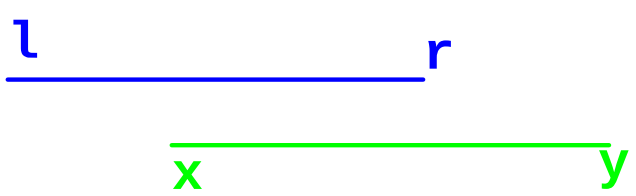
```
procedure report(p:Knoten; x:Punkt);
if x=p.key then report alle Intervalle der u/o-Liste
else if x<p.key then { report Anfangsstück der
                       u-Liste;
                       report(pl,x}
else (x>p.key)       { report Anfangsstück der
                       o-Liste;
                       report(pr,x}
```

Aufspieß-Anfragen können in Zeit **O(log N + k)** beantwortet werden.

4. Prioritätssuchbäume

Idee: Repräsentiere Intervalle durch Punkte in der Ebene!





$[x,y] \cap [l,r] \neq \emptyset$ gdw.

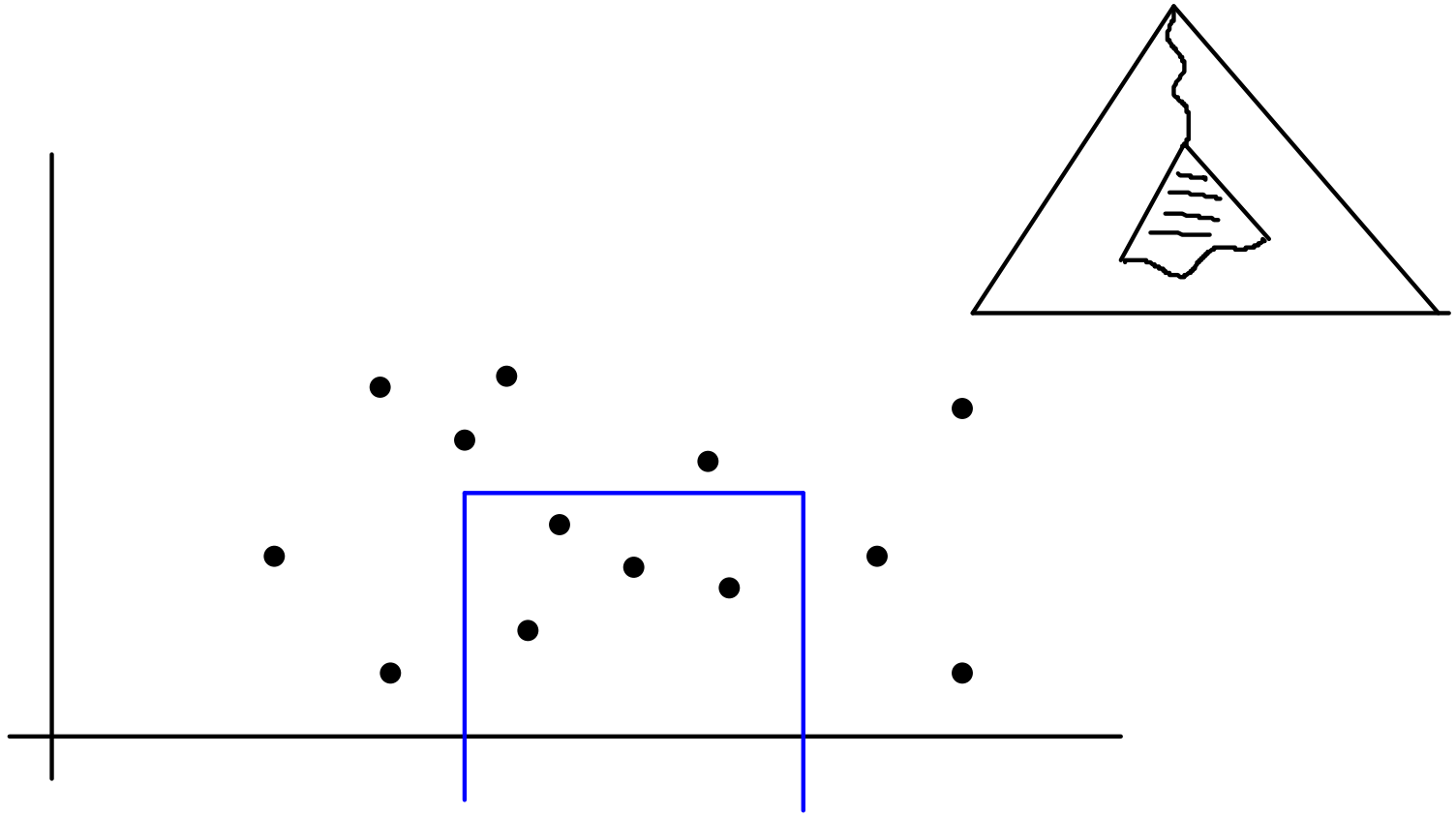
(l<=y und x<=r)

