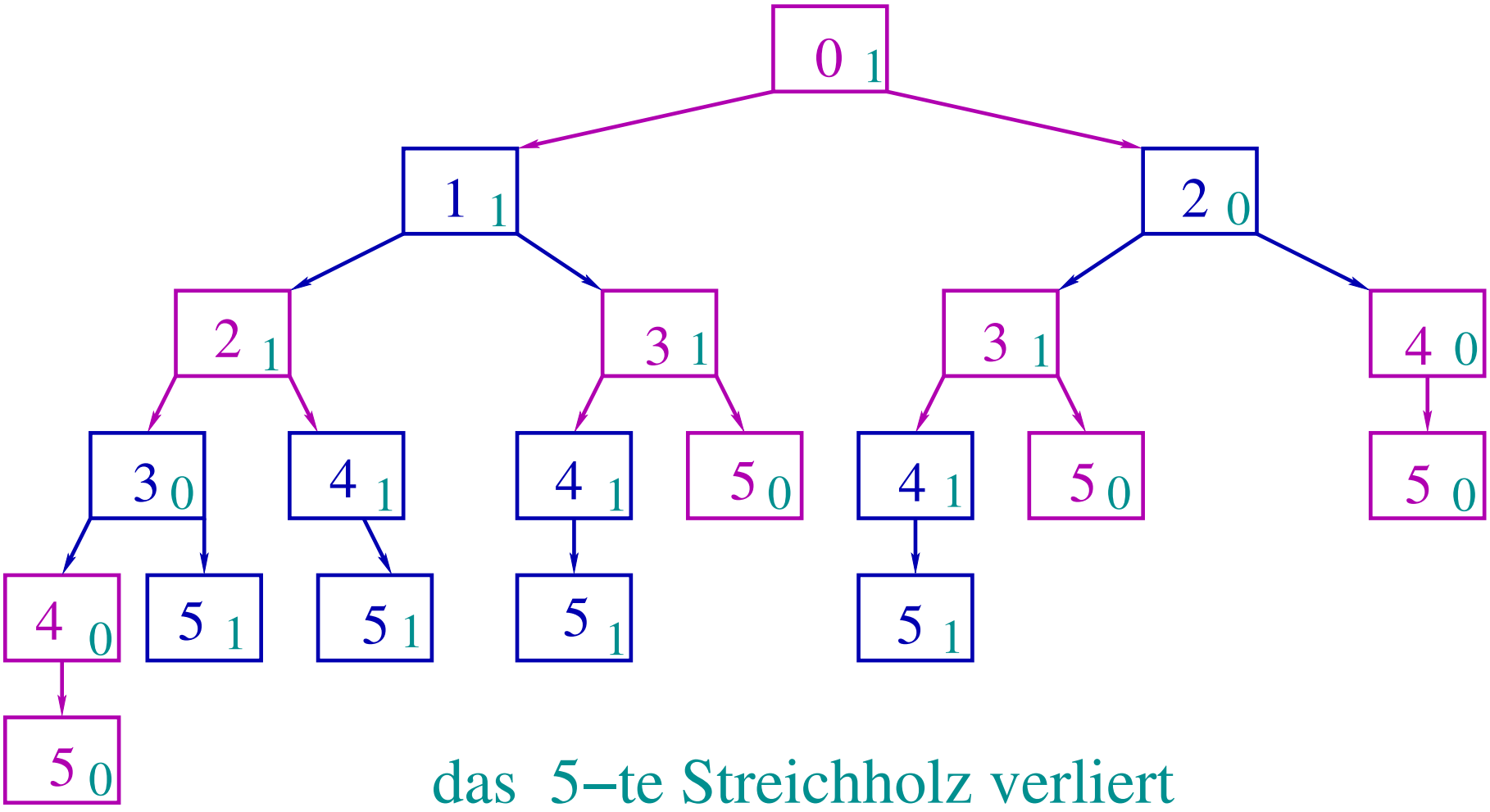
Streichholz-Spiel



das 5–te Streichholz verliert

Spielbäume

- Jedes (mehrzügige) Zwei-Personen Spiel mit beschränkt langen Spielen besitzt einen Spielbaum
- Knoten mit geradem Abstand von der Wurzel sind MAX-Knoten in denen der MAX-Spieler (Alice) am Zug ist
- Knoten mit ungeradem Abstand von der Wurzel sind MIN-Knoten in denen der MIN-Spieler (Bob) am Zug ist
- der *Spielwert in einem Blatt* repräsentiert die Auszahlung an den MAX-Spieler wenn das Spiel in diesem Blatt endet.

