Interval:

3 Methoden:

1. Von jedem Knoten Pointer für Predecessor und Successor vertauschen
2. Elemente powerweise tauschen
3. Neue Liste generieren, alte Knoten traversieren

O-Notation versteckt Konstanten

Anlegen von neuer Liste laufzeittechnisch „teuer", somit wenn möglich Referenzen verbiegen

OPT Suchbäume

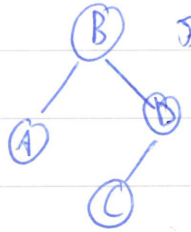
ALICE 0,25
BOB 0,2
CHARLIE 0,15
DAVID 0,4

$w_{i,j}$	A	B	C	D
A	0,25	0,45	0,6	1
B	0	0,2	0,35	0,75
C	0	0	0,15	0,55
D	0	0	0	0,4

$e_{i,j}$	0 "A"	1 "B"	2 "C"	3 "D"
1 "A"	0,25 (A)	0,65 (A)	1 (B)	1,95 (B)
2 "B"	0	0,2 (B)	0,5 (B)	1,25 (D)
3 "C"	0	0	0,15 (C)	0,7 (D)
4 "D"	0	0	0	0,4 (D)

Für root: $e_{1,r-1}$: Wurzel linker Unterbaum
$e_{r+1,n}$: Wurzel rechter Unterbaum

Für andere: l und r für Teilbaum anpassen



$w_{i,j}$: Summe der Zugriffswahrscheinlichkeiten von Knoten i bis j: $w_{i,j} = \sum_{k=i}^{j} p_k$

=> Ein Baum, der Schlüssel i bis j enthält, wird im Erwartungswert bei einer Suche nach einem zufälligen Schlüssel $w_{i,j}$ mal besucht

$e_{i,j}$: Erwartungswert der Anzahl der Vergleiche, die bei einer Suche in diesem Unterbaum durchgeführt werden (Knoten i bis j) (von optimalem Suchbaum)

-> Wir kennen diesen optimalen Suchbaum anfangs nicht.

-> Wir wissen aber: Jeder Unterbaum eines optimalen Suchbaums ist selbst ein optimaler Suchbaum!

-> Wir können die kleinsten Unterbäume sehr einfach konstruieren und daraus größere Optimale Suchbäume bauen ? -> Es ist nicht so einfach, die Struktur des Baums kann sich mit dem Einfügen jeder Knoten verändern!

Kurz Beispiel

-> Wenn $e_{1,n}$ minimal ist, ist unser Baum optimal bzgl. Anzahl Vergleiche.

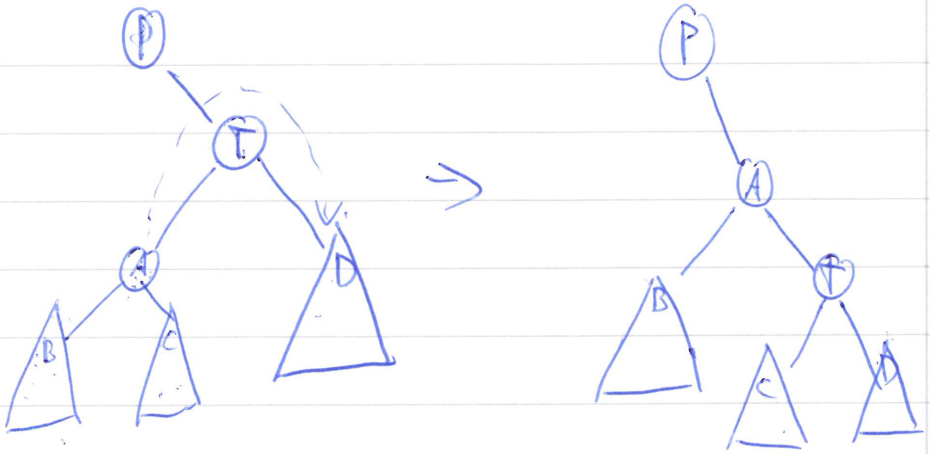
Wie berechnen wir $e_{i,j}$? Drei Fragen bzgl. Suche nach zufälligen Schlüssel:
(Anm.: Die Wurzel kennen wir (noch) nicht)

- Wie wahrscheinlich ist es, dass wir bei einer Suche den Wurzelknoten dieses Teilbaums besuchen? => d.h. Wie oft vergleichen wir mit der Wurzel?
- Wie viele Vergleiche erwarten wir im linken Teilbaum?
- Wie viele Vergleiche erwarten wir im rechten Teilbaum?

