LOUDS

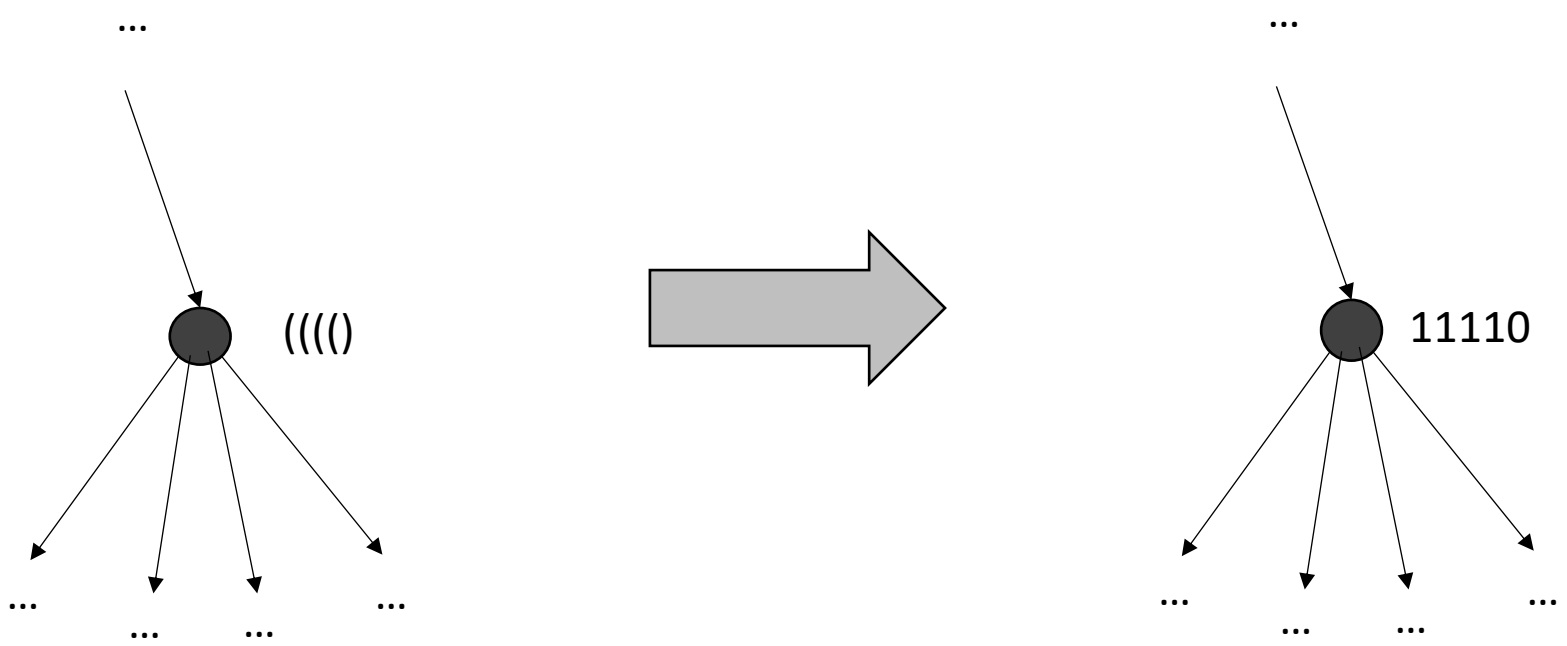
Level Ordered Unary Degree Sequence

Elena, Daniel, Christopher

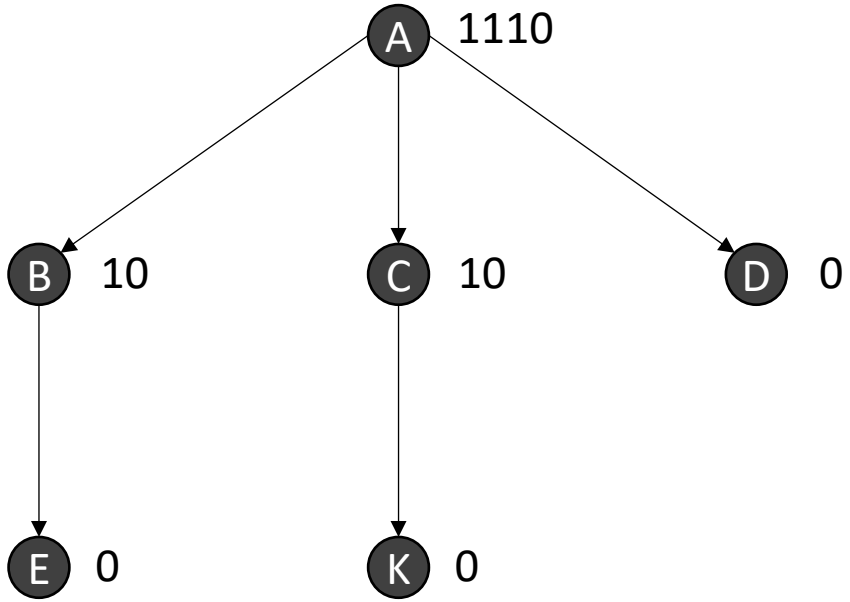
TU Dortmund SS 2018 Fachprojekt „Entwicklung einer Rust-Bibliothek am Beispiel von Succinct Trees“