Definition:

Der *Spielwert eines beliebigen MAX-Knotens* ist das Maximum der Spielwerte seiner Kinder-Knoten.

(= maximale Auszahlung die Alice in jedem Fall erreichen kann)

Der *Spielwert eines beliebigen MIN-Knotens* ist das Minimum der Spielwerte seiner Kinder-Knoten.

(= minimale Auszahlung an Alice die Bob in jedem Fall erreichen kann)

Auswertung von Spielbäumen

Eingabe: Ein Spielbaum T mit einem Spielwert in jedem Blatt

Aufgabe: Auswertung des Spielbaums T, also insbesondere Spielwert der Wurzel

(= Auszahlung an Alice die sie in jedem Fall erreichen kann, egal wie Bob spielt)

Theorem:

Sei A ein deterministischer Algorithmus, der den Baum T_k^t durch Abfragen der Blattwerte auswertet. Dann **gibt es eine Eingabe** (Wertezuweisung der Blätter), so dass A **alle** k^{2t} **Blätter von** T_k^t **inspizieren** muss.

Entscheidungsprobleme

Wir betrachten Entscheidungsprobleme (Sprachprobleme):

Sei $L \subset \Sigma^*$ eine Sprache.
Es gilt für jedes Wort w entweder $w \in L$ oder $w \notin L$.

Ein Algorithmus für das Problem soll die Eingabe w *akzeptieren* oder *verwerfen*.

Definition:

Ein randomisierter Algorithmus *akzeptiert* w *mit der* **Akzeptanzwahrscheinlichkeit** p, falls für die Summe über alle Berechnungen B gilt

$$\sum_{B:B \text{ akzeptiert } w} p_B = p.$$

Monte Carlo Algorithmen

Definition:

A ist ein **Monte Carlo Algorithmus mit einseitig beschränktem Fehler** für die Sprache L, falls für jede Eingabe w gilt

$$w \in L \quad \Rightarrow \quad \Pr[A \textit{ akzeptiert } w] \geq 1/2$$

$$w \notin L \quad \Rightarrow \quad \Pr[A \textit{ verwirft } w] = 1.$$

Der Wert $1/2$ ist nicht wesentlich!

Eine beliebige Erfolgswahrscheinlichkeit $q > 0$ oder sogar $1/\mathit{poly}(n)$ reicht aus für Erfolgswahrscheinlichkeit $\geq 1/2$, bei polynomiell vielen, unabhängigen Berechnungen.

Monte Carlo Algorithmen (2)

Definition:

A ist ein **Monte Carlo Algorithmus mit zweiseitig beschränktem Fehler** für die Sprache L, falls für jede Eingabe w gilt

$$w \in L \quad \Rightarrow \quad \Pr[A \textit{ akzeptiert } w] \geq 2/3$$

$$w \notin L \quad \Rightarrow \quad \Pr[A \textit{ verwirft } w] \geq 2/3.$$

Der Wert 2/3 ist nicht wesentlich, analog zum Fall mit einseitigem Fehler.

Wir betrachten vorerst nur Algorithmen mit einseitigem Fehler.