=> Zusammen! $\min_{i \le r \le j} \{ e_{i,r-1} + e_{r+1,j} \} + w_{i,j}$

$e_{j,i} = ?$ 0 für $j < i$?

Rechtsrotation:



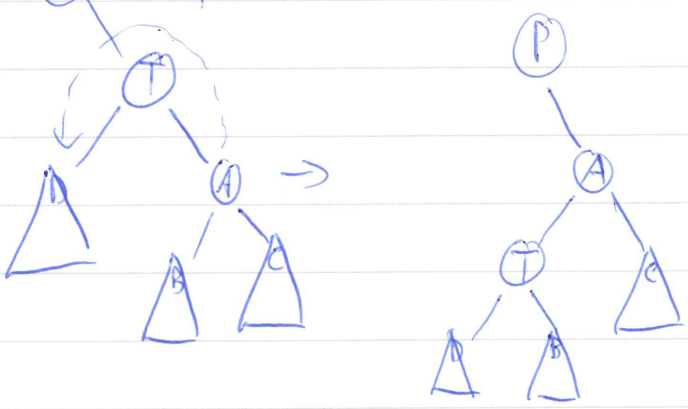
Umhängen von C: wir wissen, dass Wert von C zwischen A und T liegen -> Müssen diese vergleichen und wieder

Rechtsrotation um 5: P=2 T=5 A=4 , B,C,D: ∅

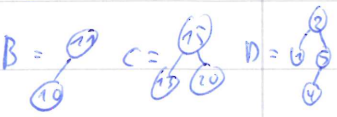
Rechtsrotation um 12: P=9 T=12 A=11 B=10 C=∅ D=

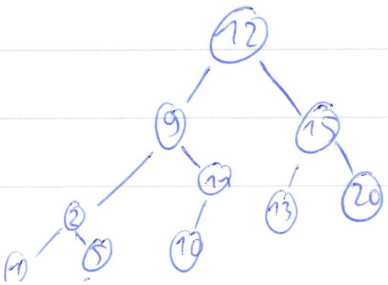
Rechtsrotation um 9:

Linksrotation:



Linksrotation um 9: P=∅ T=9 A=12 B= C= D=





Bemerkung: Löschen & Einfügen wiederholen

(3)

AVL-BÄUME

→ AVL-Eigenschaft?

Höhe des rechten und linken Unterbaums jedes Knotens unterscheidet sich höchstens um 1.

→ Höhe nach Einfügen verletzt? → rebalancieren! (beim Einfügen)

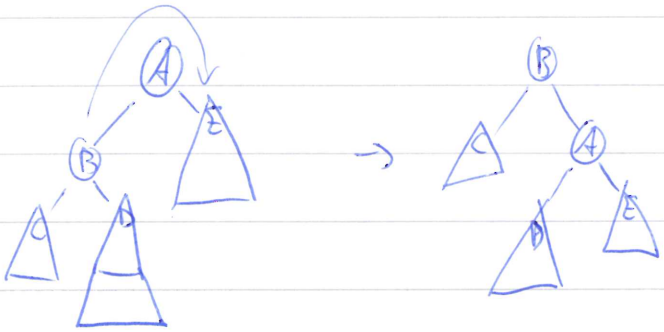
Wir brauchen immer höchstens zwei Rotationen → Warum?