Dozent: Johannes Köster

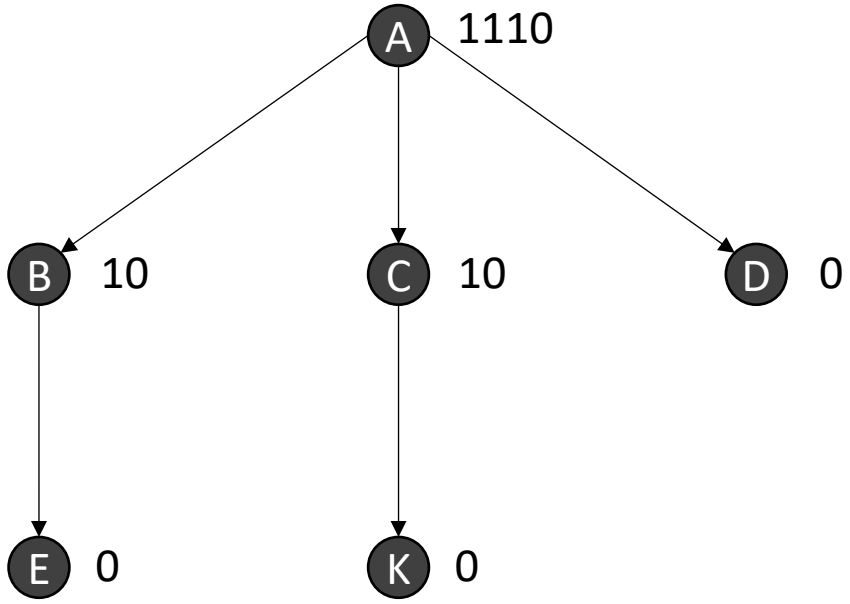
Baum als LOUDS



Baum als LOUDS



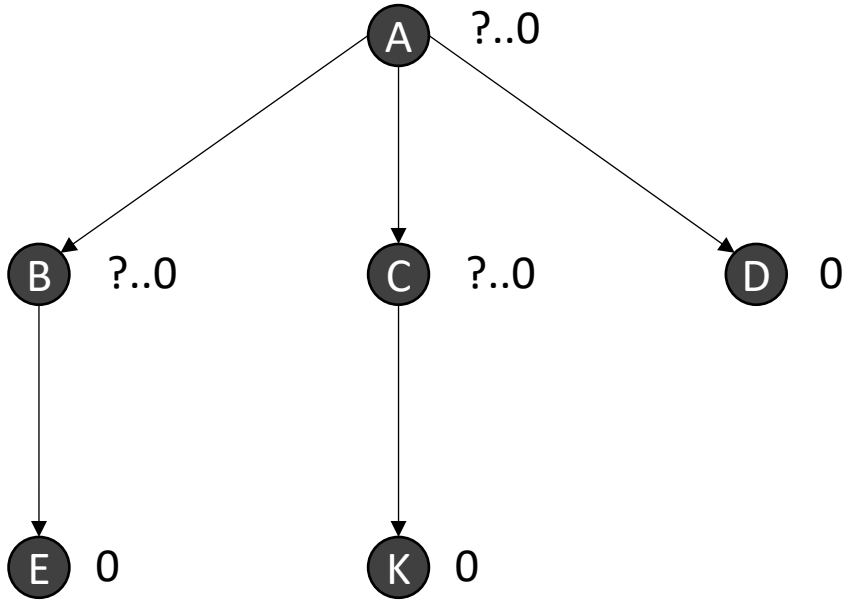
Baum als LOUDS



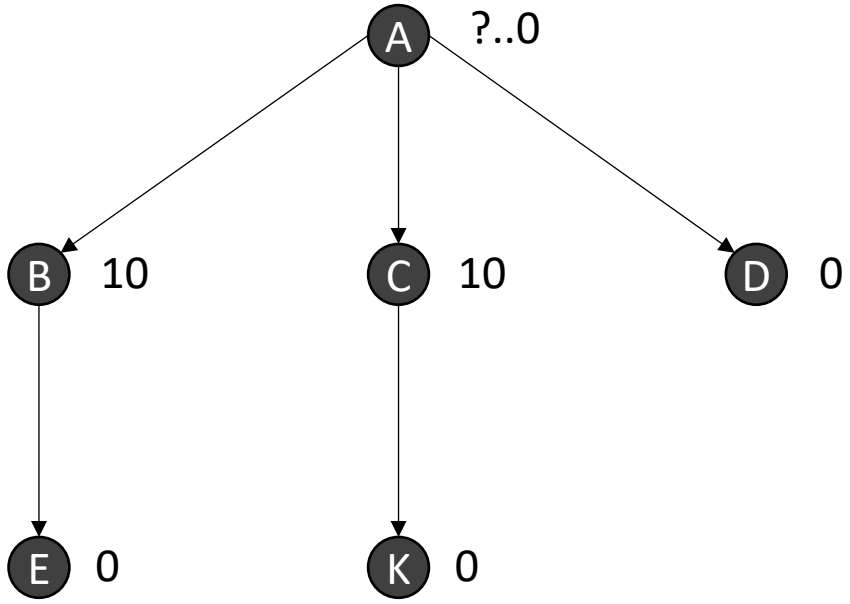
? Wie

? Erstellen

Baum als LOUDS

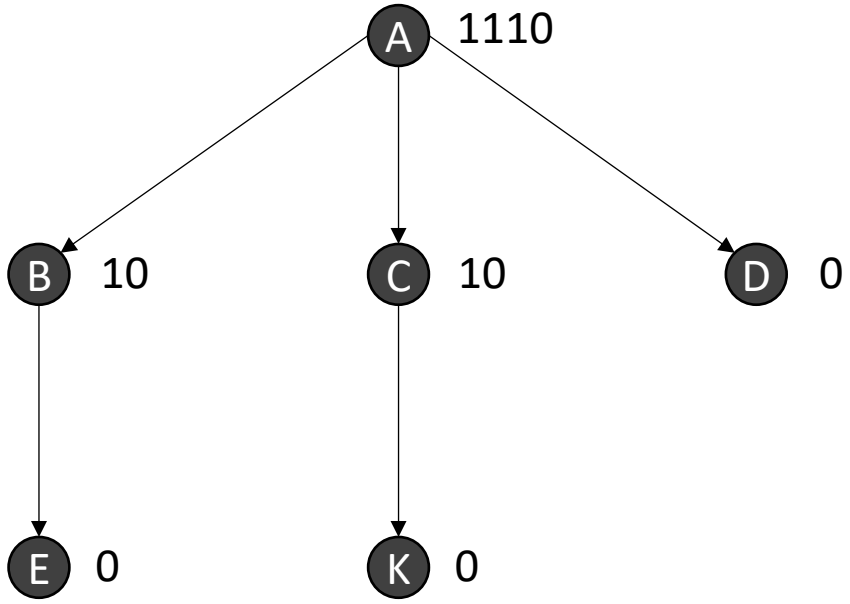