Prioritätssuchbäume sind eine1.5-dim Struktur zur Speicherung von Punkten, die die folgenden Operationen unterstützt:

Einfügen eines Punktes

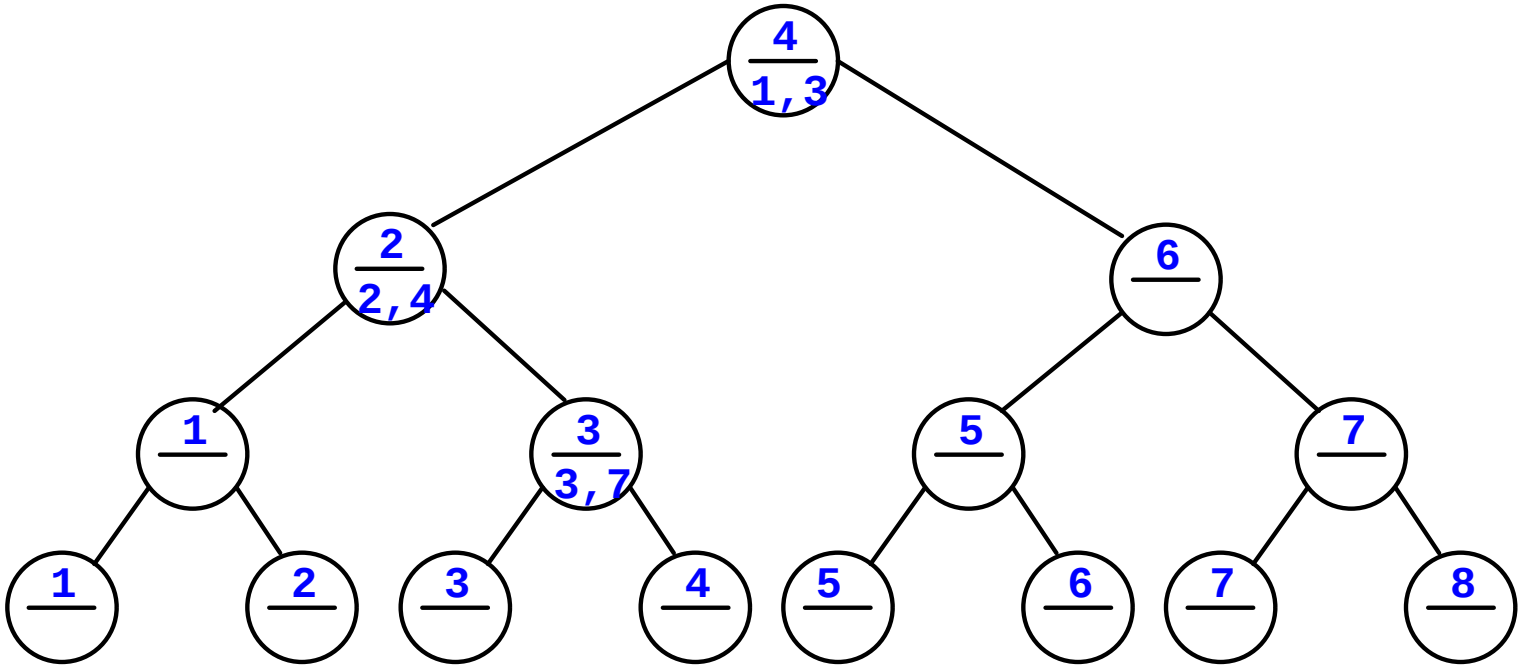
Entfernen eines Punktes

Südlich gegründete Bereichsanfragen



Prioritätssuchbäume sind
- binäre Blattsuchbäume für die x-Koord.
- Min-Heaps für die y-Koord. von Punkten.