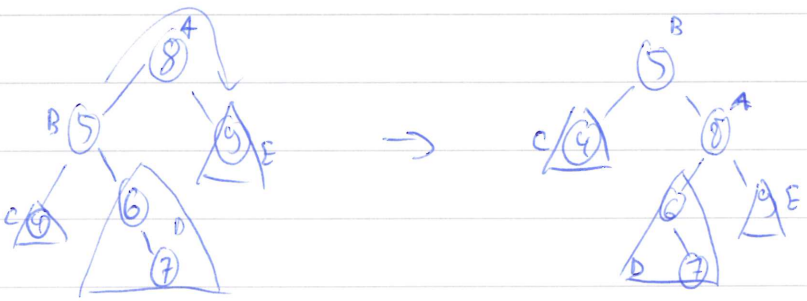
Einfach-Rotation: Rechter Teilbaum höher → / linker Teilbaum höher →

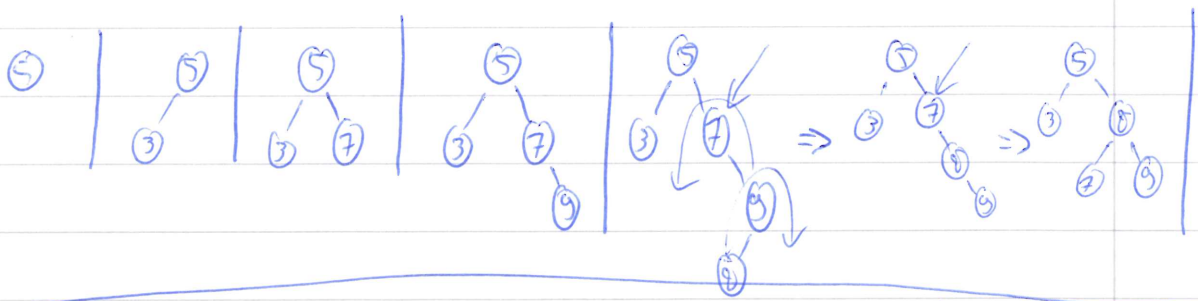
→ Rotiere um Knoten dessen Höhe verletzt ist.

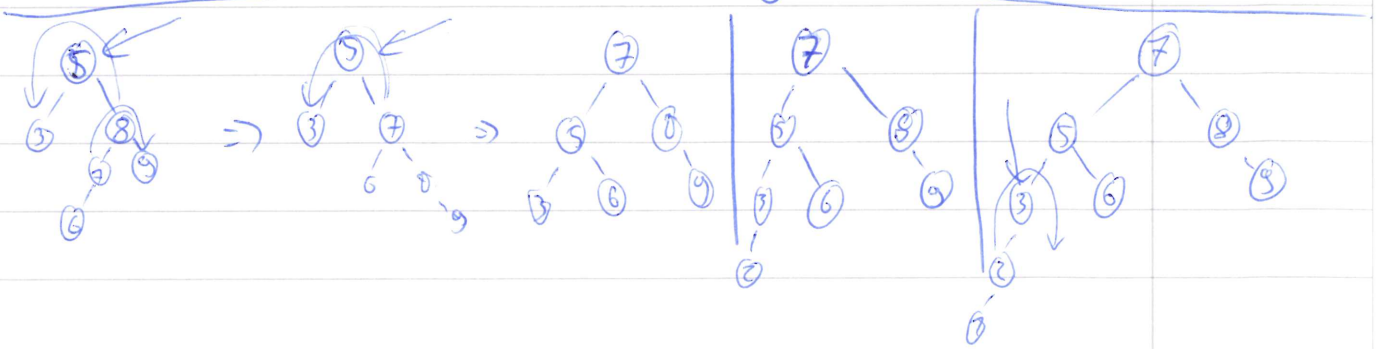
↳ Manchmal vorher Rotation im Unterbaum nötig

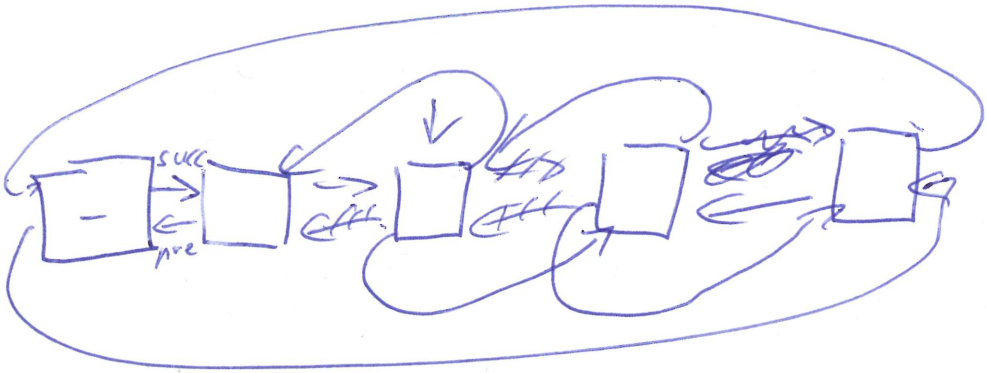
↳ Unterbaum der zu tief ist könnte durch Rotation nur umgehängt werden