Baum als LOUDS



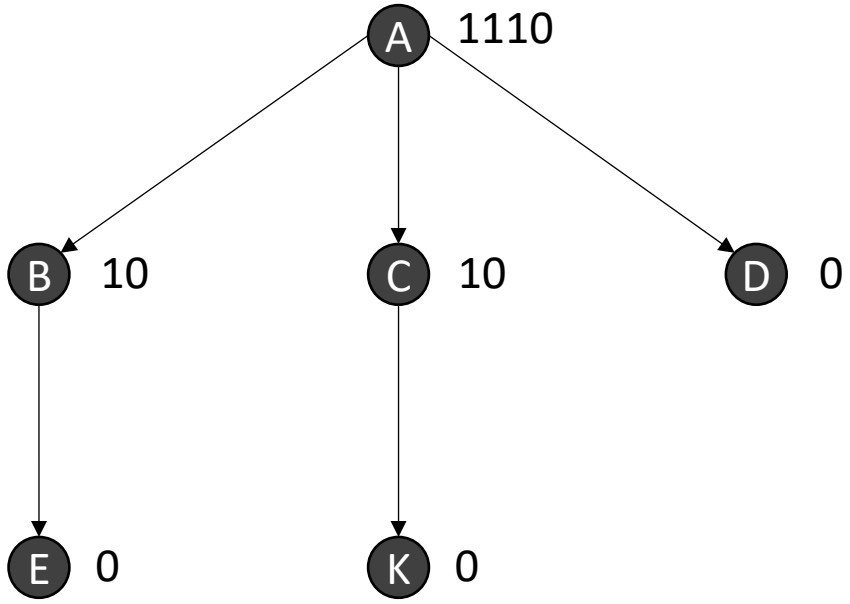
0	1	0	1	0	0	0	0

Baum als LOUDS



1	1	1	1	0	1	0	1	0	0	0	0

Baum als LOUDS



0	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	0	1	0	1	0	0	0	0

Befehle für LOUDS

In konstanter Zeit:

$isleaf(x)$ = (P[x] = 0)

$child\_rank(x) = y - prev_0(y);\ \ y = select_1(rank_0(x-1))$

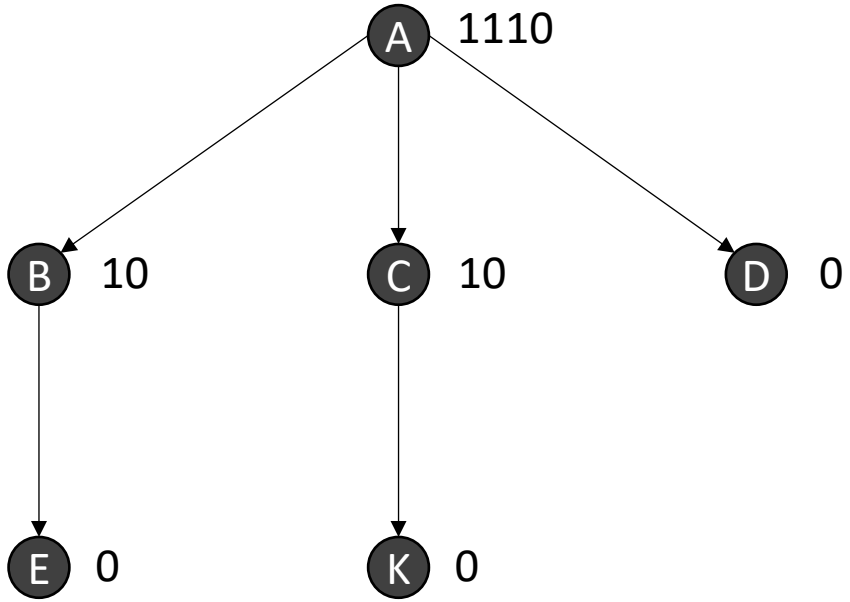
$next\_sibling(x) = select_0(select_1(y)+1);\ \ y = rank_0(x-1)+1$

