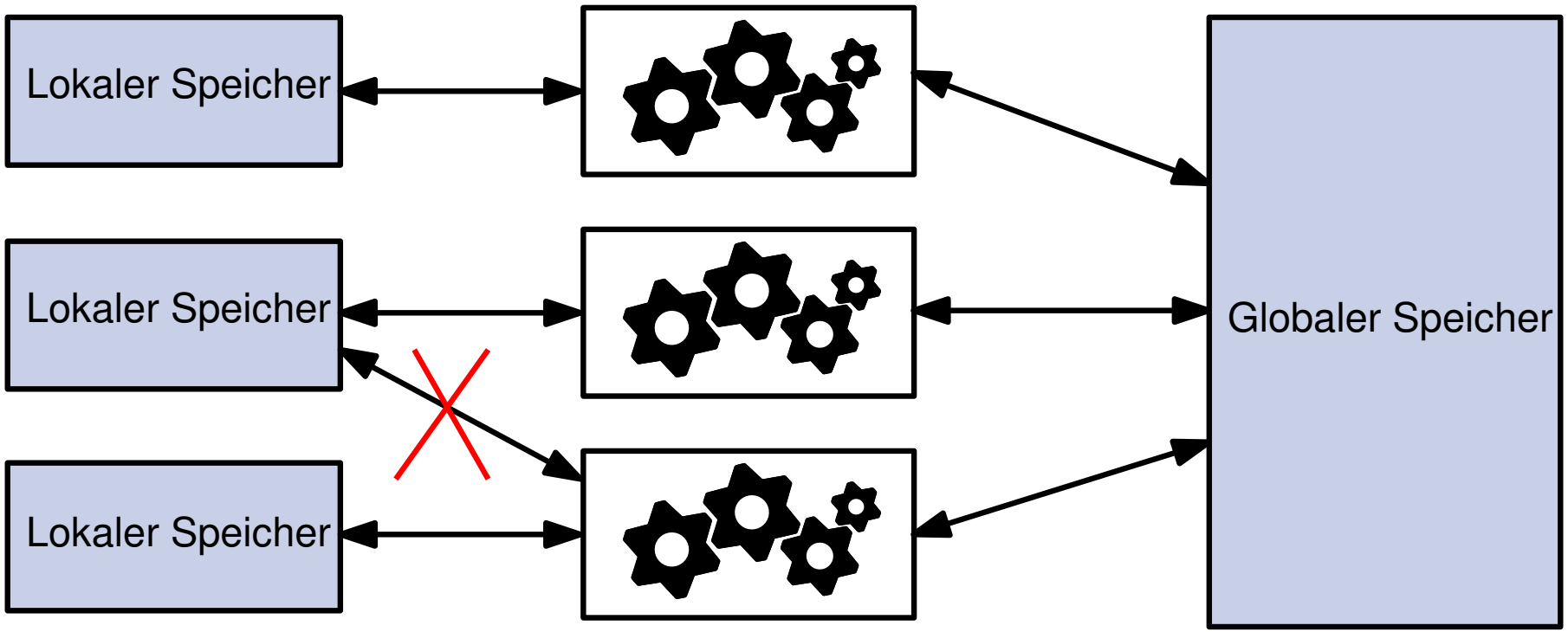
Karlsruher Institut für Technologie

Algorithmen II
Vorlesung am 29.01.2013

Parallele Algorithmen

INSTITUT FÜR THEORETISCHE INFORMATIK · PROF. DR. DOROTHEA WAGNER

PRAM Modell



- Unbegrenzte Prozessorenanzahl.
- Unbegrenzter globaler Speicher, auf den alle Prozessoren zugreifen können.
- Satz an Operationen, die jeder Prozessor einzeln ausführen kann.
- Jeder Prozessor hat einen eigenen lokalen unbegrenzten Speicher.

PRAM Modell

Problem: Was, wenn mehrere Prozessoren gleichzeitig die gleiche Speicherstelle im globalen Speicher lesen, bzw. beschreiben wollen?

gleichzeitiges Lesen erlaubt CR (*concurrent read*)	gleichzeitiges Lesen verboten ER (*exclusive read*)
gleichzeitiges Schreiben erlaubt CW (*concurrent write*)	gleichzeitiges Schreiben verboten EW (*exclusive write*)

Im folgenden Beschränkung auf CREW-PRAM-Modell.

Komplexität von parallelen Algorithmen

- Betrachte für Laufzeitanalyse immer den schlimmsten Fall.
- Berechnungsschritt = N Operationen, die gleichzeitig von N Prozessoren ausgeführt werden.

Definition 9.1 Die Laufzeit $T_{\mathcal{A}}(n)$ eines parallelen Algorithmus $\mathcal{A}$ ist

$$T_{\mathcal{A}}(n) := \max_{\substack{I \text{ Problembeispiel} \\ \text{der Größe } n}} \{\text{Anzahl der Berechnungsschritte von } \mathcal{A} \text{ bei Eingabe } I\}$$

Bemerkung:

1. Berechnung beginnt, wenn der erste Prozessor aktiv wird.
2. Berechnung endet, wenn der letzte Prozessor inaktiv geworden ist.
3. Prozessoren arbeiten synchron.

Definition 9.3 Die Prozessorenanzahl $P_{\mathcal{A}}(n)$ eines parallelen Algorithmus $\mathcal{A}$ ist

$$P_{\mathcal{A}}(n) := \max_{\substack{I \text{ Problembeispiel} \\ \text{der Größe } n}} \left\{ \begin{array}{l} \text{Anzahl an Prozessoren, die während des Ablaufs von } \mathcal{A} \\ \text{bei Eingabe } I \text{ gleichzeitig aktiv sind} \end{array} \right\}$$

Komplexität von parallelen Algorithmen

Vergleich von sequentiellen und parallelen Algorithmen für ein Problem:

$$\text{speed-up}(\mathcal{A}) := \frac{\begin{array}{c}\text{worst-case Laufzeit des schnellsten}\\ \text{bekannten sequentiellen Algorithmus}\end{array}}{\begin{array}{c}\text{worst-case Laufzeit des}\\ \text{parallelen Algorithmus } \mathcal{A}\end{array}}$$

Kombiniere Prozessorenanzahl und Laufzeit zu *Kosten* $C_{\mathcal{A}}$:

$$C_{\mathcal{A}}(n) := T_{\mathcal{A}}(n) \cdot P_{\mathcal{A}}(n)$$

kostenoptimal = asymptotisches Wachstum von $C_{\mathcal{A}}(n)$ und die schärfste asymptotische untere Schranke für die Laufzeit eines sequentiellen Algorithmus sind gleich.

Wenn untere Schranke nicht bekannt, so betrachte Effizienz:

$$E_{\mathcal{A}}(n) := \frac{\begin{array}{c}\text{worst-case Laufzeit des schnellsten}\\ \text{bekannten sequentiellen Algorithmus}\end{array}}{\text{Kosten des parallelen Algorithmus } \mathcal{A}}$$

Komplexitätsklassen

Definition 9.4: Nick's Class nach Nicholas Pippinger
Die Klasse $\mathcal{NC}$ ist die Klasse der Probleme, die durch einen parallelen Algorithmus $\mathcal{A}$ mit polylogarithmischer Laufzeit und polynomieller Prozessorenzahl gelöst werden können, d.h. $T_{\mathcal{A}}(n) \in \mathcal{O}((\log n)^{k_1})$ mit k_1 Konstante, und $P_{\mathcal{A}}(n) \in \mathcal{O}(n^{k_2})$ mit k_2 Konstante.

Definition 9.5: Steve's Class nach Stephan Cook
Die Klasse $\mathcal{SC}$ ist die Klasse der Probleme, die durch einen sequentiellen Algorithmus mit polylogarithmischem Speicherplatzbedarf und polynomieller Laufzeit gelöst werden können.

Berechnung von Summen

Summe($a_1, \ldots, a_n$)
Eingabe: n Werte $a_1, \ldots, a_n$, o.B.d.A. sei $n = 2^m$.
Ausgabe: $\sum_{i=1}^{n} a_i$

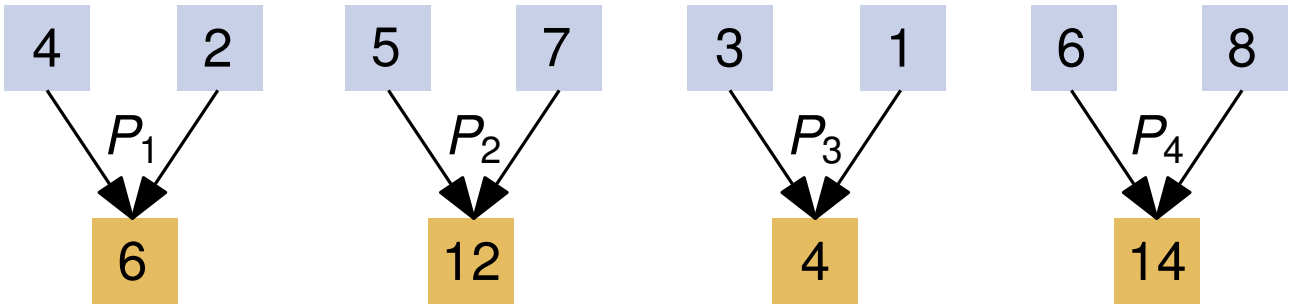
für $i = 1$ **bis** m **tue**
    Für alle j : $1 \leq j \leq \frac{n}{2^i}$ **führe parallel aus**
        Prozessor P_j berechnet $a_j := a_{2j-1} + a_{2j}$.
Gib a_1 aus.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Berechnung von Summen

Summe($a_1, \ldots, a_n$)
Eingabe: n Werte $a_1, \ldots, a_n$, o.B.d.A. sei $n = 2^m$.
Ausgabe: $\sum_{i=1}^{n} a_i$

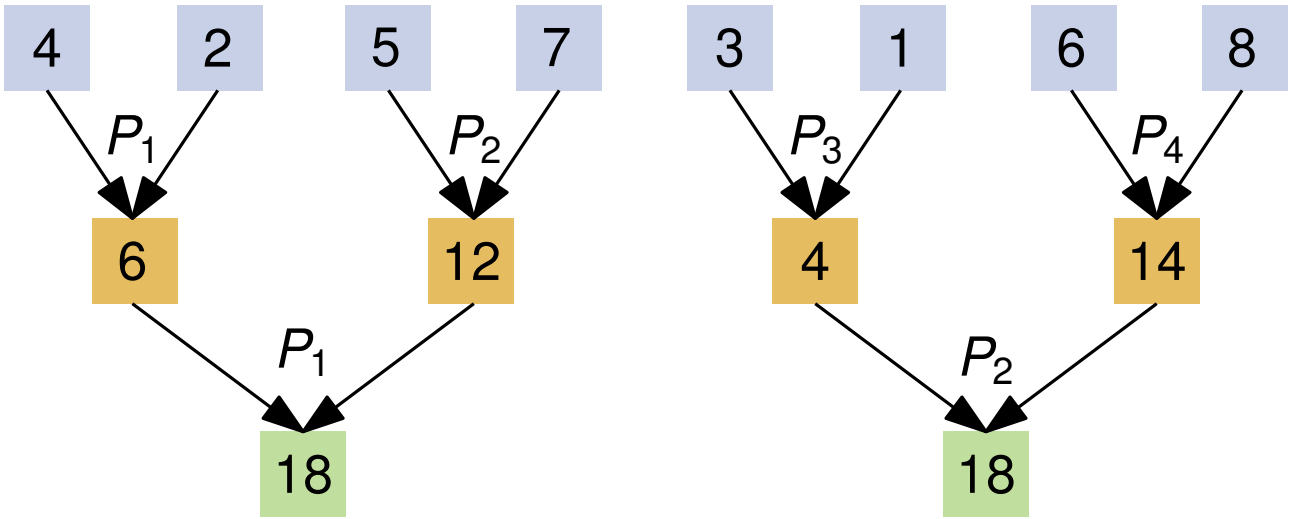
für $i = 1$ **bis** m **tue**
    Für alle j : $1 \leq j \leq \frac{n}{2^i}$ **führe parallel aus**
        Prozessor P_j berechnet $a_j := a_{2j-1} + a_{2j}$.
Gib a_1 aus.



Berechnung von Summen

Summe($a_1, \ldots, a_n$)
Eingabe: n Werte $a_1, \ldots, a_n$, o.B.d.A. sei $n = 2^m$.
Ausgabe: $\sum_{i=1}^{n} a_i$

für $i = 1$ **bis** m **tue**
    Für alle j : $1 \leq j \leq \frac{n}{2^i}$ **führe parallel aus**
        Prozessor P_j berechnet $a_j := a_{2j-1} + a_{2j}$.
Gib a_1 aus.