Primzahltest

Eingabe: Eine Zahl n in binärer Codierung

Ausgabe: Aussage "n ist prim" oder "n ist nicht prim"

Erweiterter Fermat-Test

1. Sei $n = 2^k \cdot m$ mit m ungerade
2. **if** $a^{n-1} \not\equiv 1 \mod n$ **then return**("nicht bestanden")
3. **if** es gibt $0 \leq h \leq k-1$ mit
   $a^{(n-1)/2^h} \equiv 1 \mod n$ und $a^{(n-1)/2^{h+1}} \not\equiv 1, -1 \mod n$
4. **then return**("nicht bestanden")
5. **else return**("bestanden")

Randomisierter Miller-Rabin Test

1. Sei n Eingabezahl und k maximal mit $n-1 = 2^k \cdot m$
2. Wähle $a \in \{2, \ldots, n-2\}$ uniform zufällig.
3. **if** n gerade **oder** $ggt(a, n) \neq 1$
   oder (n, a) besteht erweiterten Fermat-Test nicht
4. **then return**("n nicht prim")
5. **else return**("n ist prim")

Theorem:

Wenn die Zahl n zusammengesetzt ist, dann fehlklassifiziert der Miller-Rabin Test sie als Primzahl mit Wahrscheinlichkeit höchstens $1/4$. Im Test werden $O(\log_2 n)$ arithmetische Operationen durchgeführt.

Min-Cut Problem

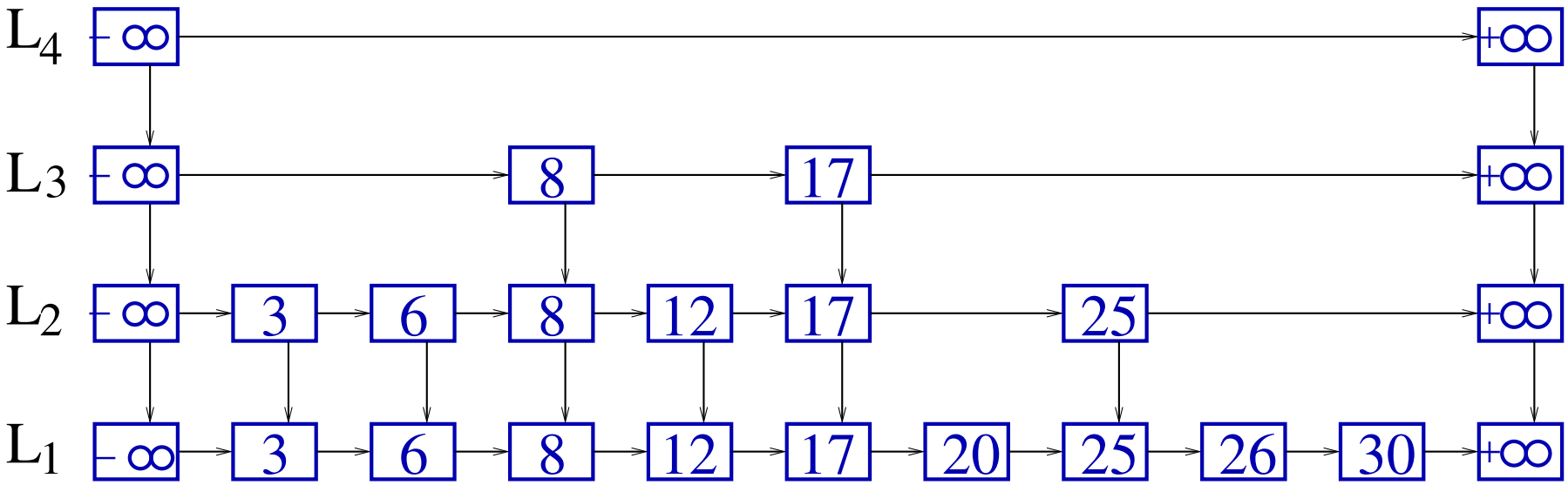
Eingabe: Multigraph $G(V, E)$ (Mehrfachkanten möglich)

Ausgabe: Minimaler Schnitt (V_1, V_2)

Randomisierter Min-Cut Algorithmus

1. **for** $i = 1$ to $|V| - 2$ **do**
2. Wähle zufällig gleichverteilt eine Kante
3. Kontrahiere diese Kante (Endknoten verschmelzen)
4. Am Ende sind noch Knoten v_1 und v_2 übrig, in die die Knotenmengen V_1 und V_2 kontrahiert wurden.
5. Gib (V_1, V_2) als minimalen Schnitt aus