$degree(x) = next_0(x) - x$

$parent(x) = prev_0(select_1(rank_0(x))) + 1$

$child(x,i) = select_0(rank_1(x)+i) + 1$

Prev und Next

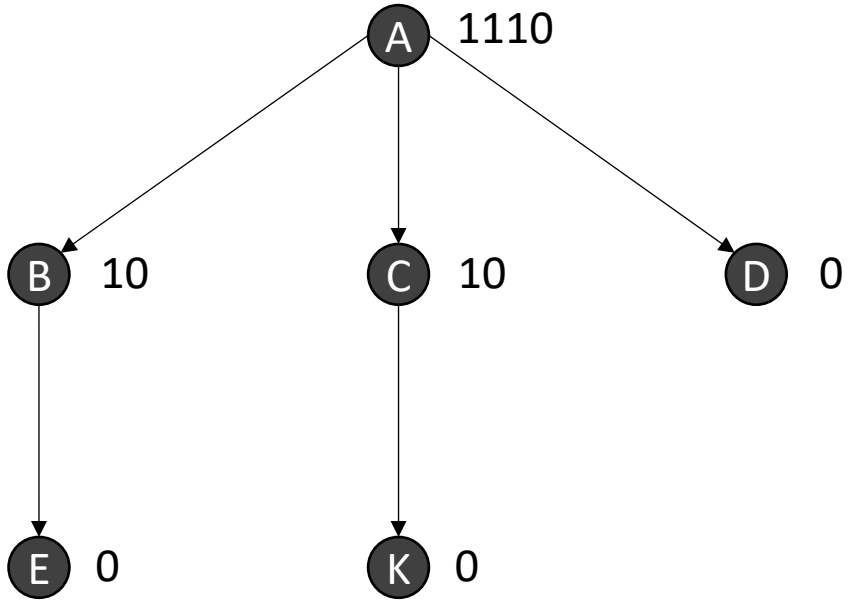


$$rank_0(x) = Anzahl\ der\ Nullen\ bis\ zur\ x - ten\ Stelle$$

$$select_0(x) = Die\ Position\ der\ x - ten\ Null$$

0	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	0	1	0	1	0	0	0	0

Prev und Next



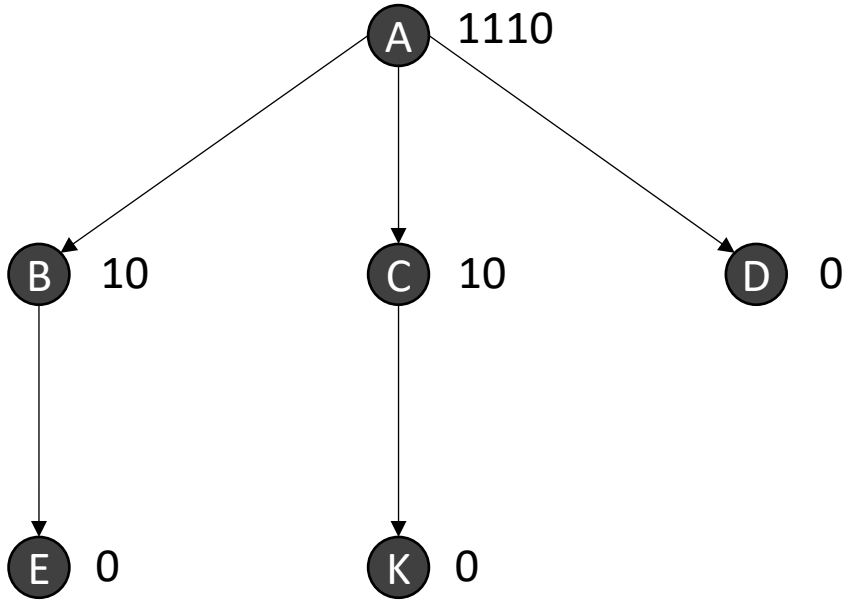
$$rank_0(x) = Anzahl\ der\ Nullen\ bis\ zur\ x - ten\ Stelle$$

$$select_0(x) = Die\ Position\ der\ x - ten\ Null$$

$$prev_0(x) = select_0(rank_0(x))$$

0	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	0	1	0	1	0	0	0	0

Prev und Next



$rank_0(x) = Anzahl\ der\ Nullen\ bis\ zur\ x - ten\ Stelle$

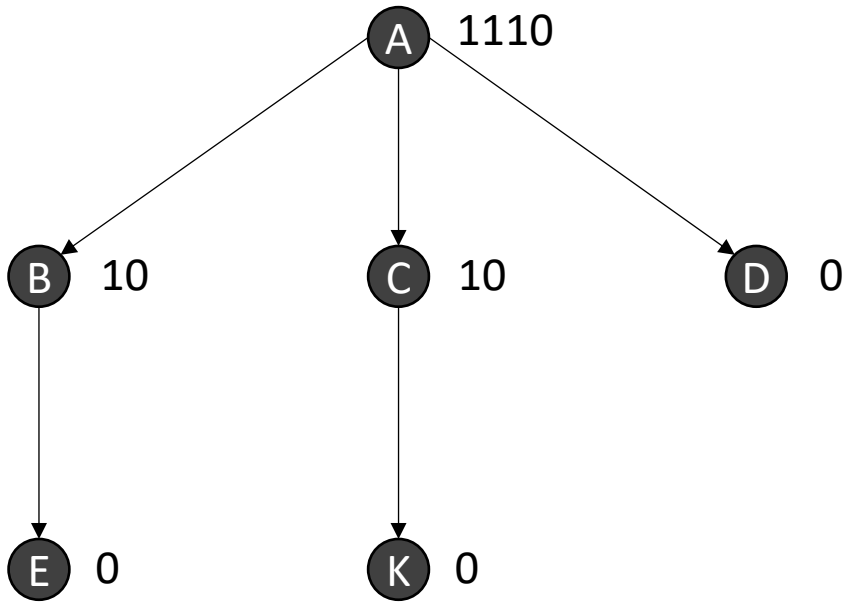
$select_0(x) = Die\ Position\ der\ x - ten\ Null$

