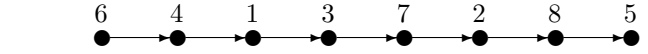
Beispiel 3.12

Gegeben ist eine Liste mit Rang und 3–Färbung in Klammern unter den Knoten:



R(i)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(0)
c(i)	(1)	(0)	(2)	(0)	(2)	(1)	(2)	(0)

Im Speicher sieht das S Array so aus:

i	1	2	3	4	5	6	7	8
S(0,i)	3	8	7	1	0	4	2	5

I. Iteration:

Die lokalen Minima bilden die unabhängige Menge $I = \{2, 3, 4, 5\}$.

Die zu entfernenden Knoten werden nun wieder in einer extra Liste zwischengespeichert. Dazu weit man jedem Knoten aus I eine neue Nummer mittels des $N(j, i)$ Array zu. Dabei steht i für die Knotennummer und j für die aktuelle Iteration. In der ersten Iteration bekommen wir:

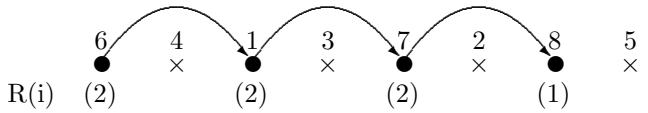
$$N(1,2) = 1,\ N(1,3) = 2,\ N(1,4) = 3,\ N(1,5) = 4.$$

Damit ist sichergestellt, daß die Knoteninformationen im Feld $U(j, N(i)) = (i, S(i), R(i))$ in aufeinander folgenden Speicherbereichen stehen.

$U(1, N(2)) = (2, 8, 1)$
$U(1, N(3)) = (3, 7, 1)$
$U(1, N(4)) = (4, 1, 1)$
$U(1, N(5)) = (5, 0, 0)$

Danach werden die Gewichte und die Pointer angepasst, mittels Algorithmus 3.4.

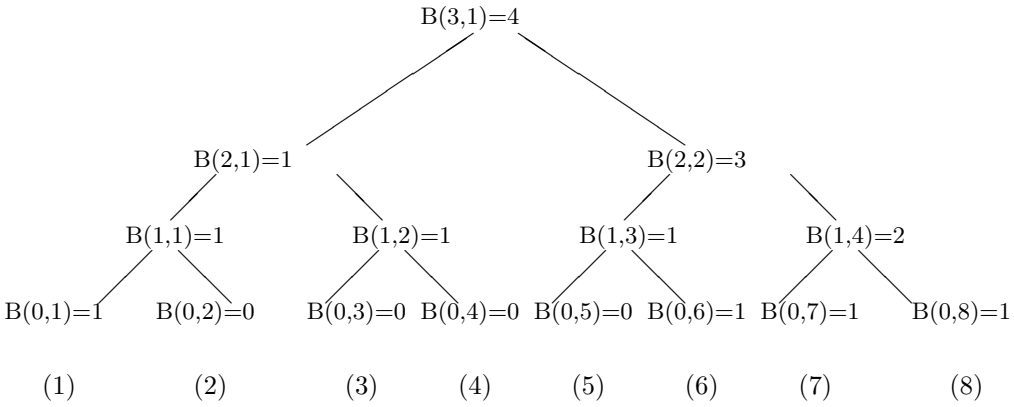
Wir bekommen jetzt die Liste:



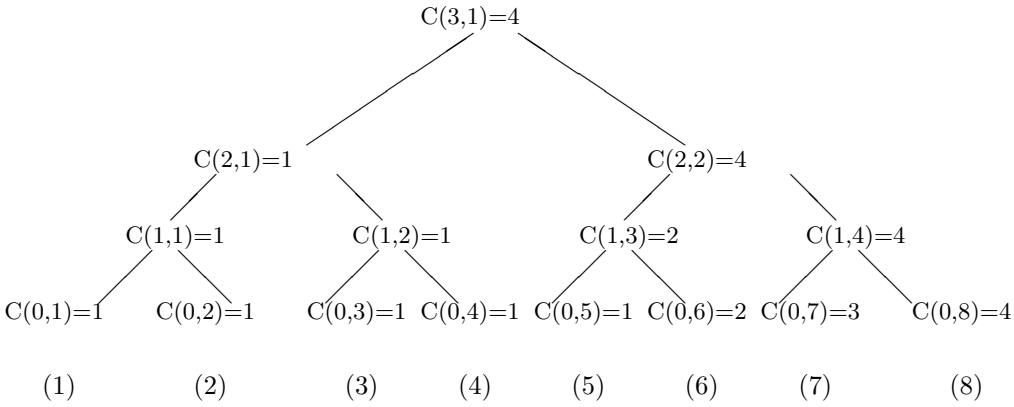
R(i)	(2)		(2)		(2)		(1)	

