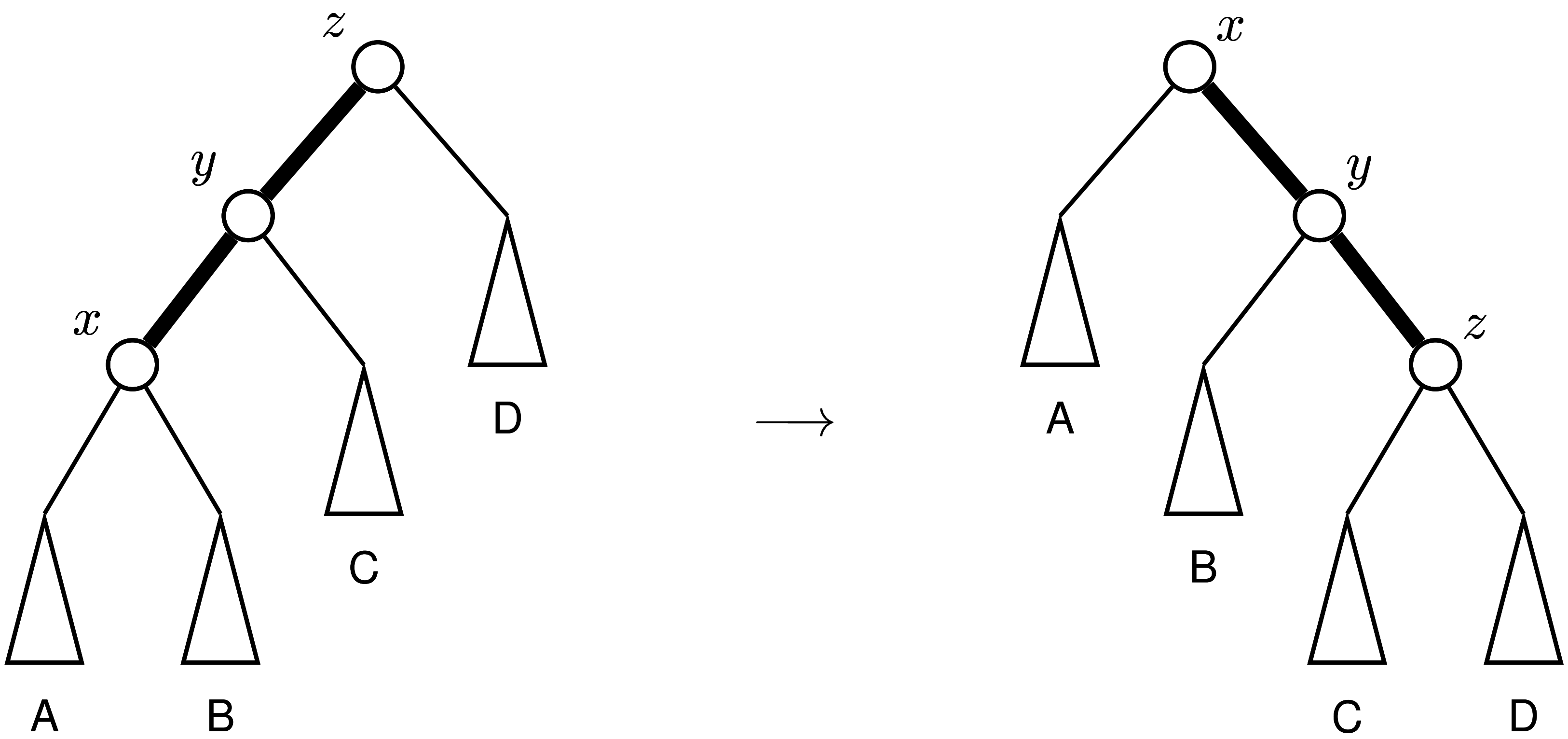
1.Fall: $y = parent(x)$ ist nicht Wurzel und