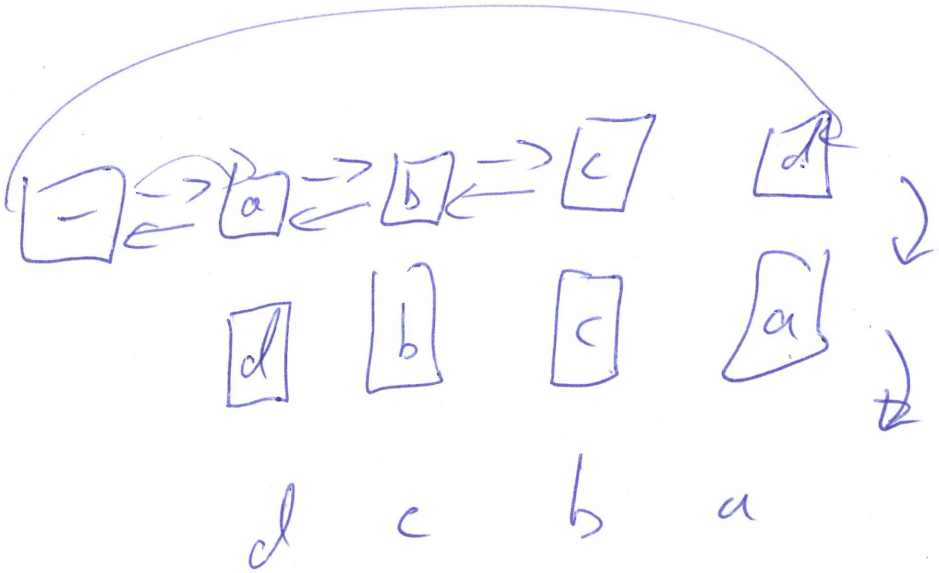




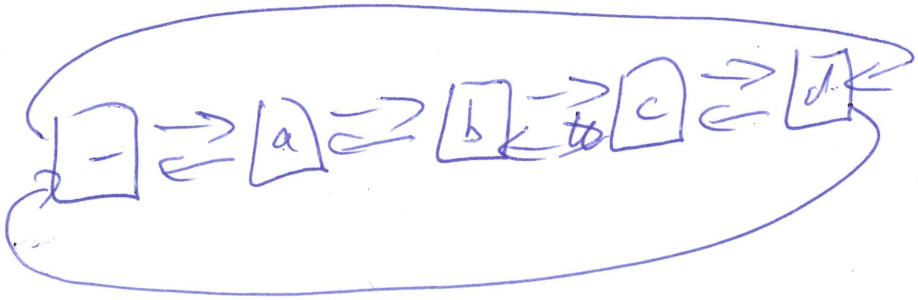


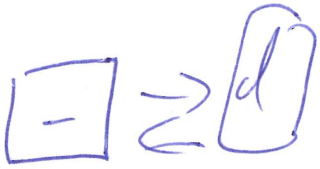






d c b a





Wann ist ein binärer Suchbaum mit Zugriffswahrscheinlichkeiten auf allen Schlüsseln optimal

↳ „Wenn wir schnell einen Schlüssel finden"

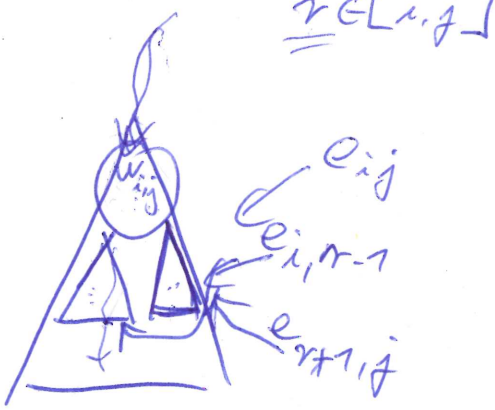
↳ Minimale Anzahl der erwarteten Vergleiche