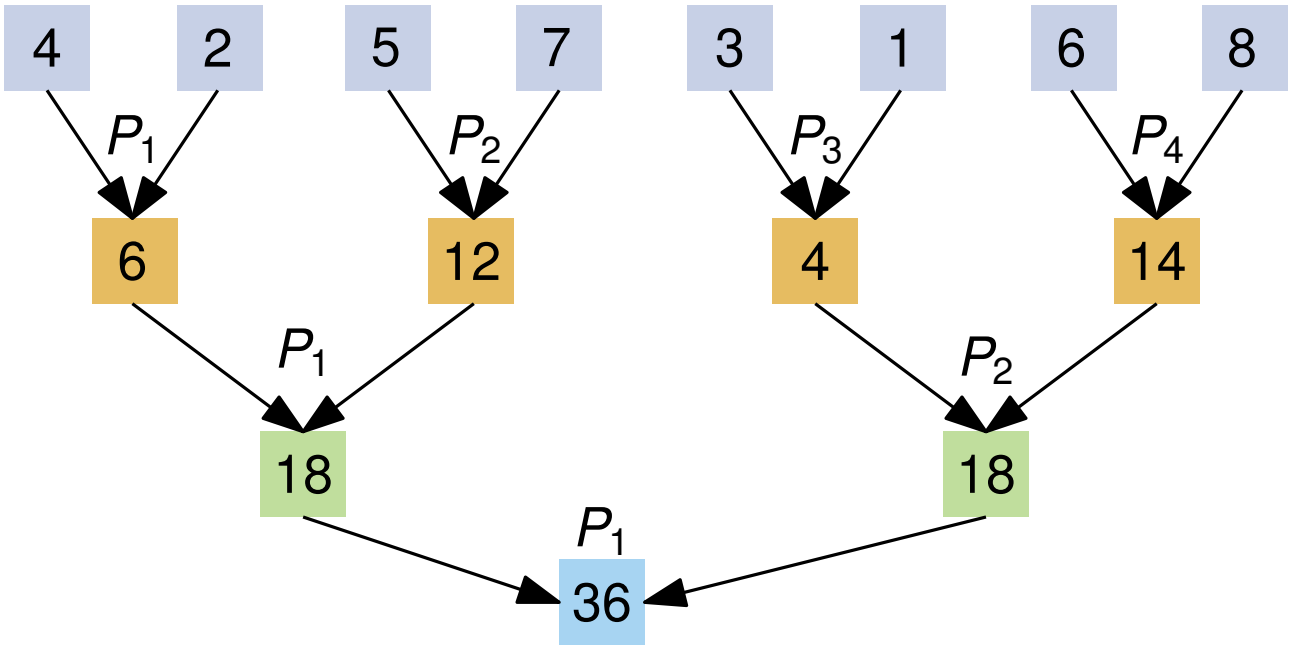


Berechnung von Summen

Summe$(a_1, \ldots, a_n)$
Eingabe: n Werte $a_1, \ldots, a_n$, o.B.d.A. sei $n = 2^m$.
Ausgabe: $\sum_{i=1}^{n} a_i$

für $i = 1$ **bis** m **tue**
  Für alle j : $1 \leq j \leq \frac{n}{2^i}$ **führe parallel aus**
    Prozessor P_j berechnet $a_j := a_{2j-1} + a_{2j}$.
Gib a_1 aus.

- $P_{\mathcal{A}}(n) = \frac{n}{2}$, $T_{\mathcal{A}}(n) = \mathcal{O}(\log n)$
- Somit gilt: $C_{\mathcal{A}}(n) \in \mathcal{O}(n \log n)$
- **$\mathcal{A}$ ist nicht kostenoptimal.**



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Berechnung von Summen

Karlsruher Institut für Technologie

$\texttt{Summe}(a_1, \ldots, a_n)$
Eingabe: n Werte $a_1, \ldots, a_n$, o.B.d.A. sei $n = 2^m$.
Ausgabe: $\sum_{i=1}^{n} a_i$

für $i = 1$ **bis** m **tue**
  Für alle j : $1 \leq j \leq \frac{n}{2^i}$ **führe parallel aus**
    Prozessor P_j berechnet $a_j := a_{2j-1} + a_{2j}$.
Gib a_1 aus.

Verbesserung:

- Verwende $\lceil \frac{n}{\log n} \rceil$ anstatt n Prozessoren.
- Die $\frac{n}{2}$ Operationen aus dem ersten Schritt werden nun von $\lceil \frac{n}{\log n} \rceil$ Prozessoren in höchstens $\frac{\log n}{2}$ parallelen Berechnungsschritten ausgeführt.
- Im i-ten Schritt werden $\frac{n}{2^i}$ Operationen in höchstens $\frac{\log n}{2^i}$ Schritten ausgeführt.
- Es ergibt sich:

$$T_{\mathcal{A}}(n) \leq c \cdot \sum_{i=1}^{\log n} \frac{\log n}{2^i} = c \cdot \log n \underbrace{\sum_{i=1}^{\log n} \frac{1}{2^i}}_{\leq 1} \in \mathcal{O}(\log n).$$

- Wegen $P_{\mathcal{A}}(n) = \lceil \frac{n}{\log n} \rceil$ gilt $C_{\mathcal{A}} \in O(n)$.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Berechnung von Summen

Karlsruher Institut für Technologie

$\texttt{Summe}(a_1, \ldots, a_n)$
Eingabe: n Werte $a_1, \ldots, a_n$, o.B.d.A. sei $n = 2^m$.
Ausgabe: $\sum_{i=1}^{n} a_i$

für $i = 1$ **bis** m **tue**
    Für alle $j: \ 1 \leq j \leq \frac{n}{2^i}$ **führe parallel aus**
        Prozessor P_j berechnet $a_j := a_{2j-1} + a_{2j}$.
Gib a_1 aus.

Bemerkung: CREW-PRAM (oder ähnliche Modelle) und ein Modell, bei dem zu Beginn nur ein Prozessor aktiv ist und alle anderen erst „aufgeweckt“ werden, unterscheiden sich in der Laufzeit im wesentlichen nur um einen Faktor log N (bei N Prozessoren, die gleichzeitig lesen wollen, bzw. aktiviert werden sollen).

Das selbe Verfahren funktioniert auch auf anderen binären Operationen, die n-fach angewendet werden. Beispiele: Minimum, Maximum, logisches Oder.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Präfixsumme

Problemstellung
gegeben: n Werte $a_1, \ldots, a_n$.

gesucht: Präfixsumme $A_k = \sum_{i=1}^{k} a_i$ für jedes k mit $1 \leq k \leq n$.

Aus technischen Gründen: Indizierung mit $a_n, a_{n+1}, \ldots, a_{2n-1}$ und o.B.d.A. $n = 2^m$ für $m \in \mathbb{N}$.

Verfahren besteht aus zwei Phasen:

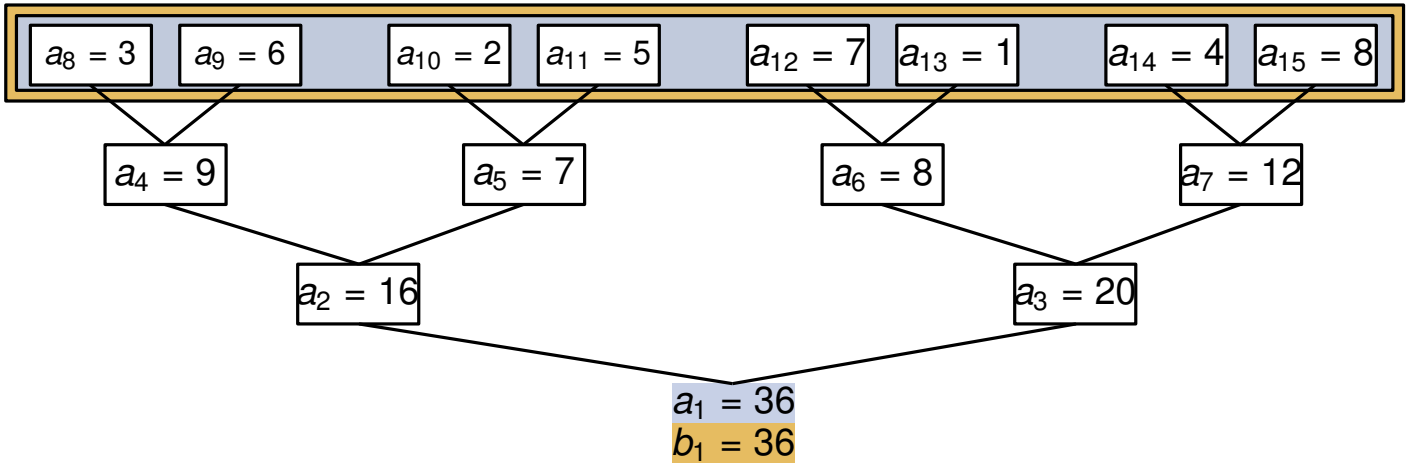
1. Phase: Berechne $S :=$ SUMME$(a_n, \ldots, a_{2n-1})$.

2. Phase: Berechne aus S und den Zwischenergebnissen von SUMME$(a_n, \ldots, a_{2n-1})$ mithilfe von gegeigneten Differenzen die Präfixsummen.

Präfixsumme

1. Phase: Berechne $S := \text{SUMME}(a_n, \ldots, a_{2n-1})$.

für $j = m - 1$ **bis** 0 **tue**
  Für alle $i : 2^j \leq i \leq 2^{j+1} - 1$ **führe parallel aus**
    $a_i := a_{2i} + a_{2i+1}$
  $b_1 := a_1$

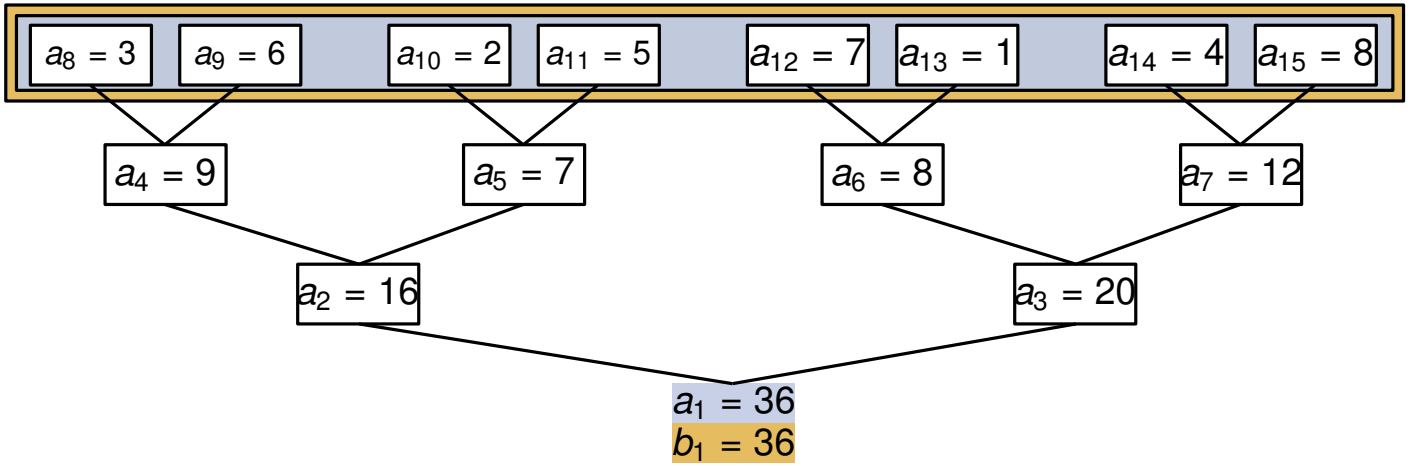


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Präfixsumme

2. Phase: Berechne aus S und den Zwischenergebnissen von SUMME$(a_n, \ldots, a_{2n-1})$ mithilfe von gegeigneten Differenzen die Präfixsummen.

für $j = 1$ **bis** m **tue**
Für alle $i : 2^j \leq i \leq 2^{j+1} - 1$ **führe parallel aus**
wenn i *ungerade* **dann** $b_i := b_{\frac{i-1}{2}}$ **sonst** $b_i := b_{\frac{i}{2}} - a_{i+1}$

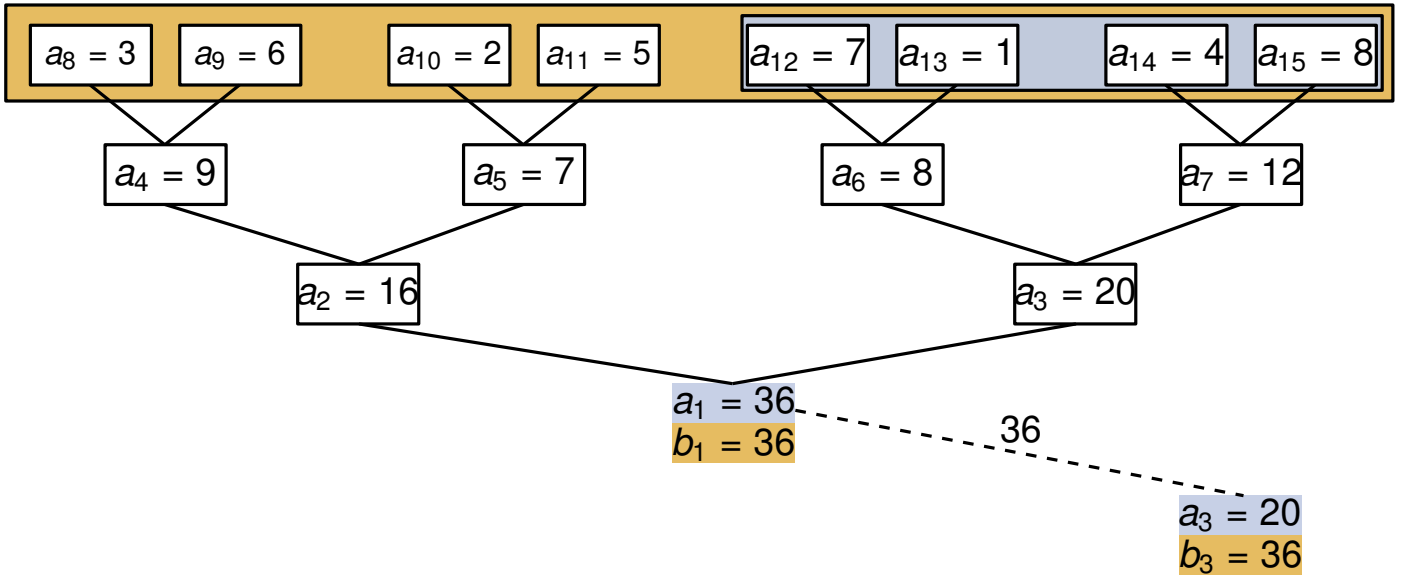


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Präfixsumme

2. Phase: Berechne aus S und den Zwischenergebnissen von SUMME$(a_n, \ldots, a_{2n-1})$ mithilfe von gegeigneten Differenzen die Präfixsummen.