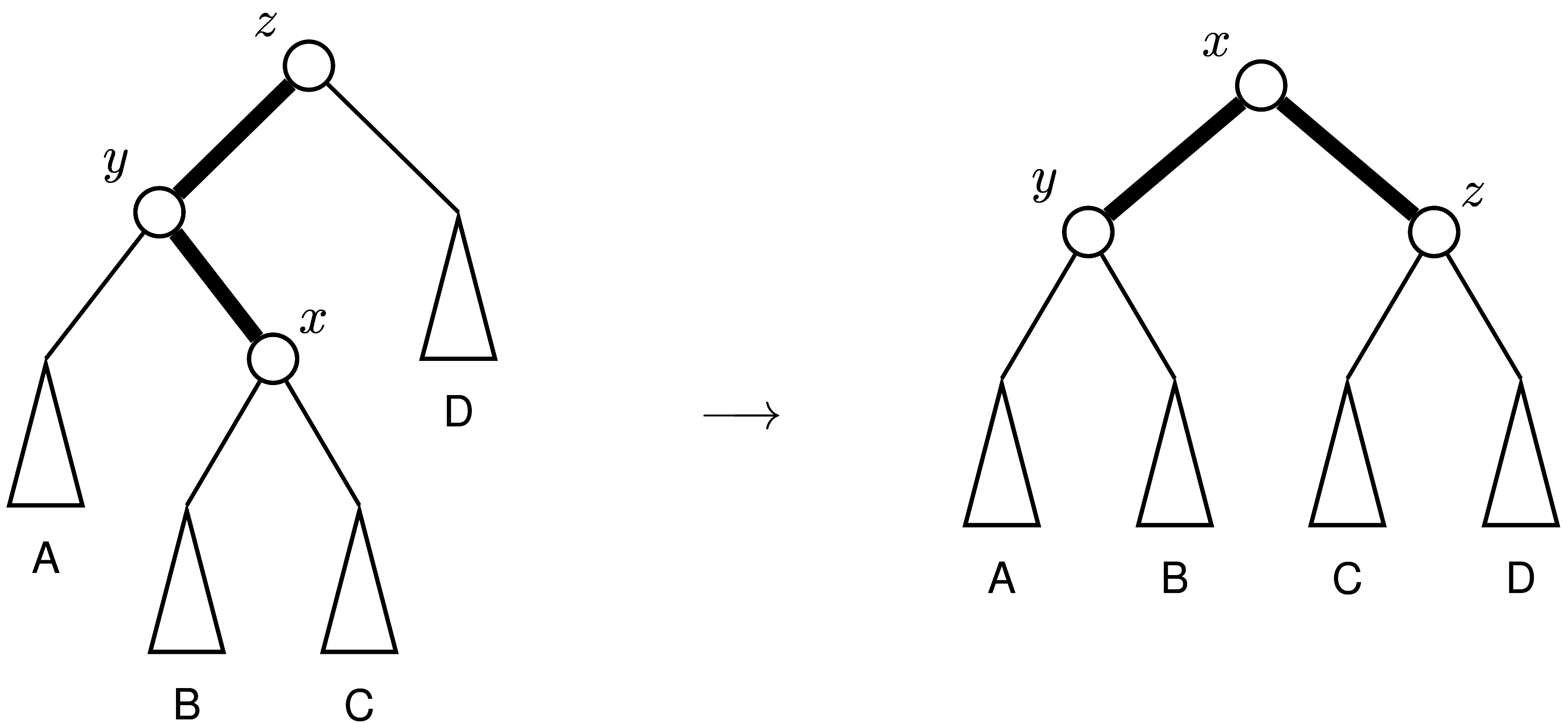
a) $key(x) < key(y) < key(z)$



b) $key(x) > key(y) > key(z)$: [analog]

2.Fall: $y = parent(x)$ ist nicht Wurzel und

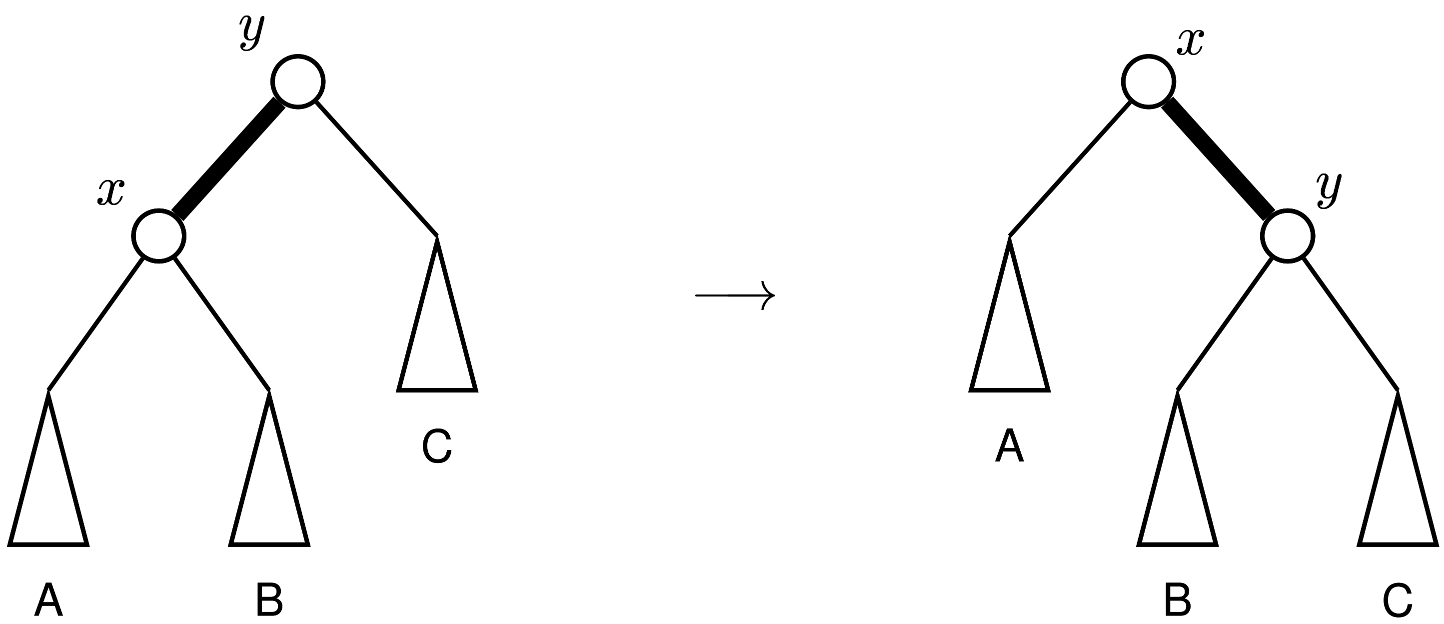
a) $key(x) > key(y) < key(z)$



b) $key(x) < key(y) > key(z)$: [analog]