$prev_0(x) = select_0(rank_0(x))$

$next_0(x) = select_0(rank_0(x) + 1)$

0	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	0	1	0	1	0	0	0	0

Parent von E

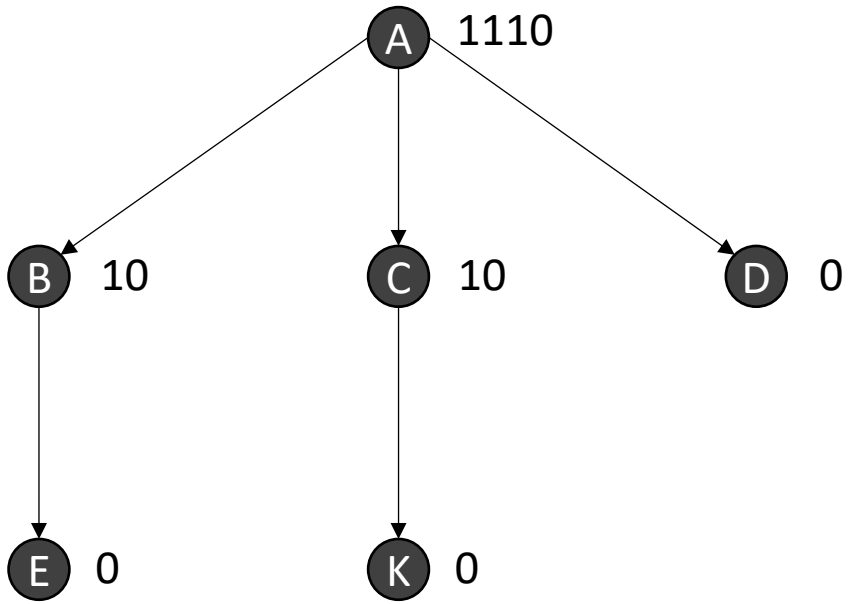


$$parent(x) = prev_0(select_1(rank_0(x))) + 1$$

$$parent(10) = prev_0(select_1(rank_0(10))) + 1$$

0	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	0	1	0	1	0	0	0	0

Parent von E



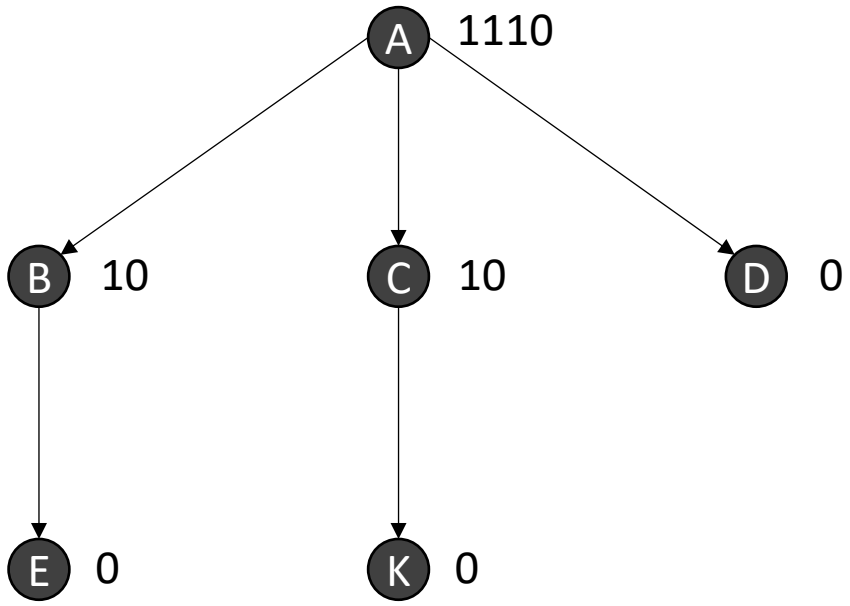
$$parent(x) = prev_0(select_1(rank_0(x))) + 1$$

$$parent(10) = prev_0(select_1(rank_0(10))) + 1$$

$$= prev_0(select_1(5)) + 1$$

0	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	0	1	0	1	0	0	0	0

Parent von E



$$parent(x) = prev_0(select_1(rank_0(x))) + 1$$

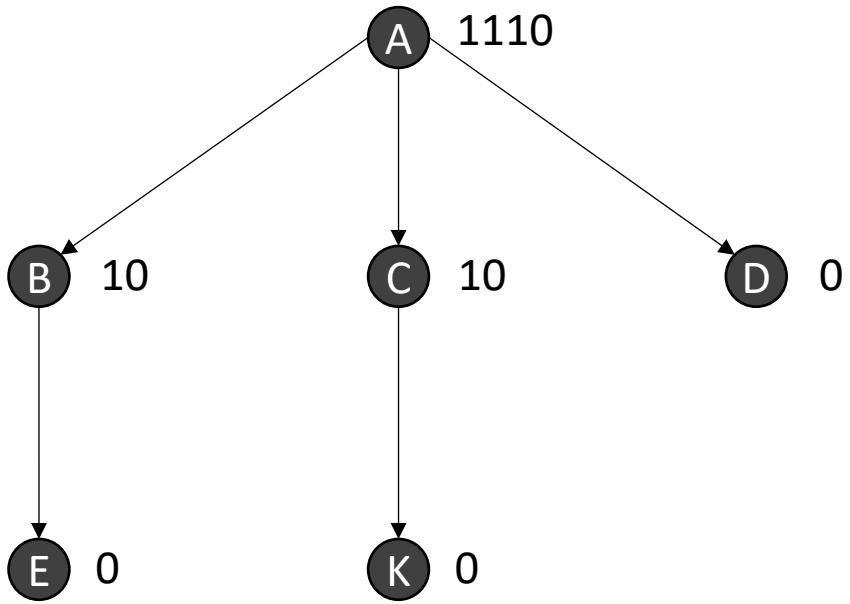
$$parent(10) = prev_0(select_1(rank_0(10))) + 1$$