für $j = 1$ **bis** m **tue**
Für alle $i : 2^j \leq i \leq 2^{j+1} - 1$ **führe parallel aus**
wenn i *ungerade* **dann** $b_i := b_{\frac{i-1}{2}}$ **sonst** $b_i := b_{\frac{i}{2}} - a_{i+1}$

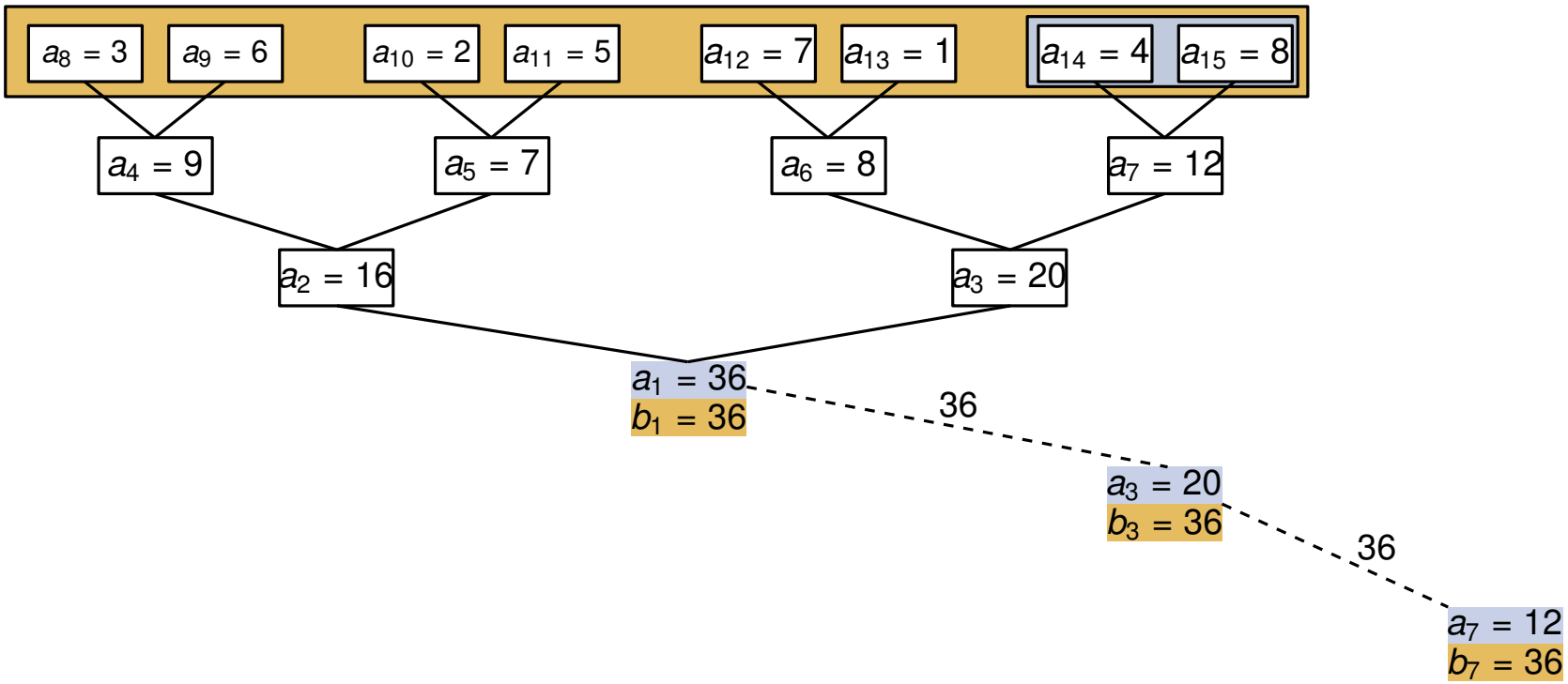


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Präfixsumme

2. Phase: Berechne aus S und den Zwischenergebnissen von SUMME$(a_n, \ldots, a_{2n-1})$ mithilfe von gegeigneten Differenzen die Präfixsummen.

für $j = 1$ **bis** m **tue**
Für alle $i : 2^j \leq i \leq 2^{j+1} - 1$ **führe parallel aus**
wenn i *ungerade* **dann** $b_i := b_{\frac{i-1}{2}}$ **sonst** $b_i := b_{\frac{i}{2}} - a_{i+1}$

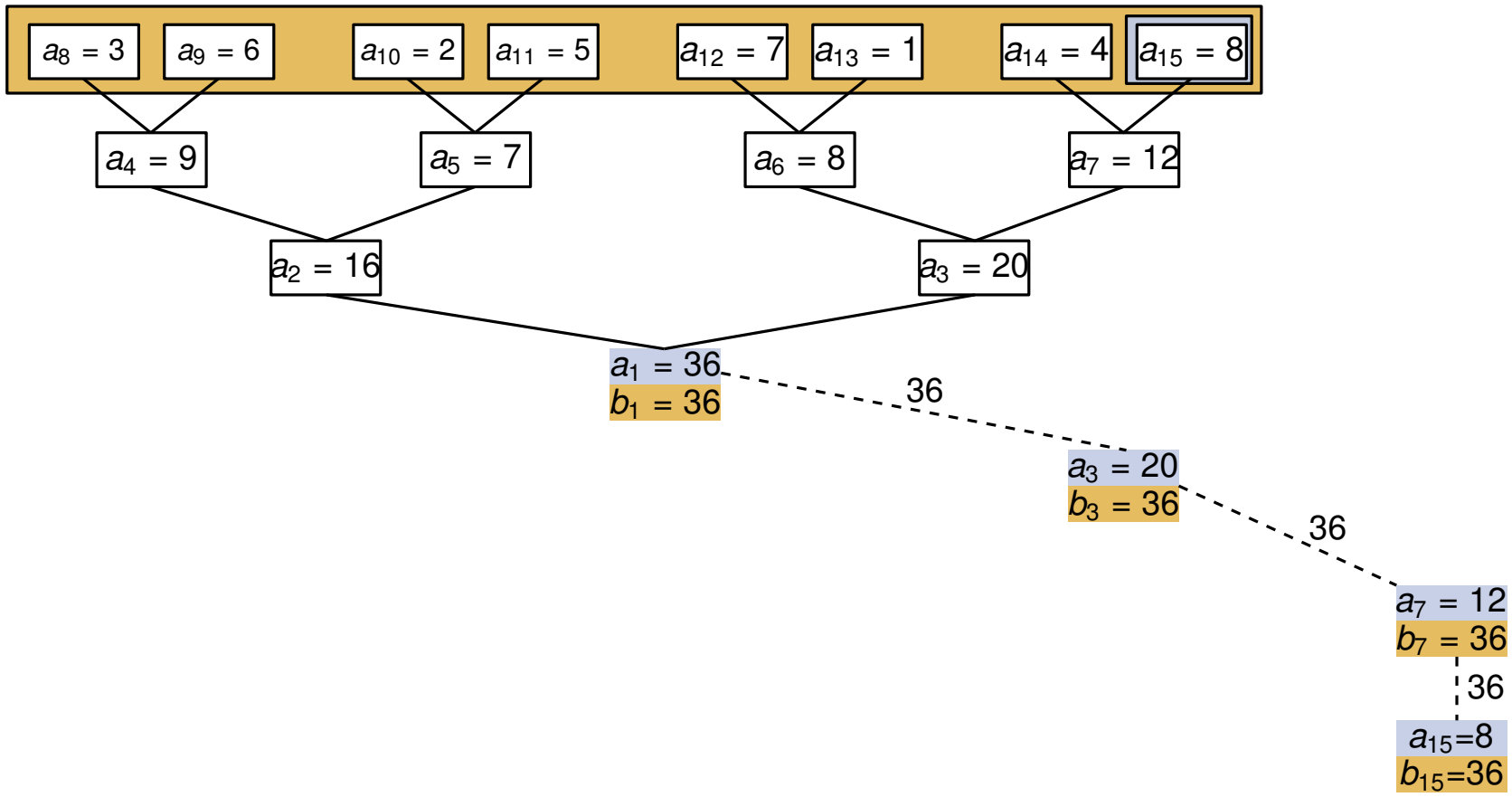


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Präfixsumme

2. Phase: Berechne aus S und den Zwischenergebnissen von SUMME$(a_n, \ldots, a_{2n-1})$ mithilfe von gegeigneten Differenzen die Präfixsummen.

für $j = 1$ **bis** m **tue**
Für alle $i : 2^j \leq i \leq 2^{j+1} - 1$ **führe parallel aus**
wenn i *ungerade* **dann** $b_i := b_{\frac{i-1}{2}}$ **sonst** $b_i := b_{\frac{i}{2}} - a_{i+1}$

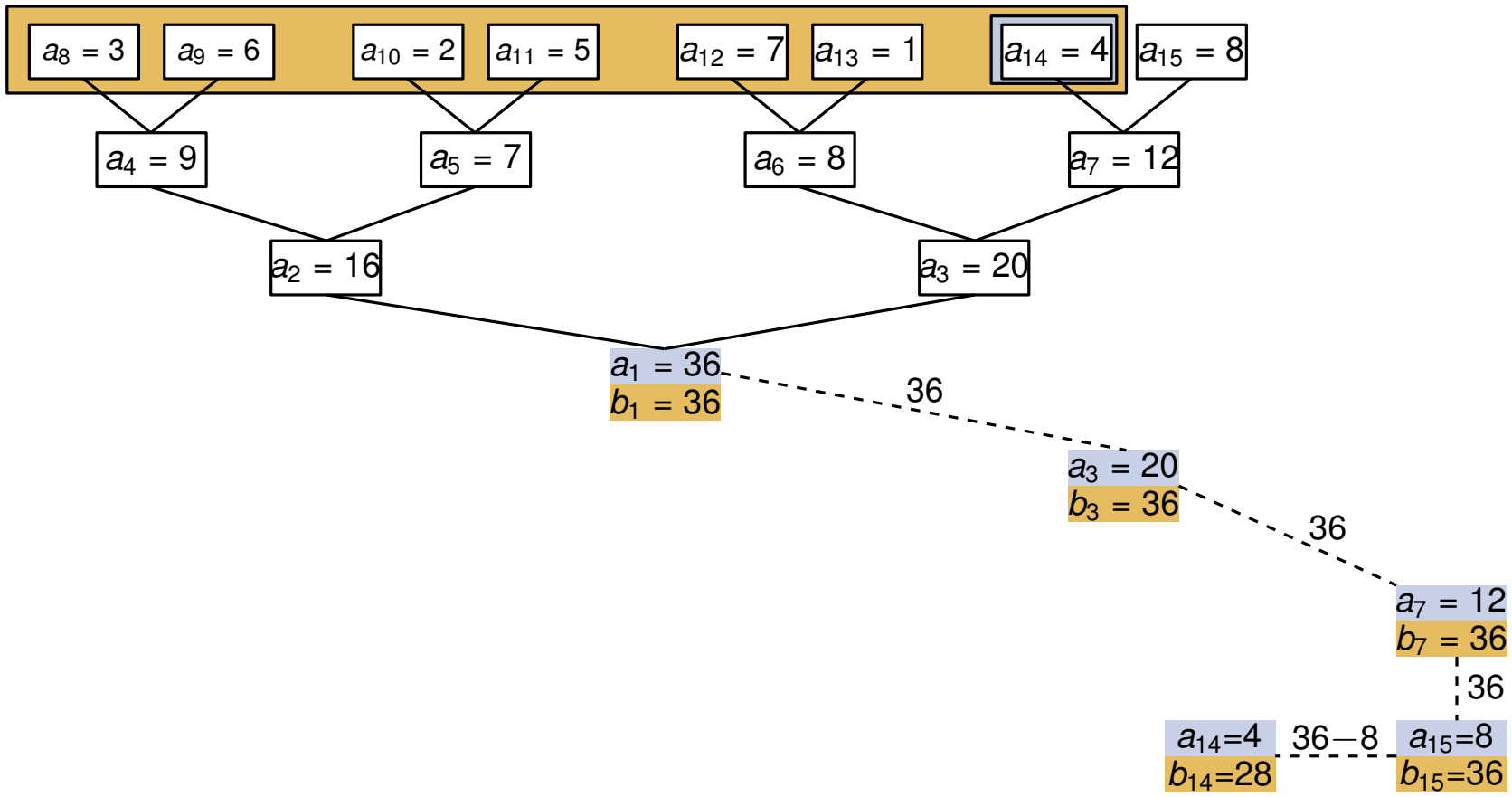


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Präfixsumme

2. Phase: Berechne aus S und den Zwischenergebnissen von SUMME$(a_n, \ldots, a_{2n-1})$ mithilfe von gegeigneten Differenzen die Präfixsummen.