$e_{i,j}$: Erwartungswert der Anzahl Vergleiche die bei einer Suche in einem Teilbaum eines optimalen Suchbaums benötigt werden

→ Wir kennen diesen optimalen Suchbaum (noch) nicht

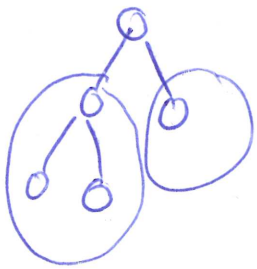
$e_{1,n}$:

$r \in [i,j]$



$e_{i,j}$ - Wir betrachten wahrscheinlich alle, dass wir bei einer Suche die Wurzel von $e_{i,j}$ berechnen

$$e_{i,j} = w_{i,j} + \min_{i \le r \le j} \{ e_{i,r-1} + e_{r+1,j} \}$$



$w_{i,j}$: Summe der Zugriffswahrscheinlichkeiten von Knoten i bis j

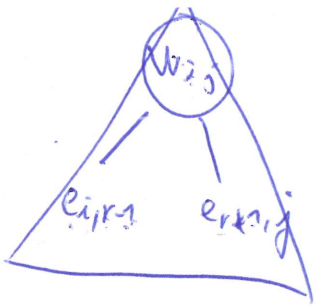
$$w_{i,j} = \sum_{k=i}^{j} p_k$$

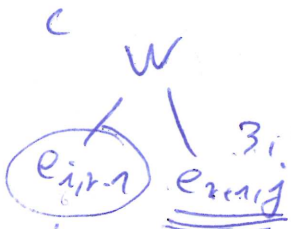
→ Ein Baum der Schlüssel i bis j enthält wird im Erwartungswert bei einer Suche nach einem zufälligen Schlüssel $w_{i,j}$ mal besucht

	0	1	2	3	4
0	w_{ij}	"A"	"B"	"C"	"D" ← j
1	A	0,25	0,45	0,6	1
2	B	0	0,2	0,35	0,75
3	C	0	0	0,15	0,55
4	D	0	0	0	0,4

↑ i

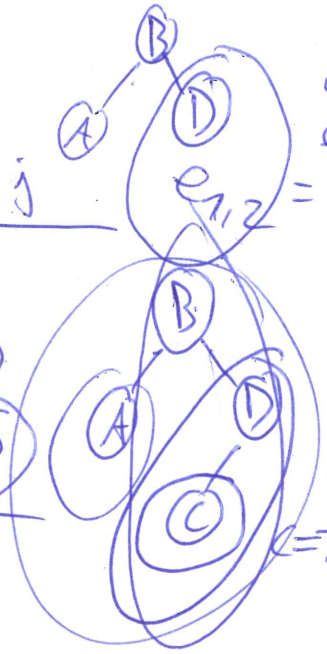
e_{ij}





	0	1	2	3	4
	e_{ij}	A	B	C	D ← j
1	A	(A) 0,25	(A) 0,65	(B) 1	(B) 1,95
2	B	0	(B) 0,2	(B) 0,5	(D) 1,25
3	C	0	0	(C) 0,15	(D) 0,7
4	D	0	0	0	(D) 0,4

↑ i



$e_{1,2,r-1}$

$e_{1,2,-1} = e_{1,1}$

$= w_{1,2} + \min_{1 \le r \le 2} \{ e_{1,1,r-1} + e_{r,2,1} \}$

Wand 1

$\{ e_{1,1,-1} + e_{2,2,1}$

$e_{1,1} + e_{3,2} \}$

Wand 2

D,C

$\Leftrightarrow 0,45 + \min \{ 0 + 0,2 ,$

$0,25 + 0 \}$

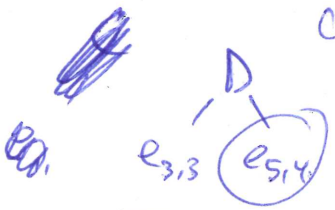
$e_{1,4} = w_{1,4} + \min_{1 \le r \le 4} \{ e_{1,r-1} + e_{r+1,4} \}$