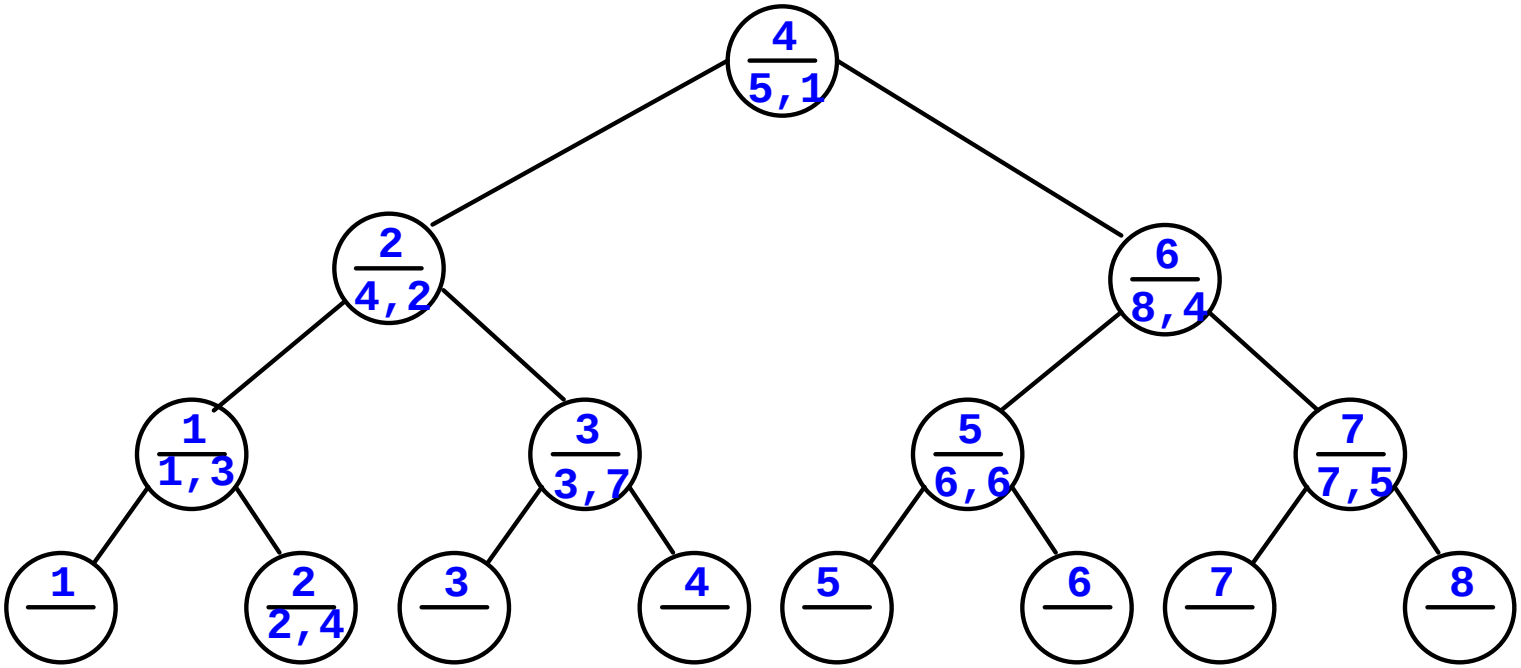
M = {(1,3),(2,4),(3,7),(4,2),(5,1),(6,6),(7,5),(8,4)}



Einfügen eines Punktes p=(x,y):
Lege p auf dem Suchpfad nach x entsprechend seinem y-Wert ab! D.h. trifft p unterwegs auf einen Punkt q mit größerem y-Wert, lege dort p ab und setze Verfahren mit q fort.