für $j = 1$ **bis** m **tue**
Für alle $i : 2^j \leq i \leq 2^{j+1} - 1$ **führe parallel aus**
wenn i *ungerade* **dann** $b_i := b_{\frac{i-1}{2}}$ **sonst** $b_i := b_{\frac{i}{2}} - a_{i+1}$

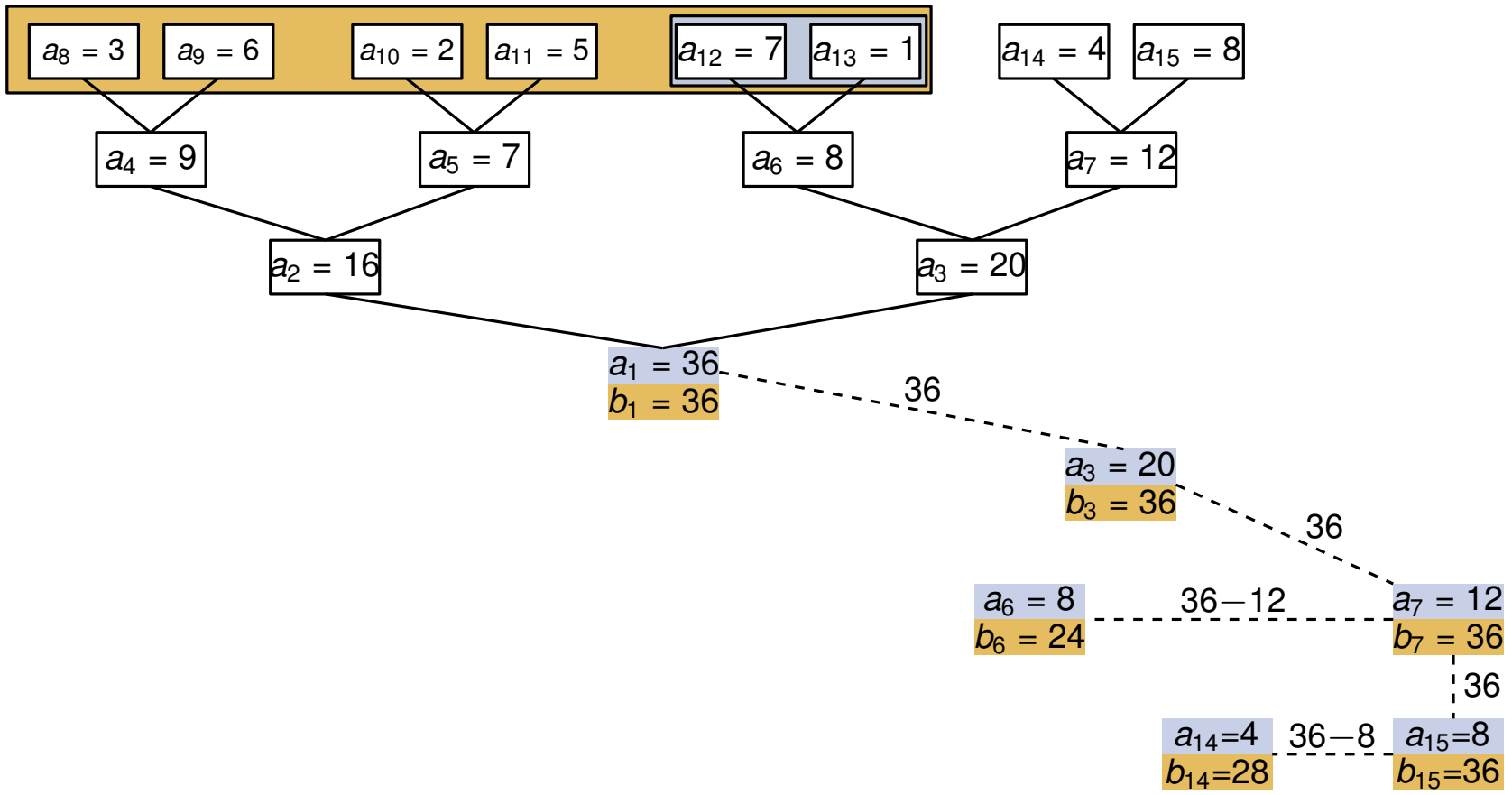


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Präfixsumme

2. Phase: Berechne aus S und den Zwischenergebnissen von SUMME$(a_n, \ldots, a_{2n-1})$ mithilfe von gegeigneten Differenzen die Präfixsummen.

für $j = 1$ **bis** m **tue**
Für alle $i : 2^j \leq i \leq 2^{j+1} - 1$ **führe parallel aus**
wenn i *ungerade* **dann** $b_i := b_{\frac{i-1}{2}}$ **sonst** $b_i := b_{\frac{i}{2}} - a_{i+1}$

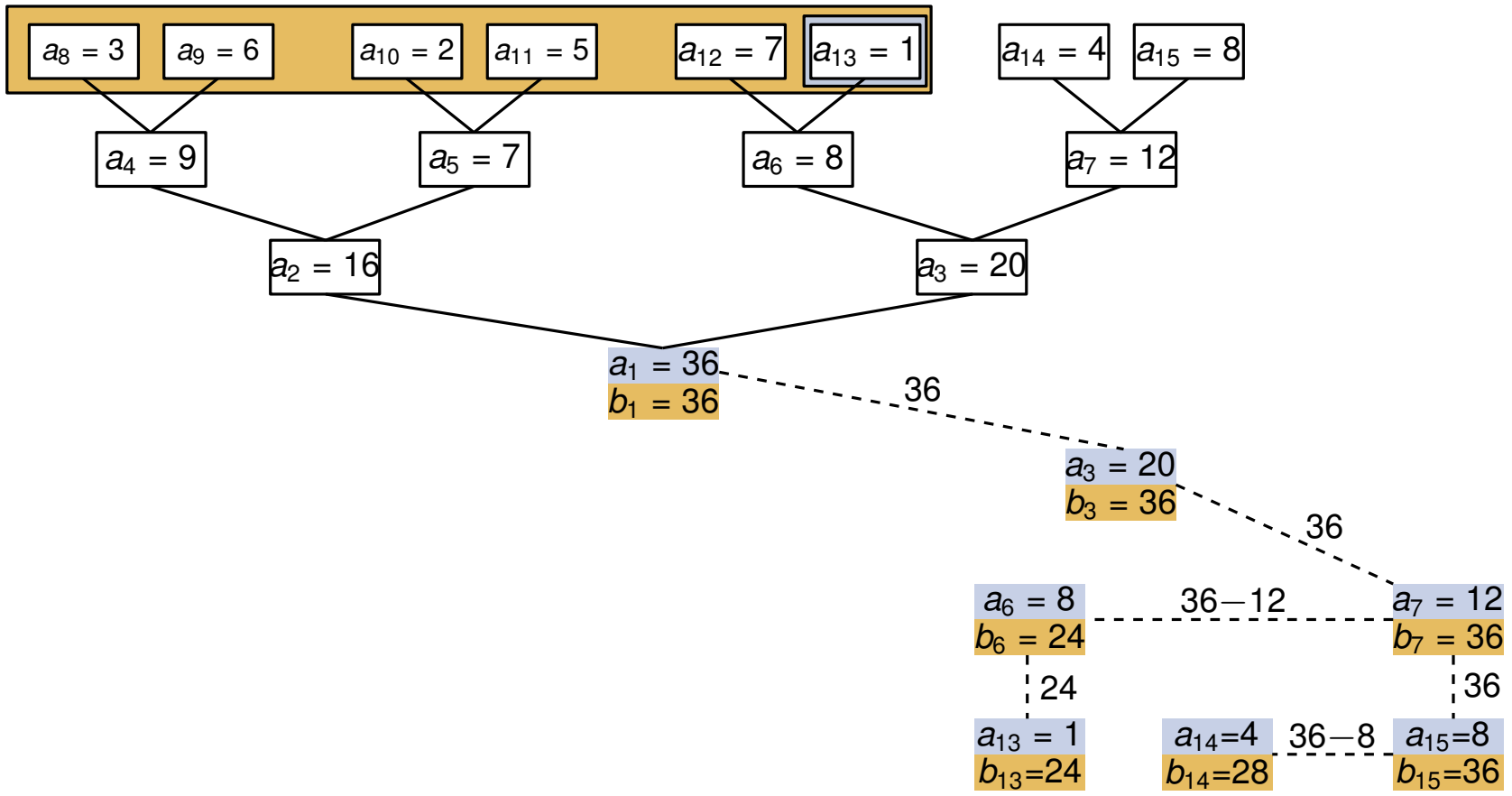


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Präfixsumme

2. Phase: Berechne aus S und den Zwischenergebnissen von SUMME$(a_n, \ldots, a_{2n-1})$ mithilfe von gegeigneten Differenzen die Präfixsummen.

für $j = 1$ **bis** m **tue**
Für alle $i : 2^j \leq i \leq 2^{j+1} - 1$ **führe parallel aus**
wenn i *ungerade* **dann** $b_i := b_{\frac{i-1}{2}}$ **sonst** $b_i := b_{\frac{i}{2}} - a_{i+1}$

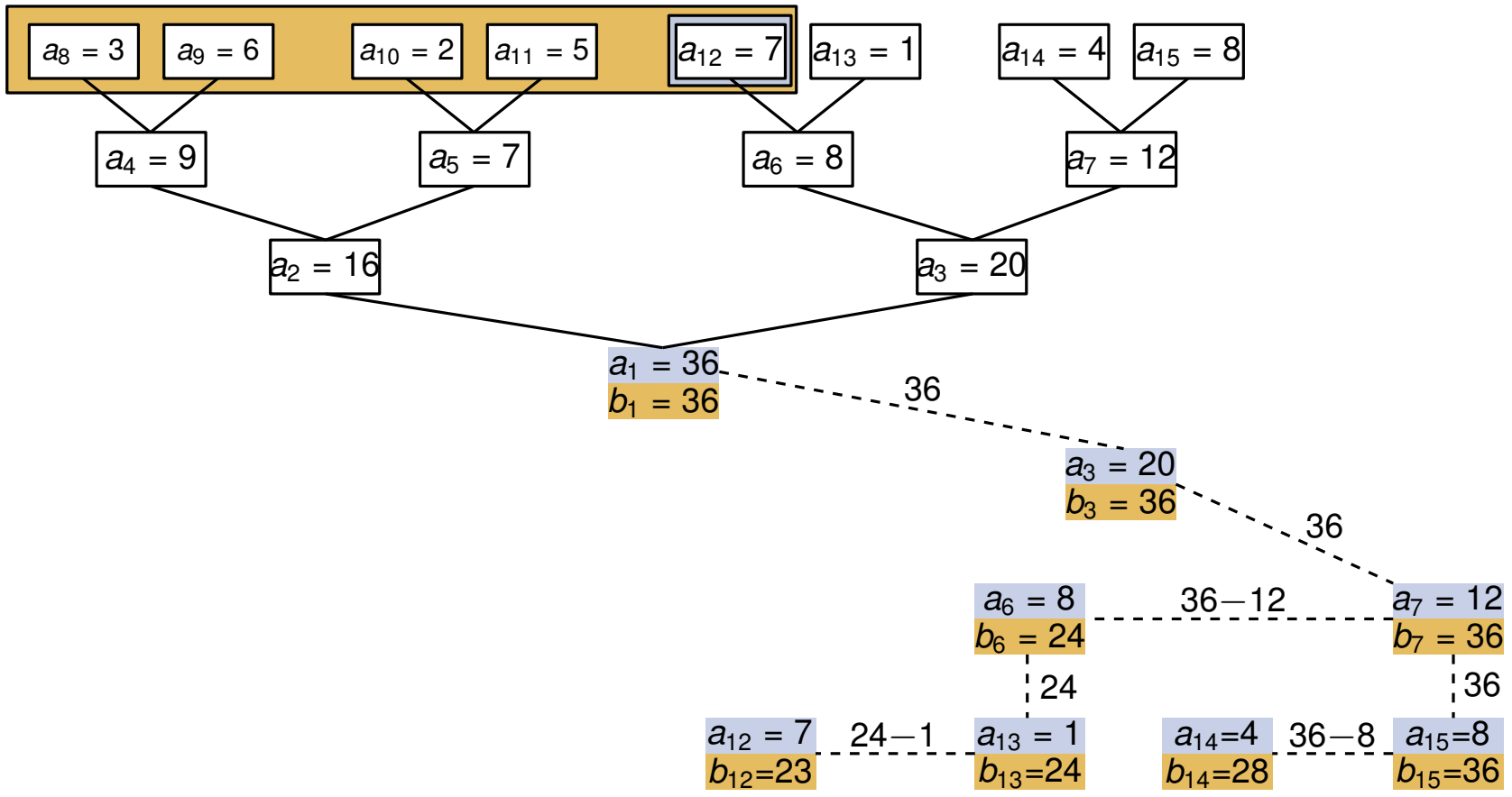


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Präfixsumme

2. Phase: Berechne aus S und den Zwischenergebnissen von SUMME$(a_n, \ldots, a_{2n-1})$ mithilfe von gegeigneten Differenzen die Präfixsummen.