$\Leftrightarrow 1 + \min \{ e_{1,0} + e_{2,4} ,$

$e_{1,1} + e_{3,4} ,$

$e_{1,2} + e_{4,4} ,$

$e_{1,3} + e_{5,4} \}$



$i = 3, j = 4$

← Wand 1

← Wand 2

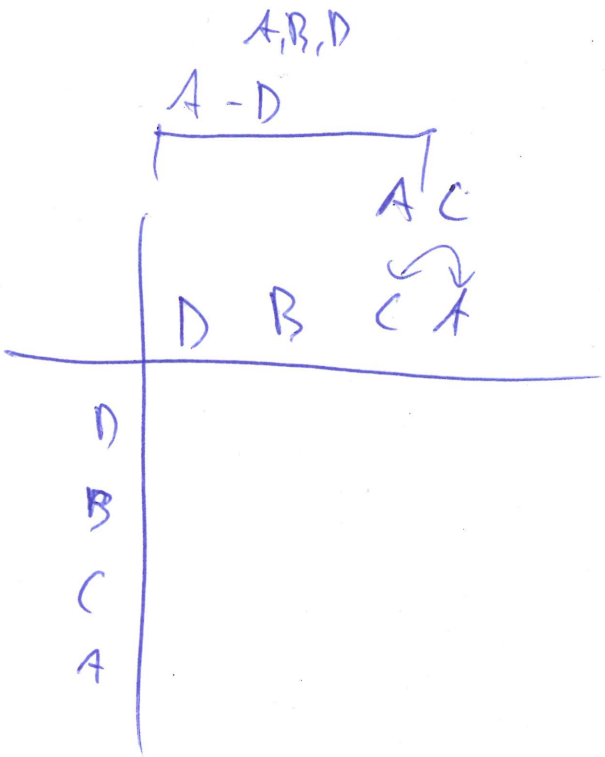
← Wand 3

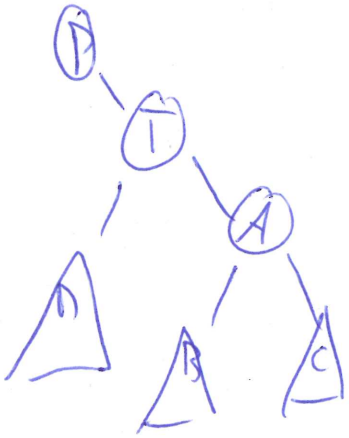
$\Leftrightarrow 1 + \min \{ 0 + 1,25 ,$

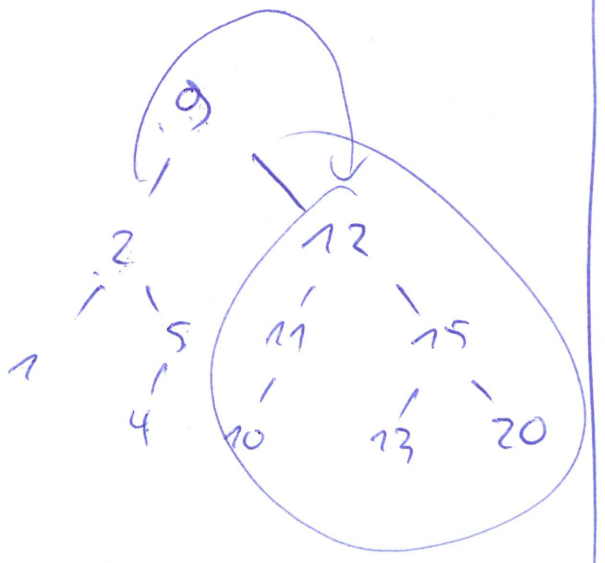
$0,25 + 0,7 , \in 0,95$ Wand 2 (B)