Einfügen eines Punktes ist in Zeit O(log N) möglich.

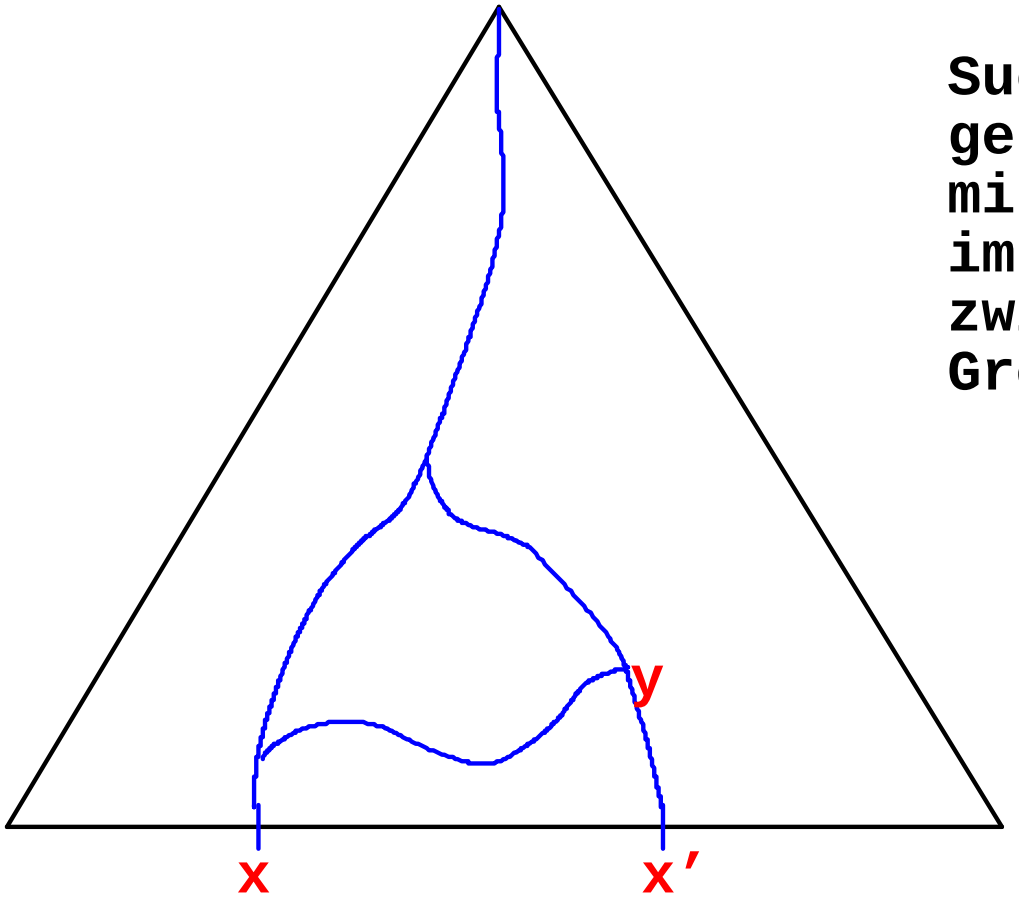
Entfernen



**Suche Punkt p im Baum und entferne ihn;
Schließe Lücke (rekursiv) durch Hochziehen
des Punktes mit kleinerem y-Wert.**

**Entfernen eines Punktes ist in Zeit O(log N)
möglich.**

Südlich gegründete Bereichsanfragen (x,x’,y):