$$= prev_0(select_1(5)) + 1$$

$$= prev_0(5) + 1$$

0	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	0	1	0	1	0	0	0	0

Parent von E



$$parent(x) = prev_0(select_1(rank_0(x))) + 1$$

$$parent(10) = prev_0(select_1(rank_0(10))) + 1$$

$$= prev_0(select_1(5)) + 1$$

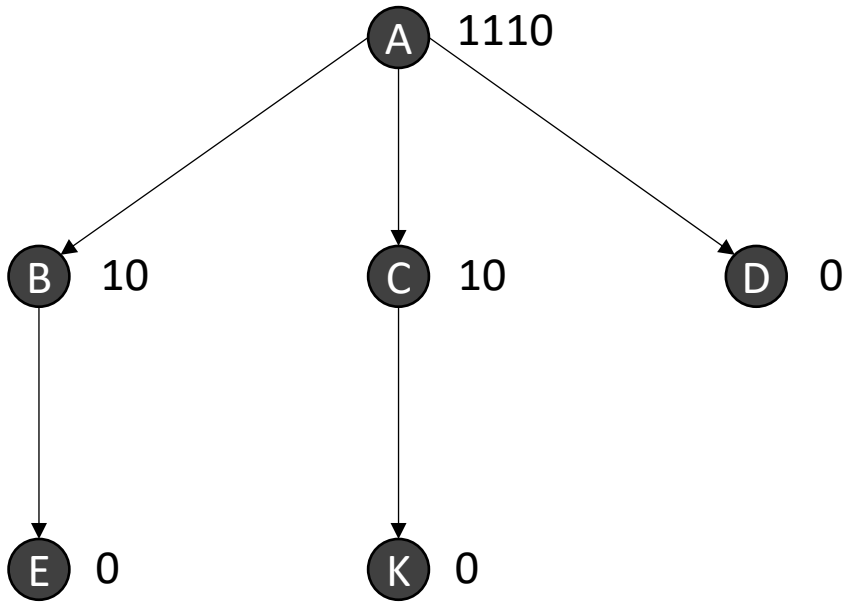
$$= prev_0(5) + 1$$

$$= 4+1$$

$$= 5$$

0	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	0	1	0	1	0	0	0	0

Third Child von A

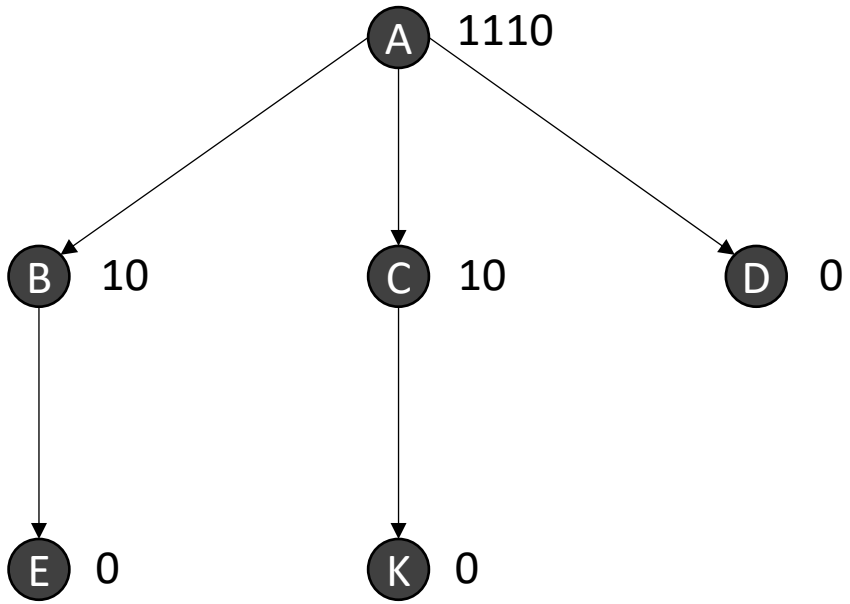


$$child(x,i) = select_0(rank_1(x) + i) + 1$$

$$child(1,2) = select_0(rank_1(1) + 2) + 1$$

0	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	0	1	0	1	0	0	0	0

Third Child von A



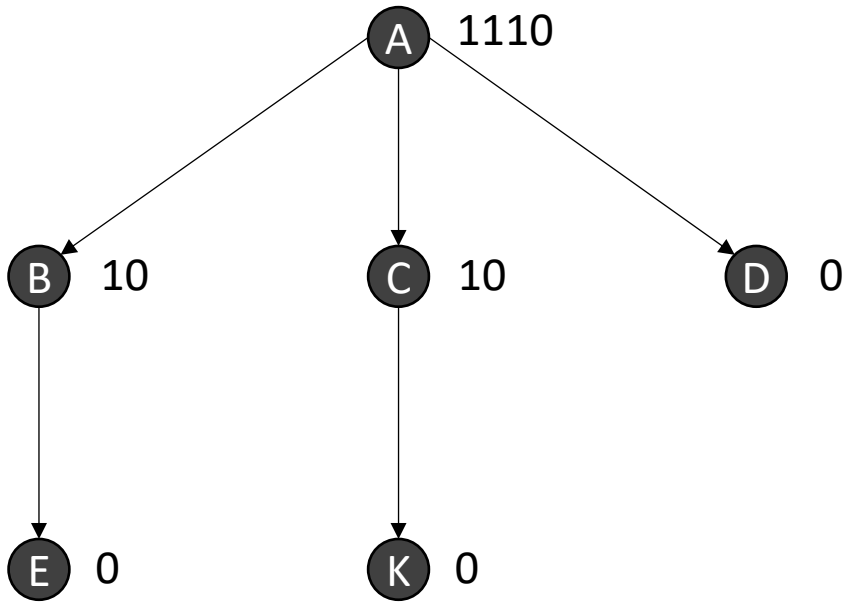
$$child(x,i) = select_0(rank_1(x) + i) + 1$$

$$child(1,2) = select_0(rank_1(1) + 2) + 1$$

$$= select_0(1 + 2) + 1$$