Skip-Liste Beispiel



Skip-Listen: Lookup(x)

1. Fange in höchster Schicht L_t an; sei $y = -\infty$
2. **repeat**
3. $t \leftarrow t - 1$, gehe eine Schicht runter zur Kopie von y
4. Finde in Schicht L_t größten Schlüssel y', der höchstens x ist
5. **if** $y' = x$ **then return**(x); **else** setze $y \leftarrow y'$;
6. **until** $t = 1$, unterste Schicht fertig durchsucht
7. **return**("Nicht gefunden")

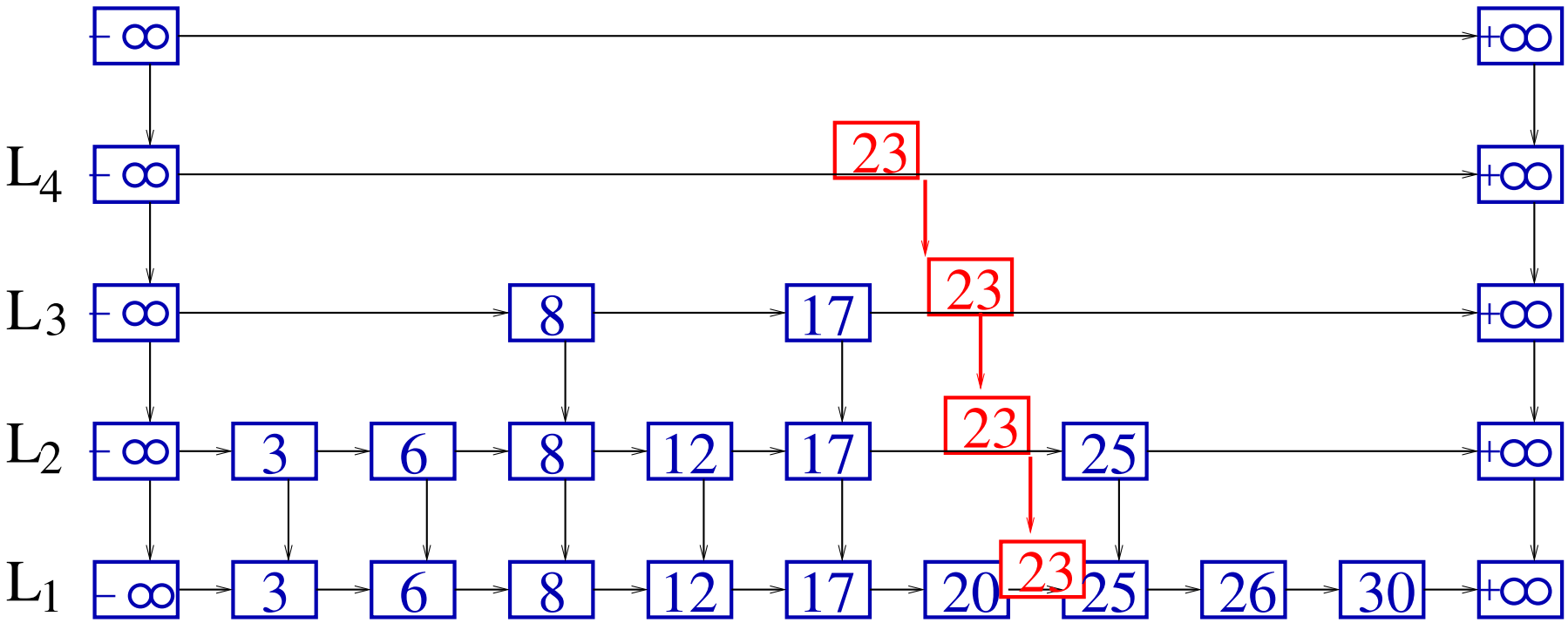
Skip-Listen: Delete(x)