3.Fall: $y = parent(x)$ ist Wurzel

$\rightsquigarrow$ Einfache Rechts- bzw. Linksrotation



Find(k,T) : Den Knoten mit dem Schlüssel k finden.

Implementierung:

Suchpfad von der Wurzel zu Knoten v bestimmen, wobei entweder $\mathrm{key}(v) = k$ oder, falls so ein Knoten nicht existiert, $\mathrm{key}(v)$ der größte Schlüssel kleiner k oder der kleinste Schlüssel größer k in T ist;

Splay(v,T);

if $(\mathrm{key}(v) = k)$

    return v;

else

    return $NULL$;

Split(k,T): T in zwei Bäume T_1 und T_2 aufspalten, wobei T_1 alle $x \in T$ mit $\mathrm{key}(x) \leq k$ enthält und T_2 alle übrigen Knoten.

Implementierung:

Find(k,T);

if $(\mathrm{key}\,(\mathrm{root}\,(T)) \leq k)$

    breche rechte Kante der Wurzel auf;

else

    breche linke Kante der Wurzel auf;

Insert(x,T): x in den Baum T einfügen.

Implementierung:

/*x zur Wurzel und T zerlegen:*/

(T_1,T_2) = Split(key(x),T);

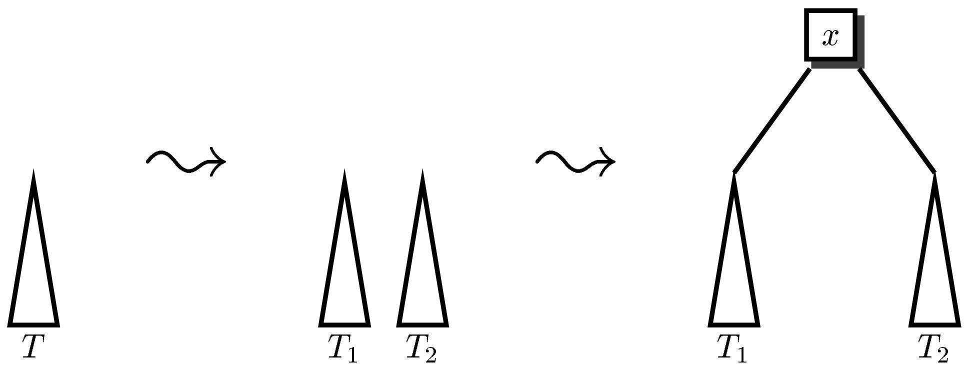
if $(\text{key}\ (\text{root}\ (T_1)) = \text{key}\ (x))$

STOP; /*Schlüssel existiert bereits*/

else

mache x zur Wurzel von T mit den beiden Unterbäumen T_1 und T_2;

Skizze:



Join(T_1,T_2): Verschmelzen von T_1 und T_2 zu einem Baum T, wobei
$\mathrm{key}(x) \leq \mathrm{key}(y) \quad \forall x \in T_1, y \in T_2$.

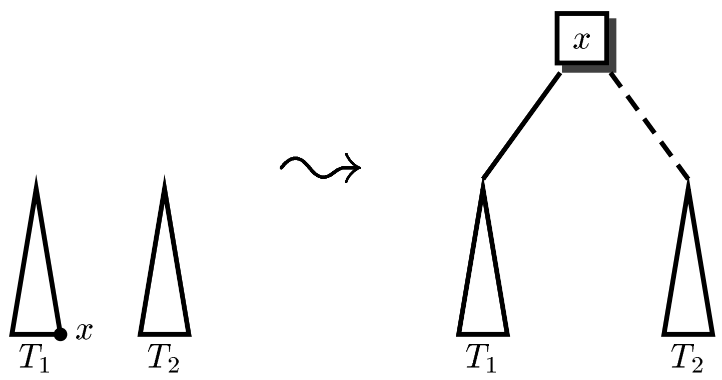
<u>Implementierung:</u>

bestimme größten Knoten x in T_1: laufe von Wurzel jeweils rechte Kante hinunter;

Splay(x,T_1);

mache Wurzel von T_2 zum Nachfolger von x;

Skizze:



Delete(k,T) Knoten mit Schlüssel k löschen, falls vorhanden.

Implementierung:

Find(k,T);

if $(\text{key}\ (\text{root}\ (T)) \neq k)$

STOP; /*Schlüssel nicht vorhanden*/

else $\{$

T_1 und T_2 Unterbäume der Wurzel von T;

T = Join(T_1,T_2);

$\}$