für $j = 1$ **bis** m **tue**
Für alle $i : 2^j \leq i \leq 2^{j+1} - 1$ **führe parallel aus**
wenn i *ungerade* **dann** $b_i := b_{\frac{i-1}{2}}$ **sonst** $b_i := b_{\frac{i}{2}} - a_{i+1}$

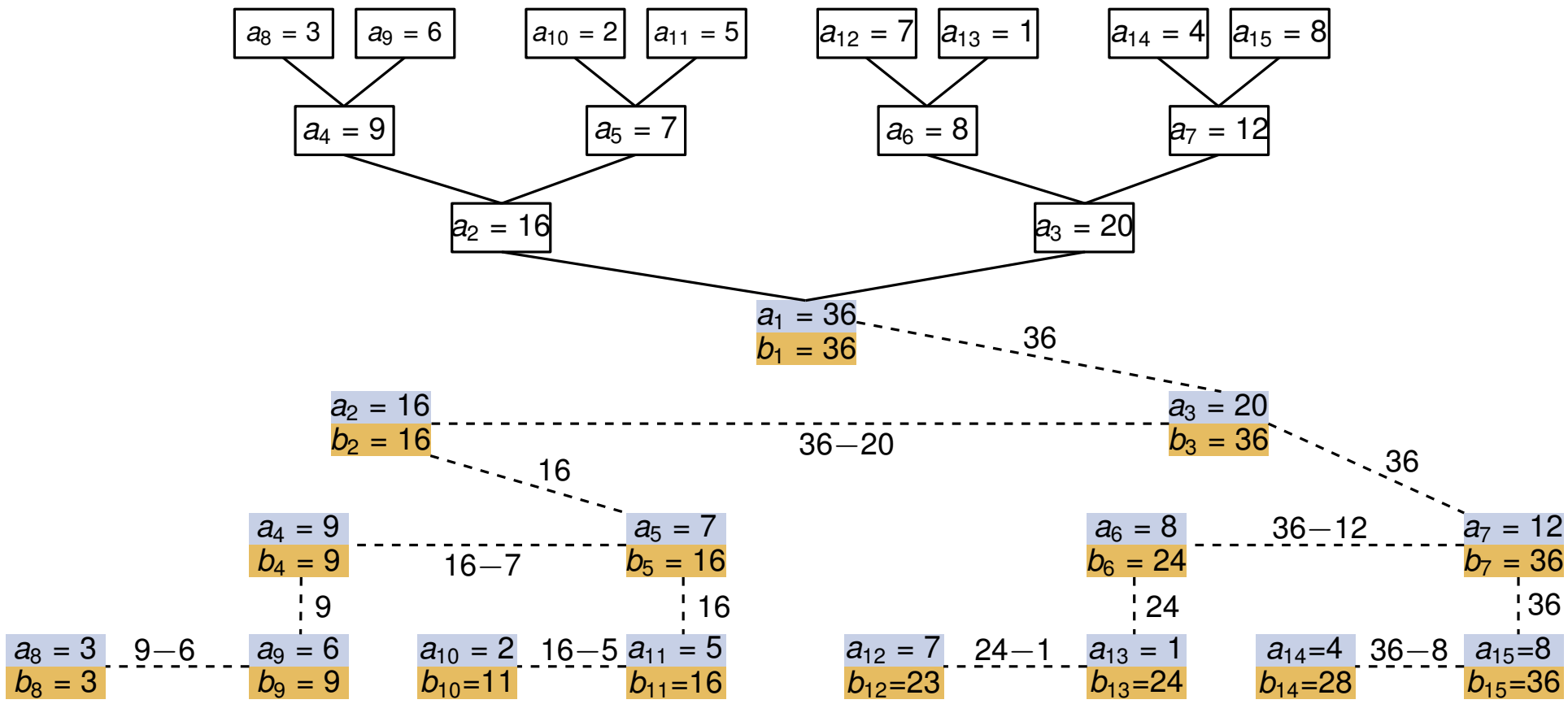


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Präfixsumme

2. Phase: Berechne aus S und den Zwischenergebnissen von SUMME$(a_n, \ldots, a_{2n-1})$ mithilfe von gegeigneten Differenzen die Präfixsummen.

für $j = 1$ **bis** m **tue**
Für alle $i : 2^j \leq i \leq 2^{j+1} - 1$ **führe parallel aus**
wenn i *ungerade* **dann** $b_i := b_{\frac{i-1}{2}}$ **sonst** $b_i := b_{\frac{i}{2}} - a_{i+1}$

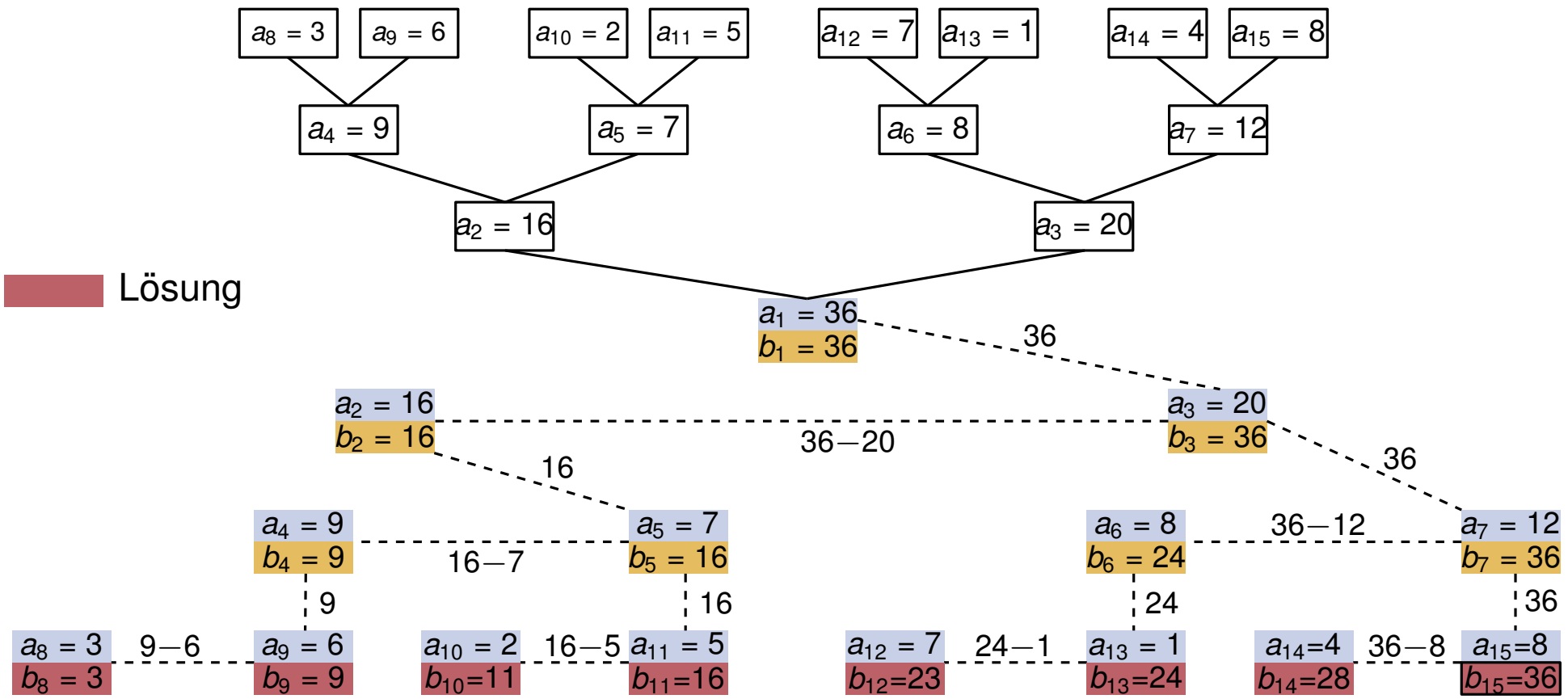


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Präfixsumme

2. Phase: Berechne aus S und den Zwischenergebnissen von SUMME$(a_n, \ldots, a_{2n-1})$ mithilfe von gegeigneten Differenzen die Präfixsummen.

für $j = 1$ **bis** m **tue**
Für alle $i : 2^j \leq i \leq 2^{j+1} - 1$ **führe parallel aus**
wenn i *ungerade* **dann** $b_i := b_{\frac{i-1}{2}}$ **sonst** $b_i := b_{\frac{i}{2}} - a_{i+1}$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Präfixsumme

1. Phase: Berechne $S := \text{SUMME}(a_n, \ldots, a_{2n-1})$.

für $j = m - 1$ **bis** 0 **tue**
  Für alle $i : 2^j \leq i \leq 2^{j+1} - 1$ **führe parallel aus**
    $a_i := a_{2i} + a_{2i+1}$
  $b_1 := a_1$

2. Phase: Berechne aus S und den Zwischenergebnissen von $\text{SUMME}(a_n, \ldots, a_{2n-1})$ mithilfe von gegeigneten Differenzen die Präfixsummen.

für $j = 1$ **bis** m **tue**
  Für alle $i : 2^j \leq i \leq 2^{j+1} - 1$ **führe parallel aus**
    wenn i *ungerade* **dann** $b_i := b_{\frac{i-1}{2}}$ **sonst** $b_i := b_{\frac{i}{2}} - a_{i+1}$

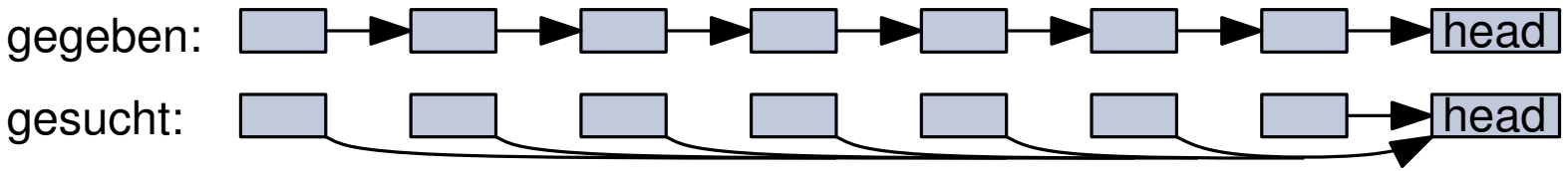
Komplexität:

- Laufzeit liegt in $O(\log n)$.
- Anzahl Prozessoren liegt in $O(n)$, da maximal $\frac{n}{2}$ Prozessoren gleichzeitig aktiv sind.
- Durch Rescheudling kann Prozessorenanzahl auf $O(\frac{n}{\log n})$ bei Laufzeit $O(\log n)$ reduziert werden.

$\Rightarrow$ Algorithmus ist kostenoptimal.

List Ranking

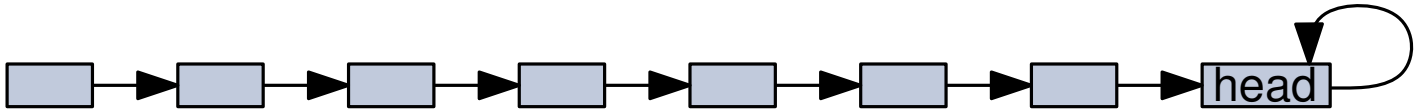
Problemstellung
gegeben: Einfach verkettete Liste L.
gesucht: Jedes Element in Liste soll auf den Kopf von L zeigen.



Vorbedingung: Array NEXT[] enthält Nachfolger NEXT[i] für jedes Element $1 \leq i \leq n$ (NEXT[head]=head).

Nachbedingung: Array NEXT[] enthält *head* für jedes Element $1 \leq i \leq n$.

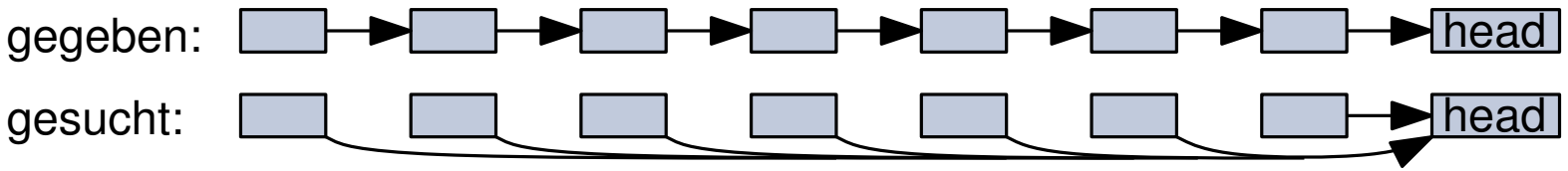
für $j = 1$ **bis** $\lceil \log n \rceil$ **tue**
  Für alle i : $1 \leq i \leq n$ **führe parallel aus**
    NEXT[i] := NEXT[NEXT[i]]



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

List Ranking

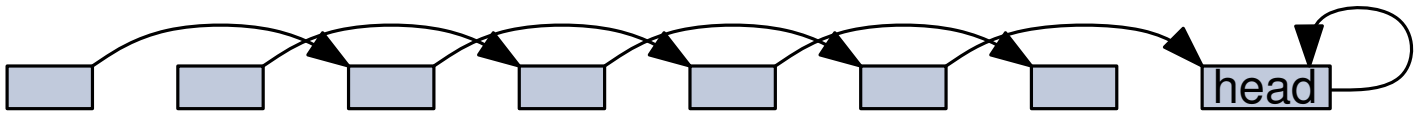
Problemstellung
gegeben: Einfach verkettete Liste L.
gesucht: Jedes Element in Liste soll auf den Kopf von L zeigen.