Der nächste Schritt ist 2.4 in Algorithmus 3.11. Das kompaktifizieren der Liste erfolgt mittels dem Präfix–Summen–Algorithmus. Jedem Knoten $i \notin I$ wird eine 1 zugewiesen. Alle anderen erhalten eine 0 als Wert der Präfix–Summe.

1.Phase:



2.Phase:



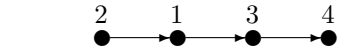
der Bereich des Feldes $C(0,i)$ gibt den entsprechenden Speicherplatz an. Als nächsten Schritt berechen wir die Relation von alten zu neuen Knotennummern. Dies geschieht mittels des Array $K(j,i)$, dabei steht j für die Iteration und i für die ursprüngliche Knotennummer.

i	1	2	3	4	5	6	7	8
K(1,i)	1	0	0	0	0	2	3	4

Daraus wir das neue S–Array generiert:

i	1	2	3	4
S(1,i)	3	1	4	0

Auerdem müssen die R–Werte angepasst werden. Zu beginn der 2.Iteration haben wir nun folgende Situation:



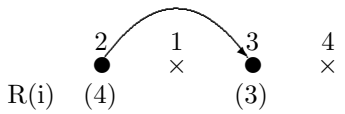
R(i)	(2)	(2)	(2)	(1)
c(i)	(1)	(0)	(1)	(0)

Daraus ergibt sich die unabhängige Menge $I = \{1, 4\}$.

$$N(2,1) = 1,\ N(2,4) = 2.$$

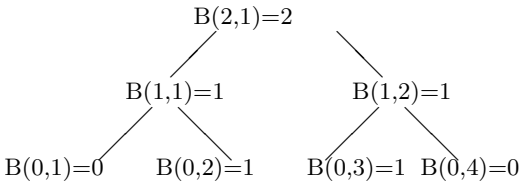
$U(2, N(1)) = (1, 3, 2)$
$U(2, N(4)) = (4, 0, 1)$

Wieder wurden die Knoten entfernt und im U–Array zwischengespeichert. Danach hat die Liste folgende Gestalt:



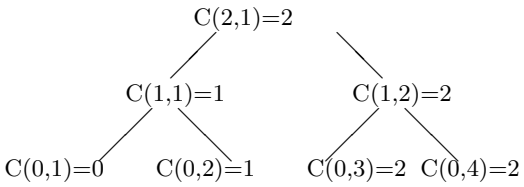
R(i) (4) (3)

Auch hier wird wieder mittels Präfix–Summe kompaktifiziert:
1.Phase:



(1) (2) (3) (4) Knoten i

2.Phase:

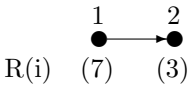


(1) (2) (3) (4) Knoten i

Daraus ermitteln wir das $K(2,i)$ und das $S(2,i)$ Array:

i	1	2	3	4	i	1	2
K(2,i)	0	1	2	0	S(2,i)	2	0

Nun ist die Liste auf $\frac{n}{\log n}$ geschrumpft. Die while–Schleife wird verlassen. Es folgt der 3. Schritt von Algorithmus 3.11, mittels Pointer jumping auf der kurzen Liste werden die Ränge ermittelt.



Die Rekonstruktion der ursprünglichen Liste erfolgt nun in k Iterationen, die die Iterationen der while–Schleife umkehren.
Dazu durchläuft man die Iterationen rückwärts.