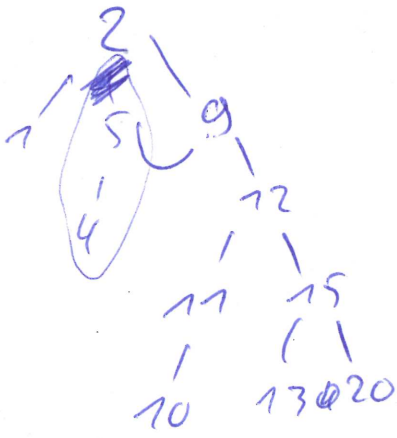
$0,65 + 0,4$

$1 + 0 \}$









$P = \phi$

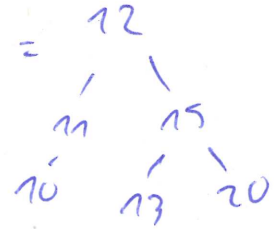
$t = 9$

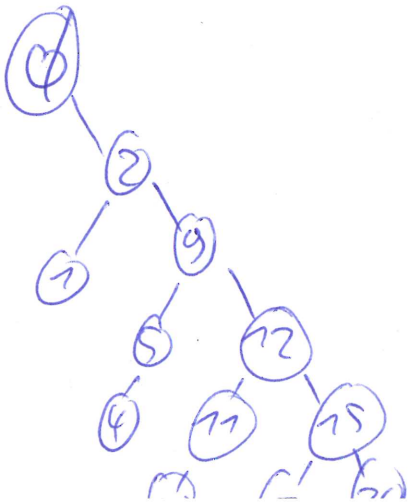
$A = 2$

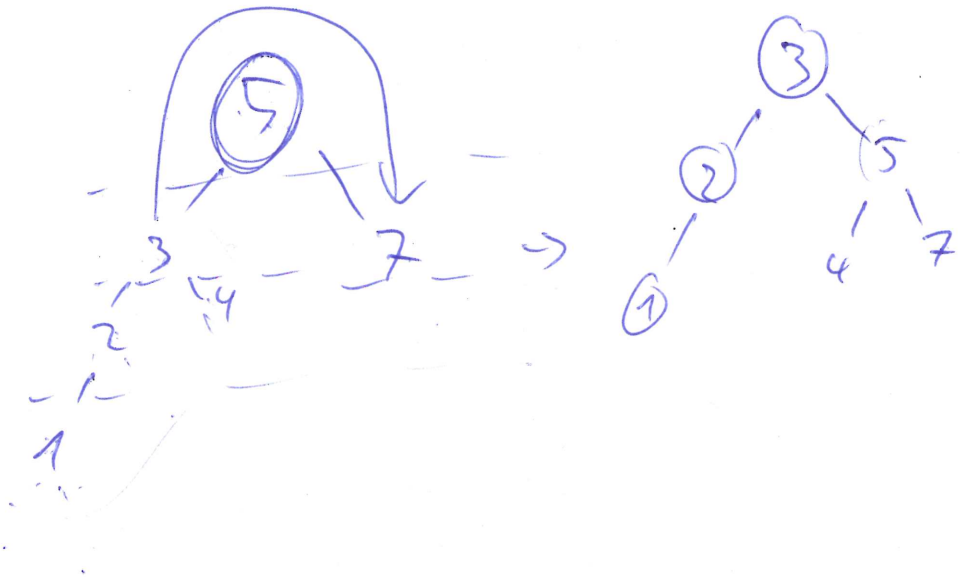
$B = 1$

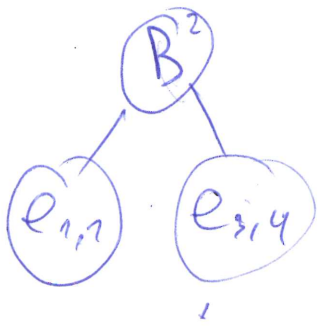
$C = 5, 4$

$D =$






5
3
7
4
2
1
3
2
5
1
4
7

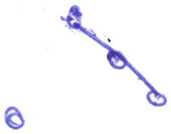


r=2
i=1
j=4 = D
i=C=3

r=4
i=

1< Knoten

Pro Knoten muss 3 Kanten
keine Kantenschnitte



3< Kanten

-2+1 = -1

$A \leq r \leq D$ e_{D-1}