Vorbedingung: Array NEXT[] enthält Nachfolger NEXT[i] für jedes Element $1 \leq i \leq n$ (NEXT[head]=head).

Nachbedingung: Array NEXT[] enthält *head* für jedes Element $1 \leq i \leq n$.

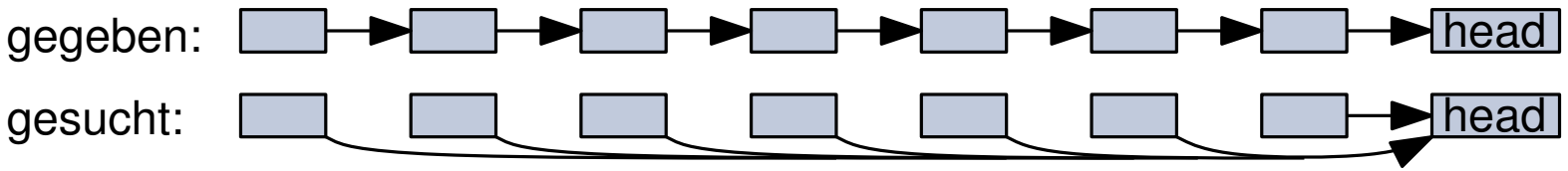
für $j = 1$ **bis** $\lceil \log n \rceil$ **tue**
  Für alle i : $1 \leq i \leq n$ **führe parallel aus**
    NEXT[i] := NEXT[NEXT[i]]



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

List Ranking

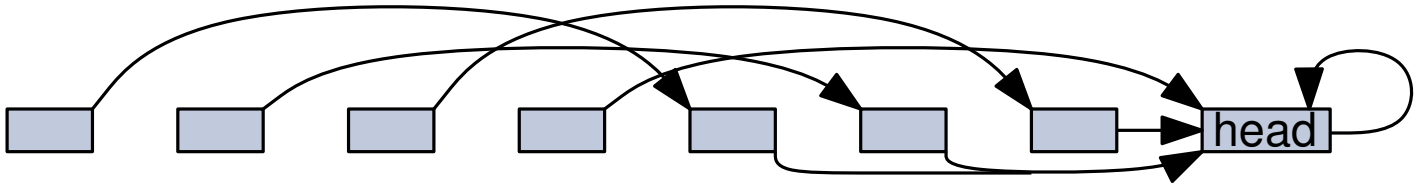
Problemstellung
gegeben: Einfach verkettete Liste L.
gesucht: Jedes Element in Liste soll auf den Kopf von L zeigen.



Vorbedingung: Array NEXT[] enthält Nachfolger NEXT[i] für jedes Element $1 \leq i \leq n$ (NEXT[head]=head).

Nachbedingung: Array NEXT[] enthält *head* für jedes Element $1 \leq i \leq n$.

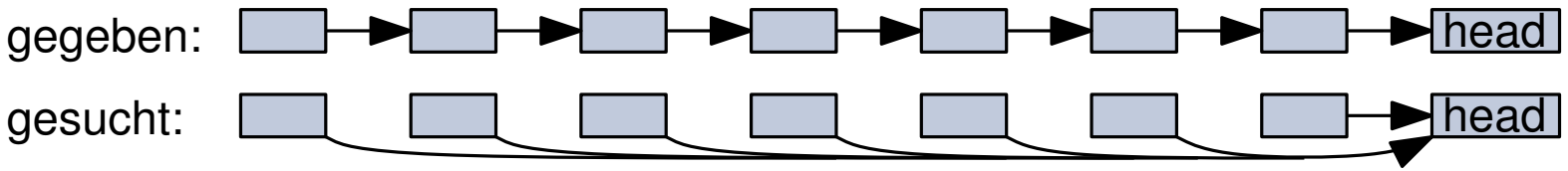
für $j = 1$ **bis** $\lceil \log n \rceil$ **tue**
  Für alle i : $1 \leq i \leq n$ **führe parallel aus**
    NEXT[i] := NEXT[NEXT[i]]



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

List Ranking

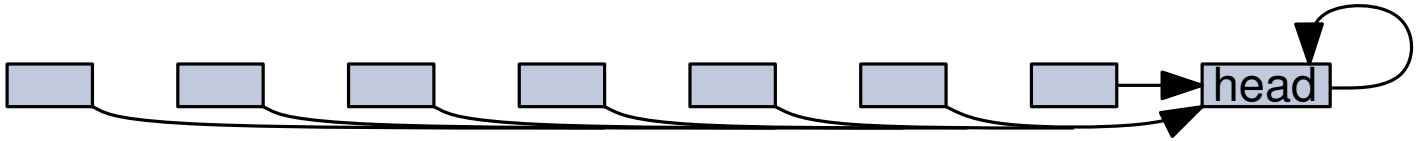
Problemstellung
gegeben: Einfach verkettete Liste L.
gesucht: Jedes Element in Liste soll auf den Kopf von L zeigen.



Vorbedingung: Array NEXT[] enthält Nachfolger NEXT[i] für jedes Element $1 \leq i \leq n$ (NEXT[head]=head).

Nachbedingung: Array NEXT[] enthält *head* für jedes Element $1 \leq i \leq n$.

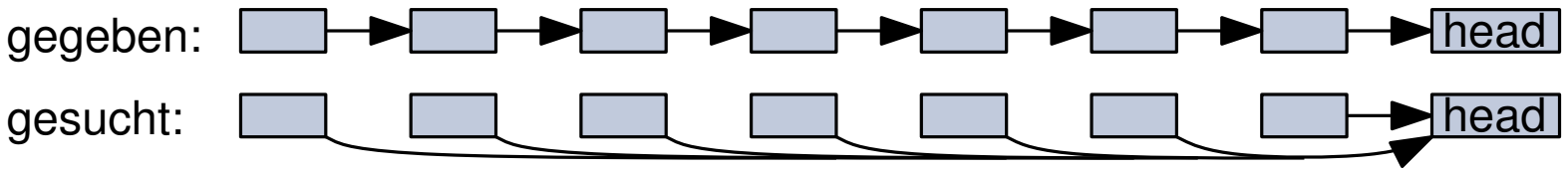
für $j = 1$ **bis** $\lceil \log n \rceil$ **tue**
  Für alle i : $1 \leq i \leq n$ **führe parallel aus**
    NEXT[i] := NEXT[NEXT[i]]



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

List Ranking

Problemstellung
gegeben: Einfach verkettete Liste L.
gesucht: Jedes Element in Liste soll auf den Kopf von L zeigen.



Vorbedingung: Array NEXT[] enthält Nachfolger NEXT[i] für jedes Element $1 \leq i \leq n$ (NEXT[head]=head).

Nachbedingung: Array NEXT[] enthält *head* für jedes Element $1 \leq i \leq n$.

für $j = 1$ **bis** $\lceil \log n \rceil$ **tue**
  Für alle i : $1 \leq i \leq n$ **führe parallel aus**
    NEXT[i] := NEXT[NEXT[i]]

- LISTRANKING kann in $O(\log n)$ Zeit mithilfe von n Prozessoren gelöst werden.
- Verfahren funktioniert auch auf Wurzelbäumen, in denen jeder Knoten nur seinen direkten Vorgänger kennt: $O(\log h)$ Laufzeit und n Prozessoren, wobei h die Baumhöhe und n die Anzahl der Knoten bezeichnet.

Binäroperationen einer partitionierten Menge

Problem: *Binäroperationen auf einer partitionierten Menge (BPM)*
Gegeben n Werte, die in p Gruppen aufgeteilt sind. Bestimme die p Werte, die sich als Binärverbindungen (Summe, Minimum, logisches Oder, etc.) der Werte der p Gruppen ergeben.

Beispiel:

Eingabe:	1 + 28 + 12	4 + 12 + 7 + 8 + 9 + 10	45 + 1 + 3 + 2
Ausgabe:	41	50	51

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Binäroperationen einer partitionierten Menge

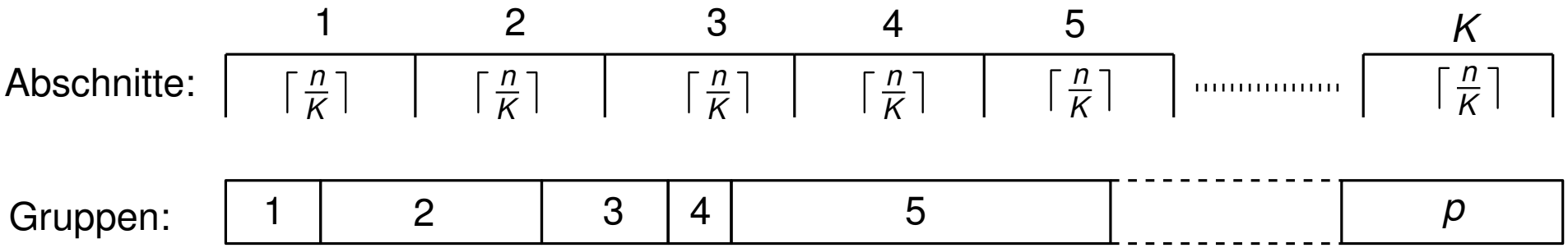
Problem: *Binäroperationen auf einer partitionierten Menge (BPM)*
Gegeben n Werte, die in p Gruppen aufgeteilt sind. Bestimme die p Werte, die sich als Binärverbindungen (Summe, Minimum, logisches Oder, etc.) der Werte der p Gruppen ergeben.

Lemma: BPM kann mithilfe von K Prozessoren in Laufzeit

$$t(n) \in \begin{cases} O(\lceil \frac{n}{K} \rceil + \log K), & \text{falls } n > K \\ O(\log n), & \text{sonst.} \end{cases}$$

gelöst werden.

Skizze:



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Zusammenhangskomponenten

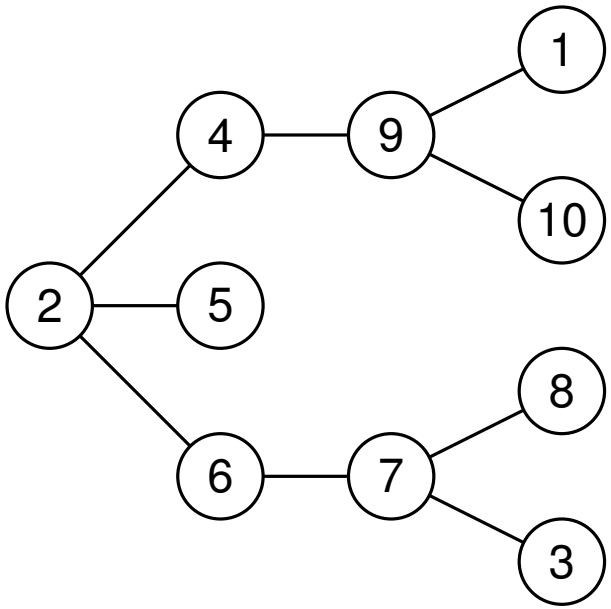
Problemstellung
gegeben: Graph $G = (V, E)$, $V = \{1, \ldots, n\}$
gesucht: Zusammenhangskomponenten von G.

Datenstrukturen:

- Array K der Länge n: K[i] = Index der aktuellen Komponente in der Knoten i liegt.

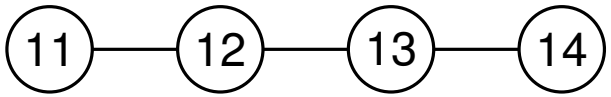
  Also: Für alle Knoten $i_1, \ldots, i_\ell$ der selben Komponente gilt: $K[i_1] = \ldots = K[i_\ell] = k$. Insbesondere wird gelten: $k = \min\{i_1, \ldots, i_\ell\}$. Knoten k ist *Repräsentant der Komponente*.

- Array N der Länge n: N[i] = Index der aktuellen Zusammenhangskomponente kleinsten Index, in der i oder ein Nachbar von i liegt.



Ablauf:

1. Initialisierung: Alle Knoten liegen in eigenen Komponenten.
2. Vereinige zusammenhängende Komponenten in $\lceil \log n \rceil$ Phasen mit jeweils 5 Schritten.