1. Führe Lookup(x) aus
2. **if** x gefunden **then**
3. Sei $s \leftarrow \ell(x)$ die oberste Schicht mit x
4. Entferne x aus allen Schichten $L_s, L_{s-1}, \ldots, L_1$
5. Aktualisiere die Zeiger
6. Entferne ggf. überflüssige leere Schichten
7. Aktualisiere Index t der höchsten nicht-leeren Schicht
8. **else return**("x nicht gefunden")

Skip-Listen: Insert(x)

1. $s = 0$
2. **repeat**
3. $\quad s \leftarrow s + 1$, ziehe Zufallsbit b_s
4. $\quad$ **if** $b_s = 1$ **then** setze höchste Schicht für x auf $\ell(x) \leftarrow s$
5. **until** $b_s = 1$
6. Sei L_t die bislang höchste Schicht
7. **if** $\ell(x) \geq t$ **then**
8. $\quad$ Erzeuge leere Schichten $L_{t+1}, L_{t+2}, \ldots, L_{\ell(x)+1}$
9. $\quad$ Aktualisiere Index der höchsten Schicht auf $t \leftarrow \ell(x) + 1$;
10. Führe Lookup(x) durch und füge x ab Schicht $L_{\ell(x)}$ an den entsprechenden Positionen ein

Skip-Listen



w.h.p.

Definition:

Wir sagen, dass ein Ereignis E **mit hoher Wahrscheinlichkeit** (engl: **with high probability**, kurz **w.h.p.**) eintritt, wenn gilt:

Es gibt ein konstantes $c \geq 1$, so dass die Wahrscheinlichkeit für E mindestens

$$\Pr[E] \quad \geq \quad 1 - \frac{1}{n^c}$$

ist.

Dabei ist n die Eingabelänge, und wir können annehmen, dass $n > n_0$ für einen Mindestwert n_0.

Fingerprinting Motivation

Gleichheitstest (Vergleich der Inhalte weit entfernter Datenbanken)

- System R_I (in Europa) speichert eine Datenbank (Bit-Folge x)
- Anderes System R_{II} (in den USA) soll Kopie enthalten.
- Datenbank ist riesig: $|x| = n = 10^{16}$ Bits
- Das Kommunikationsprotokoll zwischen R_I und R_{II} soll entscheiden ob die Versionen der Datenbank gleich sind.
- Jedes deterministische Protokoll muss $\geq n$ Bits austauschen

Primzahlensatz

Sei $\pi(x)$ sei die Anzahl der Primzahlen in $[2, x]$. Es gelten

$$\frac{x}{\pi(x)} = \ln x + o(\ln x) \qquad \text{und} \qquad \lim_{x\to\infty} \frac{\pi(x)}{x/\ln x} = 1$$

Daneben gilt

$$\pi(x) > \frac{x}{\ln x} \qquad (x > 67)$$

Fingerprinting (allgemein)

- Ziel: Prüfe (viel zu) große Bit-Folgen x und y auf Gleichheit
- Stattdessen: Vergleiche (viel kürzere) **Fingerabdrücke** $h(x), h(y)$
- Dann haben aber viele Bit-Folgen denselben Fingerabdruck...

- Ansatz: Nutze **Menge M** von Abbildungen h
- Wähle eine Abbildung $h \in M$ **zufällig** aus

- Für jedes **feste** Paar $x \neq y$ wird eine **zufällig gewählte** Abbildung h mit guter Wahrscheinlichkeit $h(x) \neq h(y)$ liefern
- Um diese Wahrscheinlichkeit zu erhöhen kann man den Vergleich mehrmals unabhängig wiederholen.