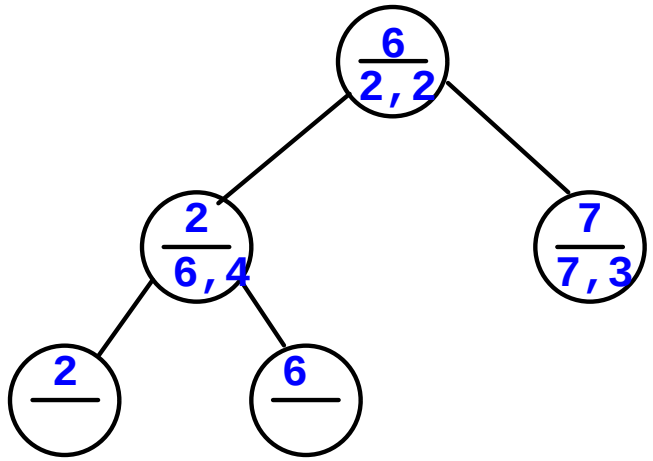


Suche nach x und x’, gebe alle Punkte mit y-Wert <y aus im Bereich zwischen diesen Grenzen.

Ausführbar in Zeit $O(\log N + k)$.

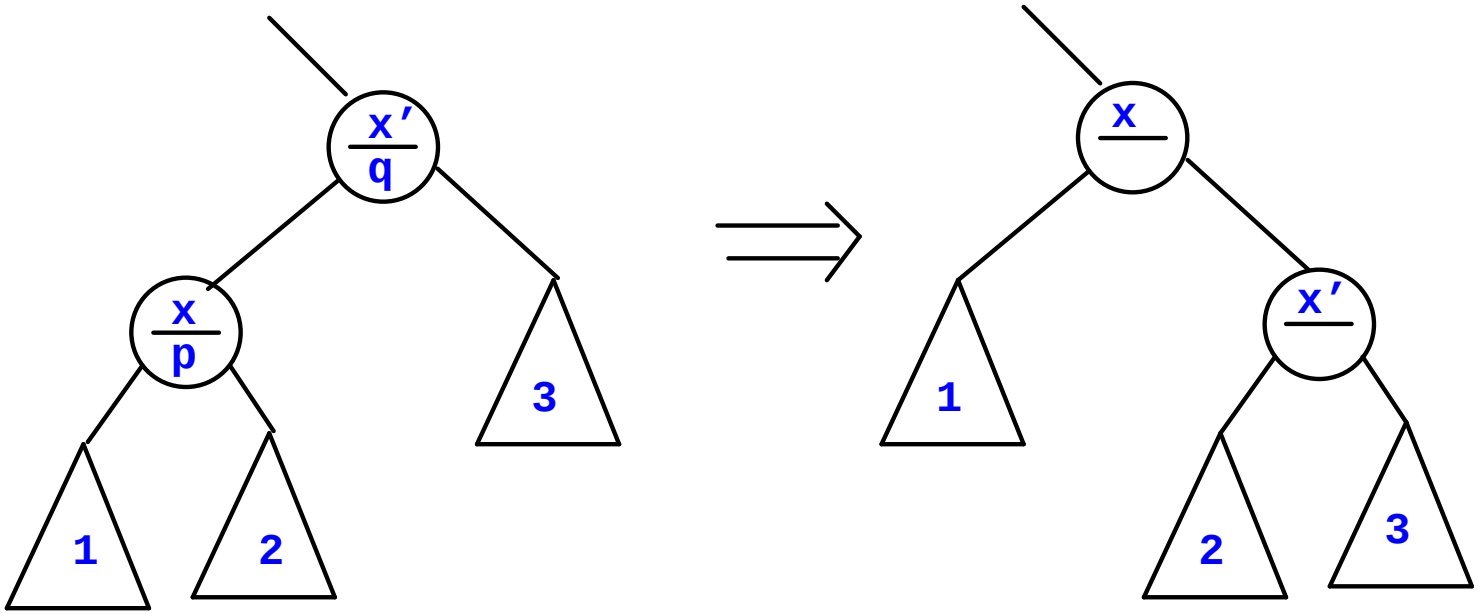
Möglichkeiten zur vollen Dynamisierung:

Kein starres Skelett, sondern mit der Punktmenge wachsenden oder schrumpfenden Suchbaum unterlegen.



Einfügen von (5,3)

Unterlegen balancierter Bäume:



Rotation konserviert die x-Ordnung und zerstört i.a. die y-Ordnung.