Die 6 Farben können wir nun auf 3 reduzieren, indem wir die Farben $c \in \{3,4,5\}$ verändern.

Algorithmus 2.43 (3–Färbung)
Eingabe: gerichteter Kreis $G = (V, E)$ mit Färbung $c \in \{0, ..., 5\}$
Ausgabe: 3–Färbung

for $3 \leq l \leq 5$ do
  for $1 \leq i \leq n$ pardo
    if $c(i) = l$ then
      {Bestimme die Farbe $h \in \{0,1,2\}$ mit $c(S(i)) \neq h$
      und $c(P(i)) \neq h$}
      $c(i) = h$

Jede Iteration benötigt $O(1)$ Zeit und $O(n)$ Arbeit.

Algorithmus 2.42 benötigt $O(1)$ parallele Zeit. Nach $O(\log^* n)$ Iterationen erhällt man eine gültige 6–Färbung.
Danach folgt Algorithmus 2.43, der daraus eine gültige 3–Färbung erzeugt. Dieser braucht parallele Zeit $O(1)$ und $O(n)$ Arbeit.
Damit braucht man Gesamt $O(\log^* n)$ Zeit und Arbeit $O(n \cdot \log^* n)$.
Leider ist der Algorithmus aufgrund des $\log * n$ Faktors nicht optimal.
Unser Ziel ist es also diesen Faktor zu eliminieren. dazu machen wir folgende
Beobachtung: Das Sortieren von n Zahlen aus der Menge $\{0, ..., O(\log n)\}$ kann in Zeit $O(\log n)$ und Arbeit $O(n)$ parallel erfolgen.
Man benutzt dazu eine Kombination von Radix–Sort und dem Präfix–Summen–Algorithmus.
Der optimale Algorithmus arbeitet nun in 3 Schritten:
1. wende Algorithmus 2.42 einmal an
2. Sortiere die Knoten der Farbe nach
3. Benutze Algorithmus 2.43

Ausformuliert sieht das dann folgendermaßen aus:

Algorithmus 2.44: (optimale Färbung)
Eingabe: gerichteter Kreis $G = (V, E)$ mit $i \in V$ für $1 \leq i \leq n$. E ist durch das Array S gegeben.
Ausgabe: 3–Färbung $c(i)$

begin
1. for $1 \leq i \leq n$ pardo
   $c(i) = i$; {initiale Färbung}
2. { Ausführen von Algorithmus 2.42 }
3. Sortiere die Knoten nach ihrer Farbe

4. for $i = 3$ to $2 \cdot \lceil \log n \rceil$ do
   for alle Knoten v mit $c(v) = i$ pardo
   {Bestimme die Farbe $h \in \{0, 1, 2\}$ mit $c(S(i)) \neq h$
   und $c(P(i)) \neq h$}

end

Wir können nun einen gerichteten Kreis mit n Knoten in Zeit $O(\log n)$ und Arbeit $O(n)$ 3–färben.

Beweis:

Schritt 1 und 2 benötigen $O(1)$ parallele Zeit und $O(n)$ Operationen.

Schritt 3 braucht $O(\log n)$ Zeit und $O(n)$ Arbeit.

Die Analyse von Schritt 4 ist etwas komplizierter. Da wir die Knoten nach Farben sortiert haben, liegen Knoten gleicher Farbe in aufeinanderfolgenden Speicherplätzen.

Wir nehmen an, daß wir den Platz des 1. und letzten Knotens einer Farbe kennen. Die Anzahl der Knoten mit Farbe i sei n_i.

Dann können wir jeden Schleifendurchlauf in konstanter Zeit mittels $O(n_i)$ Operationen bearbeiten.

Für den Arbeitsaufwand ergibt sich somit

$$W(n) = O(n + \sum_{i=3}^{2 \cdot \lceil \log n \rceil} n_i) = O(n)$$

Da wir keinen simultanen Speicherzugriff haben, kann der Algorithmus auf einer EREW-PRAM impelementiert werden.