Hashing

Zur Erinnerung: Hashing mit Verkettung

- Sei $\mathcal{U}$ ein Universum von möglichen Schlüsseln
- Schlüssel werden in einem Array von Listen (Hashtabelle) gespeichert
- Unterstützte Operationen *insert()*, *delete()*, *lookup()*
- Hashfunktion $h : \mathcal{U} \to \{0, 1, 2, \ldots, m-1\}$ bestimmt Zelle, in der ein Schlüssel gespeichert wird
- Da $m \ll |\mathcal{U}|$, wird $h(x) = h(y)$ für viele $x \neq y$ gelten

Universelles Hashing

- Wähle Hashfunktion uniform zufällig aus Klasse H von Funktionen
- H soll c-**universell** sein:
  Für jedes feste Schlüsselpaar $x \neq y$ soll gelten $\Pr[h(x) = h(y)] \leq \frac{c}{m}$.

Analyse der Laufzeit von n Operationen

Bezeichne K_n die *erwartete* Anzahl der Schlüssel in einer Zelle nach $n-1$ Operationen bei aktueller Schlüsselmenge S.

$$
\begin{aligned}
K_n &= \sum_{h \in H} \frac{1}{|H|} |\{y \in S \mid h(x) = h(y)\}| \\
&= \frac{1}{|H|} \sum_{h \in H} \sum_{y \in S \,:\, h(x)=h(y)} 1 \\
&= \frac{1}{|H|} \sum_{y \in S} \sum_{h \in H \,:\, h(x)=h(y)} 1 \\
&= \frac{1}{|H|} \sum_{y \in S} |\{h \,:\, h(x) = h(y)\}| \\
&\leq \frac{1}{|H|} \sum_{y \in S} c \cdot \frac{|H|}{m} = c \cdot \frac{|S|}{m} \leq c \cdot \frac{n-1}{m}
\end{aligned}
$$

Symmetry Breaking

- In fast homogenen Strukturen (Mengen, Graphen) werden Rollen oder Prioritäten zugewiesen

- Oder unter vielen symmetrischen Lösungen muss eine Lösung ausgewählt werden

- Ansatz:
  Jede Einheit (Knoten, Element...) wählt zufällig eine Rolle aus

Shared-Memory Architekturen

CRCW PRAM Modell

- Mehrere Prozessoren mit je einem lokalen Speicher
- Haben Zugriff auf den selben globalen Speicher
- In synchronisierten parallelen Schritten:
  1. Ein globales Register lesen
  2. Lokale Operationen durchführen
  3. Versuchen ein globales Register zu beschreiben

     (wenn mehrere Prozessoren dasselbe Register schreiben wollen, gewinnt irgendeiner von ihnen)

Zusammenhangskomponenten

Randomisierter paralleler Algorithmus

1. **for** $i = 1$ to n **pardo** $Vater[i] = i$;
2. **while** es lebendige Kanten gibt **pardo**
3. Jede Wurzel wählt Geschlecht aus {männl.,weibl.} uniform zufällig
4. Die Kinder übernehmen dieses Geschlecht
5. **for** alle lebendigen $\{u, v\}$ mit
   u zu männl. Wurzel w_0 und v zu weibl. Wurzel w_1 **pardo**
6. Kanten-Prozessor uv *versucht* $Vater[w_0] := w_1$ zu setzen
   (irgendeiner ist erfolgreich)
7. **for** $i = 1$ to n **pardo**
8. $Vater[i] := Vater[Vater[i]]$
   (die Kinder von all diesen w_0 übernehmen den neuen Vater)
9. Alle Kanten $\{u, v\}$ mit $Vater[u] = Vater[v]$ nicht mehr lebendig

Unabhängige Mengen

Randomisierter paralleler Algorithmus

1. Setze $I \leftarrow \emptyset$
2. **while** $V \neq \emptyset$ **pardo**
3. **if** v isoliert
4. **then** $I \leftarrow I \cup \{v\}$ und $V \leftarrow V \setminus \{v\}$
5. **else** Markiere v mit Wahrscheinlichkeit $1/(2\deg(v))$
6. **for** $\{u, v\} \in E$ mit markierten Endknoten u, v **pardo**
7. Lösche die Markierung vom Endknoten mit kleinerem Grad (beide bei Gleichheit der Grade)
8. Sei X Menge der markierten Knoten.
9. Setze $I \leftarrow I \cup X$ und $V \leftarrow V \setminus (X \cup N(X))$