0	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	0	1	0	1	0	0	0	0

Third Child von A



$$child(x,i) = select_0(rank_1(x) + i) + 1$$

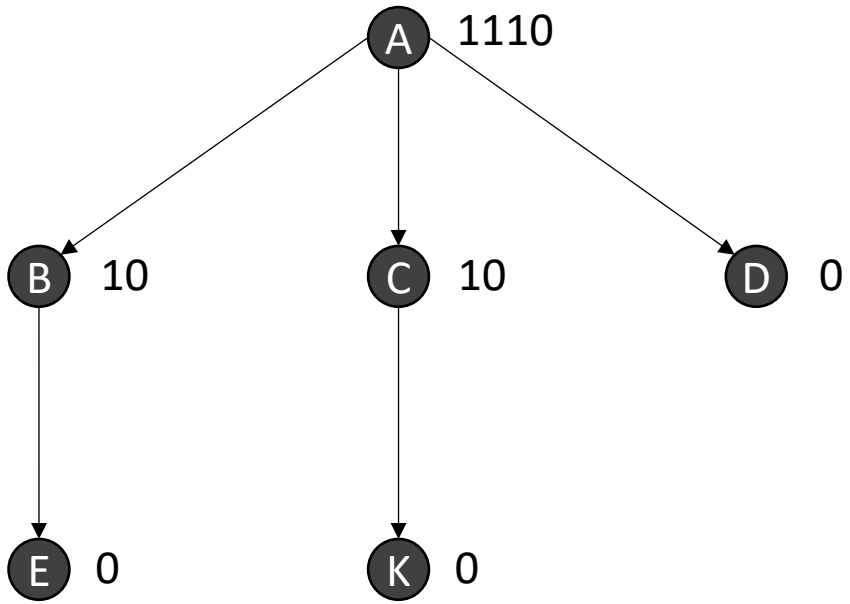
$$child(1,2) = select_0(rank_1(1) + 2) + 1$$

$$= select_0(1 + 2) + 1$$

$$= select_0(3) + 1$$

0	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	0	1	0	1	0	0	0	0

Third Child von A



$$child(x,i) = select_0(rank_1(x) + i\ ) + 1$$

$$child(1,2) = select_0(rank_1(1) + 2) + 1$$

$$= select_0(1 + 2) + 1$$

$$= select_0(3) + 1$$

$$= 8+1$$

$$= 9$$

0	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	0	1	0	1	0	0	0	0

LOUDS Zusammenfassung

- ✓ wenige Befehle in konstanter Zeit
- ✓ alle Befehle, die in konstanter Zeit arbeiten,
  auf $rank$ und $select$ zurückführbar
- ✓ einfach zu implementieren
- ✓ lightweight Brute-Force
- ✓ schneller als DFUDS und BP