Legende: i $K[i]$ $N[i]$

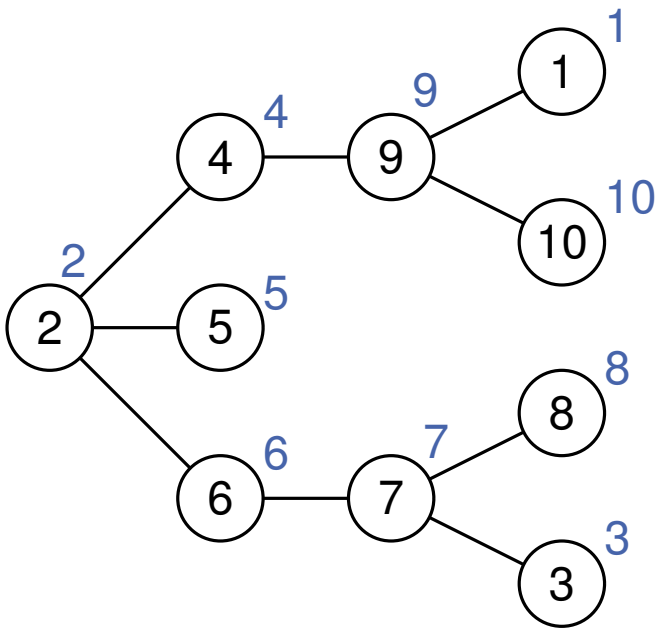
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

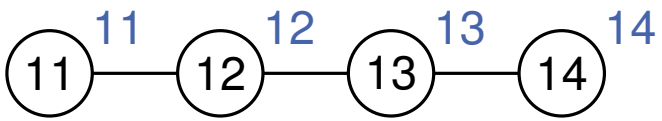
Zusammenhangskomponenten

Initialisierung

Für alle i : $1 \leq i \leq n$ **führe parallel aus**
$K[i] := i$

Vorheriger Schritt:





Legende: i $K[i]$ $N[i]$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Zusammenhangskomponenten

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 1. Schritt:

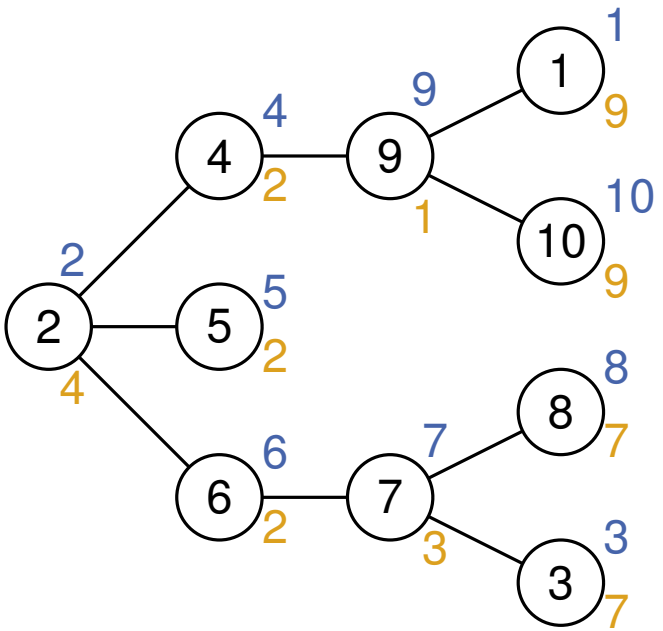
Für alle i : $1 \leq i \leq n$ **führe parallel aus**

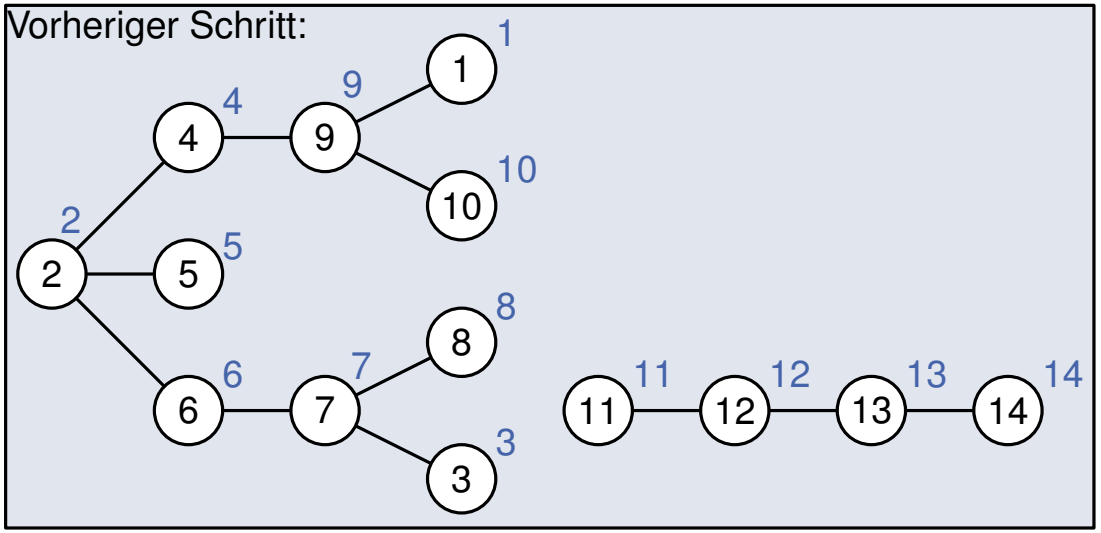
wenn *es Nachbarn j von i mit* $K[i] \neq K[j]$ *gibt* **dann**

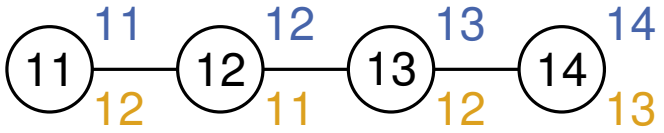
setze $N[i] := \min\{K[j] : \{i,j\} \in E \text{ und } K[i] \neq K[j]\}$

sonst N[i] = K[i]

$\ell = 1$







Legende: i $K[i]$ $N[i]$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

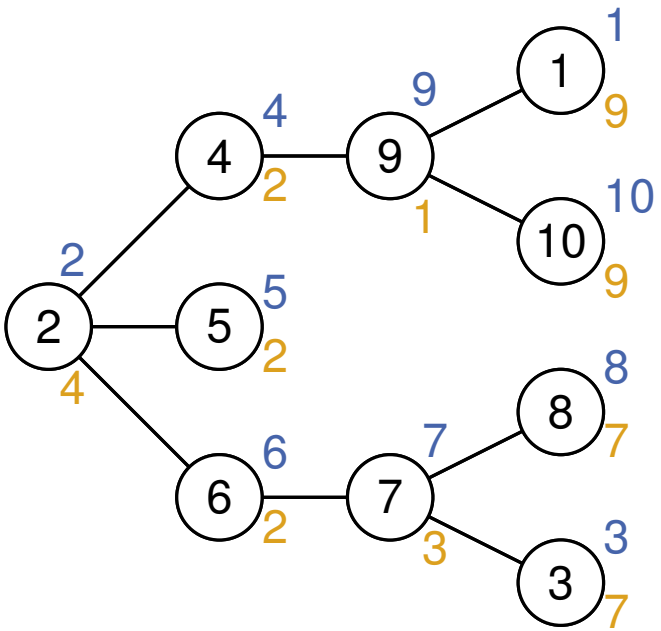
Zusammenhangskomponenten

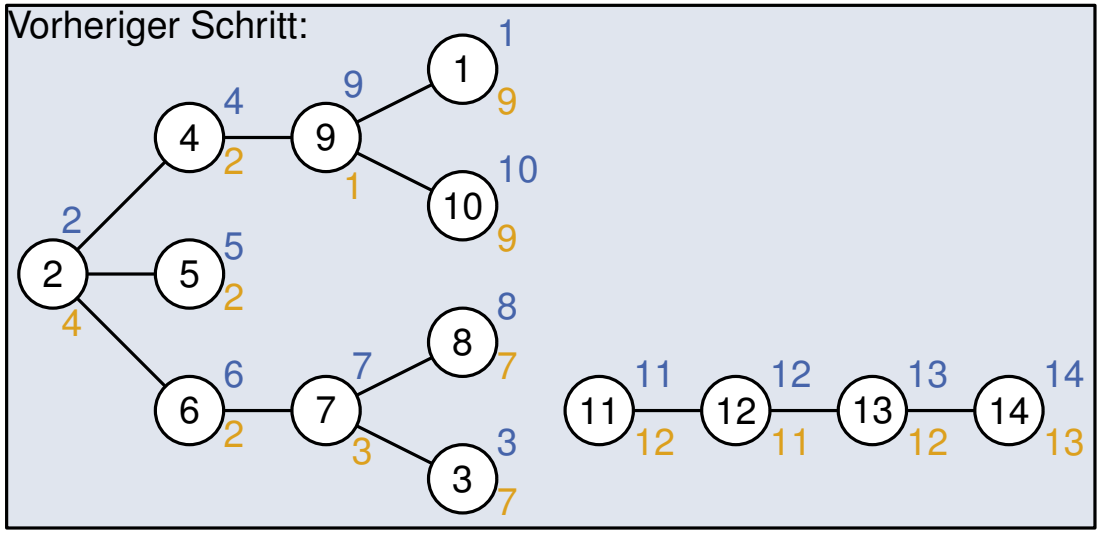
Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 2. Schritt:

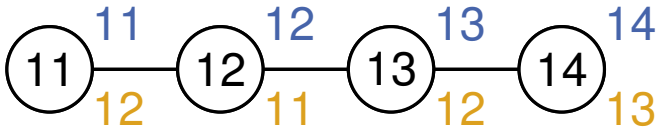
Für alle i : $1 \leq i \leq n$ **führe parallel aus**

$N[i] := \min\{N[j] : K[i] = K[j], N[j] \neq K[j]\}$

$\ell = 1$ Keine Veränderung!







Legende: i $K[i]$ $N[i]$

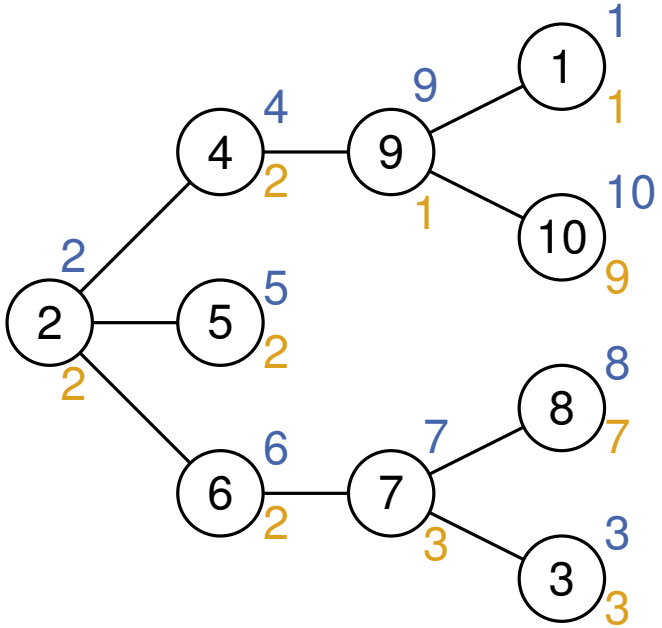
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Zusammenhangskomponenten

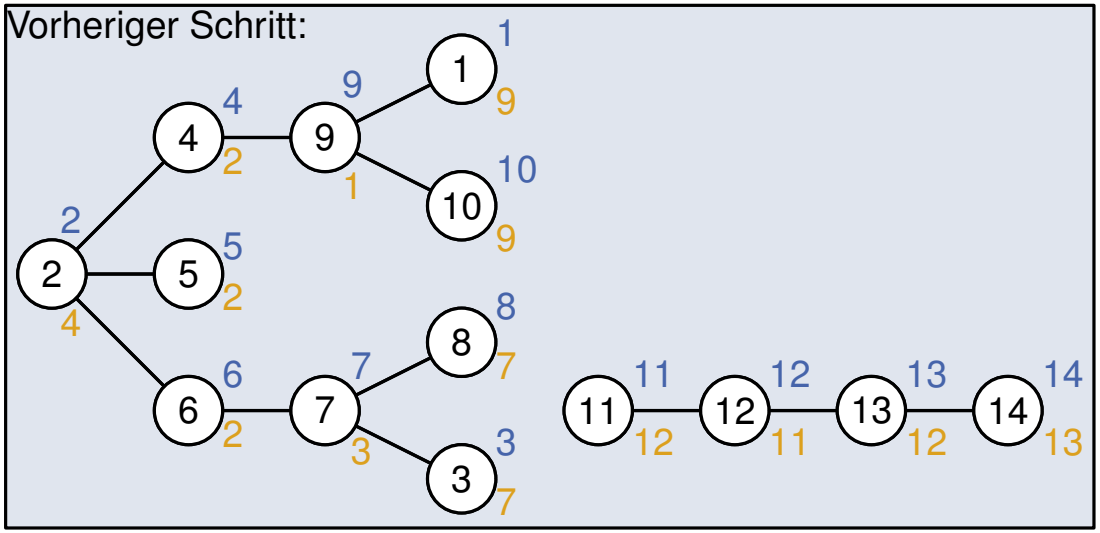
Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 3. Schritt:

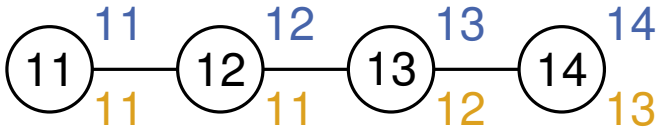
Für alle i : $1 \leq i \leq n$ **führe parallel aus**
$N[i] := \min\{N[i], K[i]\}$

$\ell = 1$



Vorheriger Schritt:





Legende: i $K[i]$ $N[i]$

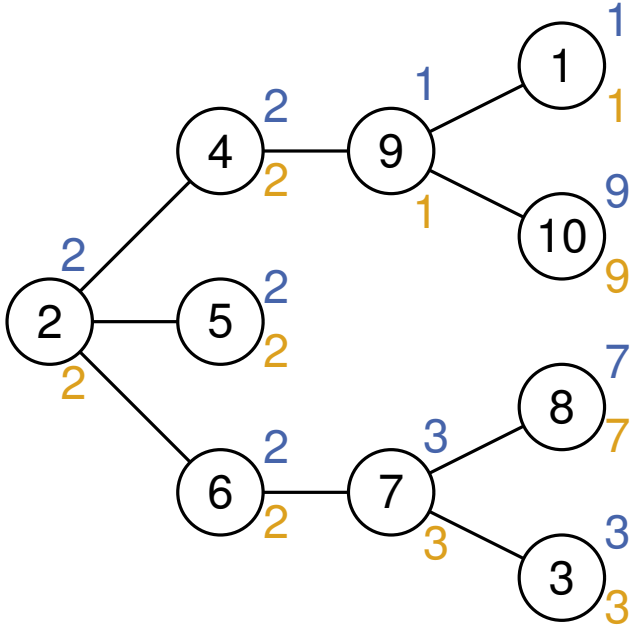
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Zusammenhangskomponenten

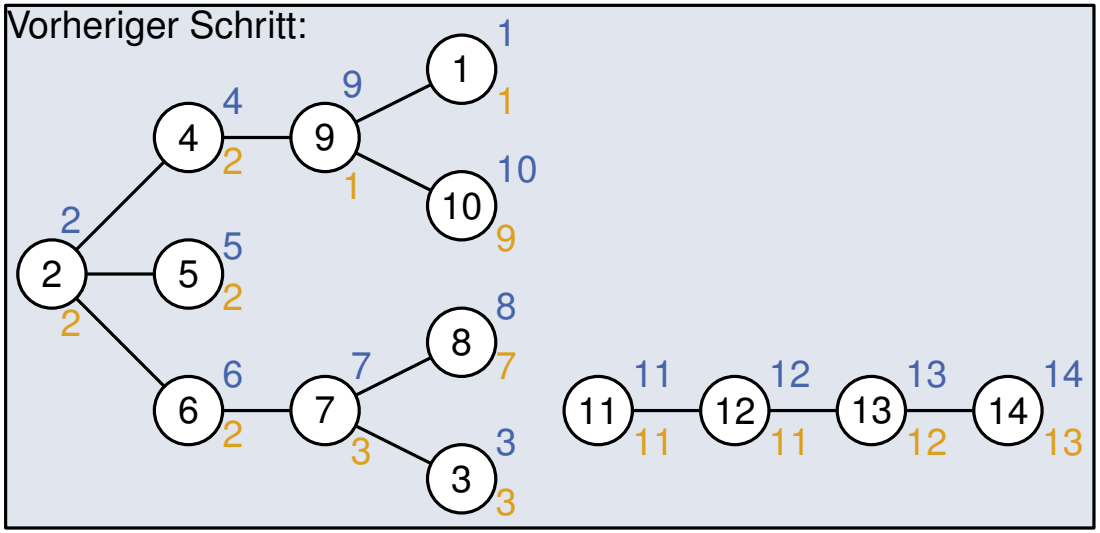
Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 4. Schritt:

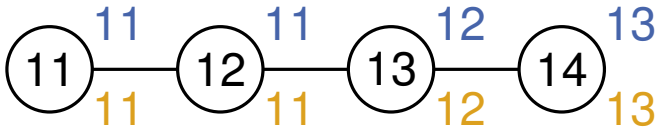
Für alle i : $1 \leq i \leq n$ **führe parallel aus**
$K[i] := N[i]$

$\ell = 1$



Vorheriger Schritt:





Legende: i $K[i]$ $N[i]$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

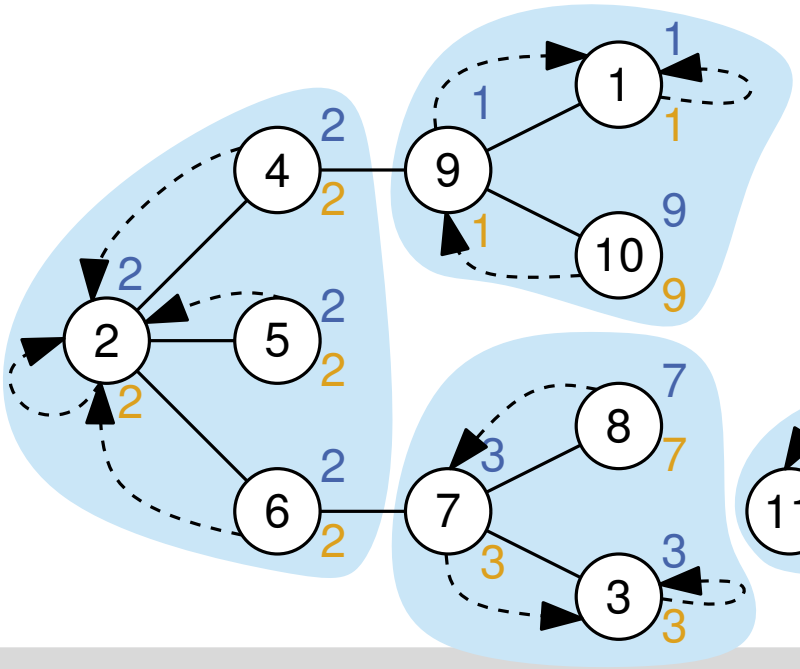
Zusammenhangskomponenten

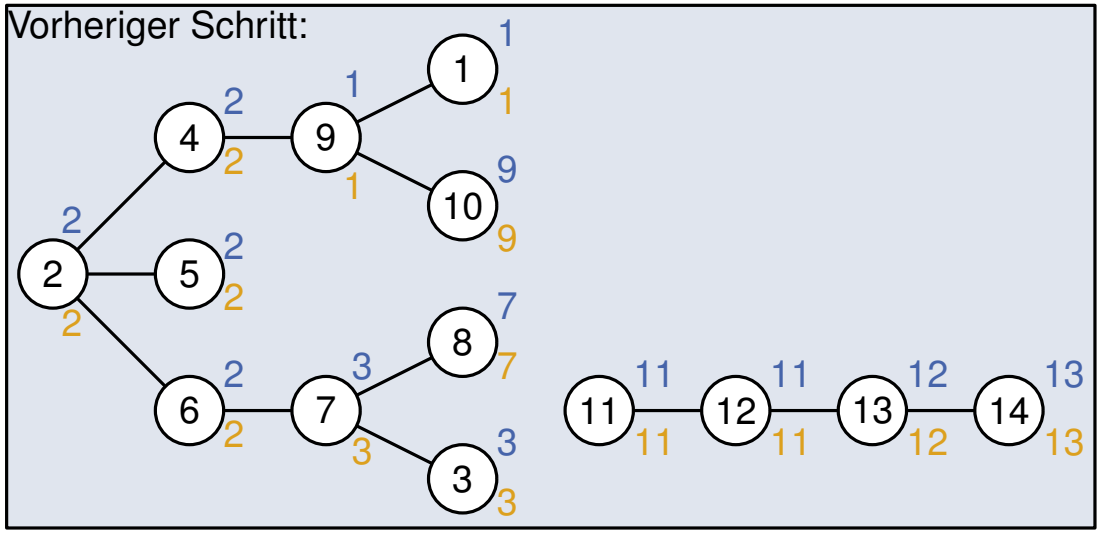
Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 5. Schritt:

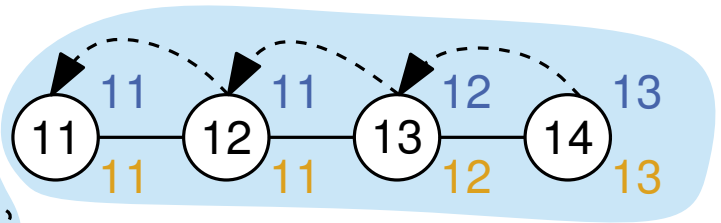
für $m = 1$ **bis** $\lceil \log n \rceil$ **tue**
Für alle $i : \ 1 \leq i \leq n$ **führe parallel aus**
$K[i] := K[K[i]]$

} LISTRANK auf Wurzelbäumen

$\ell = 1$







Legende: i $K[i]$ $N[i]$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

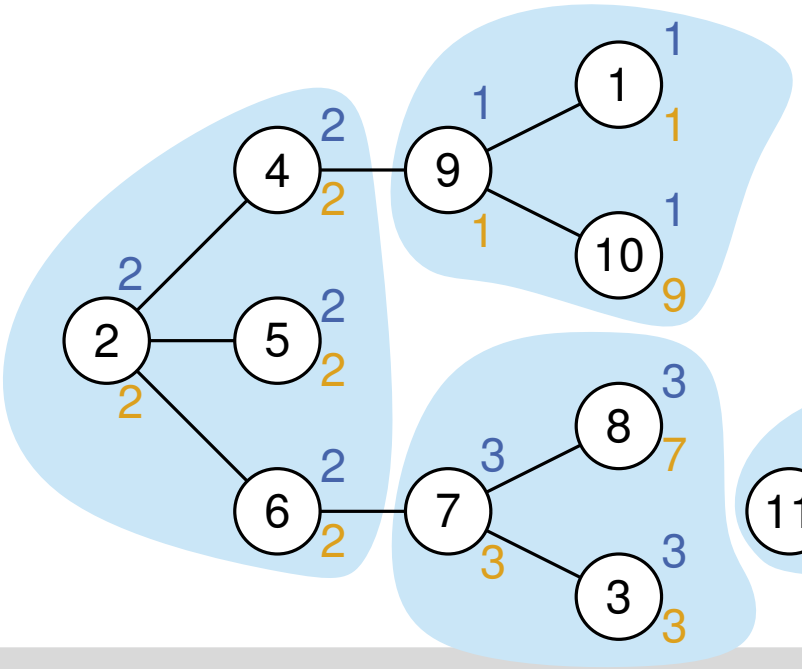
Zusammenhangskomponenten

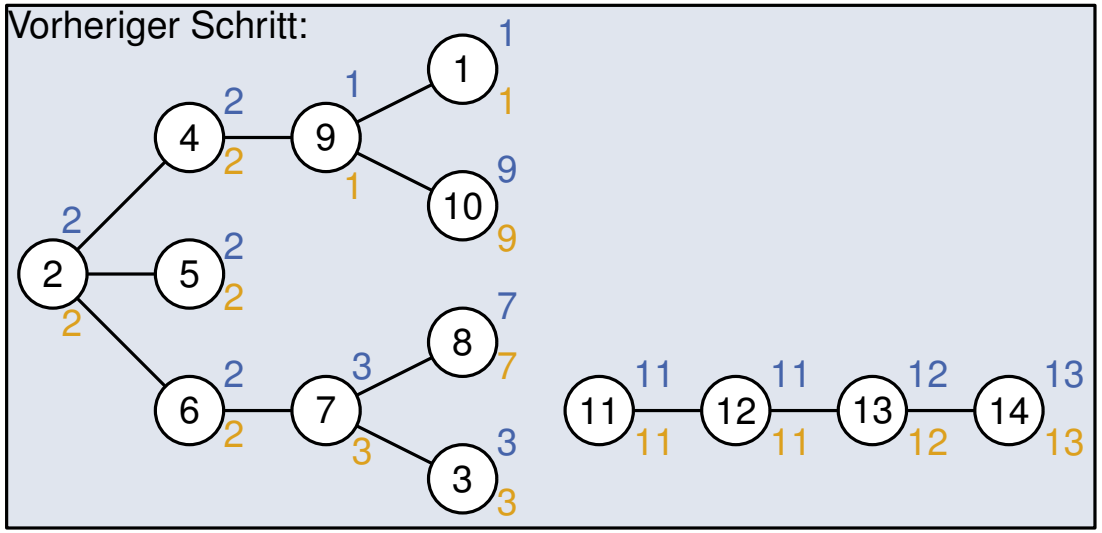
Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 5. Schritt:

für $m = 1$ **bis** $\lceil \log n \rceil$ **tue**
Für alle $i : 1 \leq i \leq n$ **führe parallel aus**
$K[i] := K[K[i]]$

} LISTRANK auf Wurzelbäumen

$\ell = 1$





Legende: i $K[i]$ $N[i]$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Zusammenhangskomponenten

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 1. Schritt:

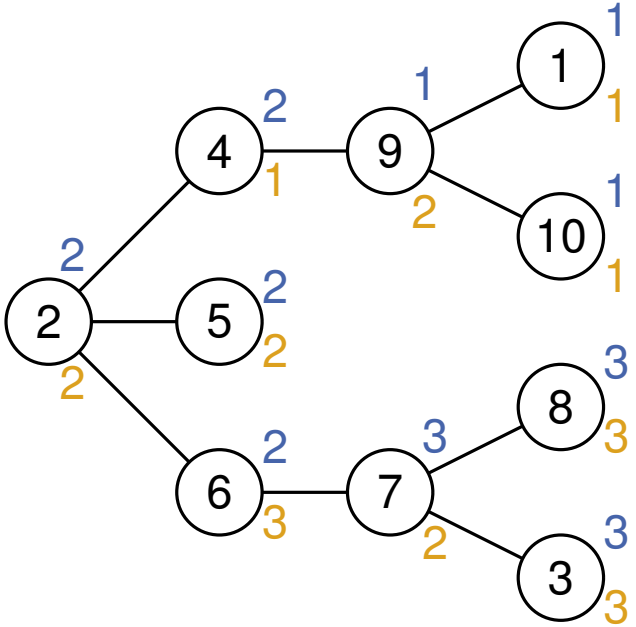
Für alle i : $1 \leq i \leq n$ **führe parallel aus**

wenn *es Nachbarn j von i mit* $K[i] \neq K[j]$ *gibt* **dann**

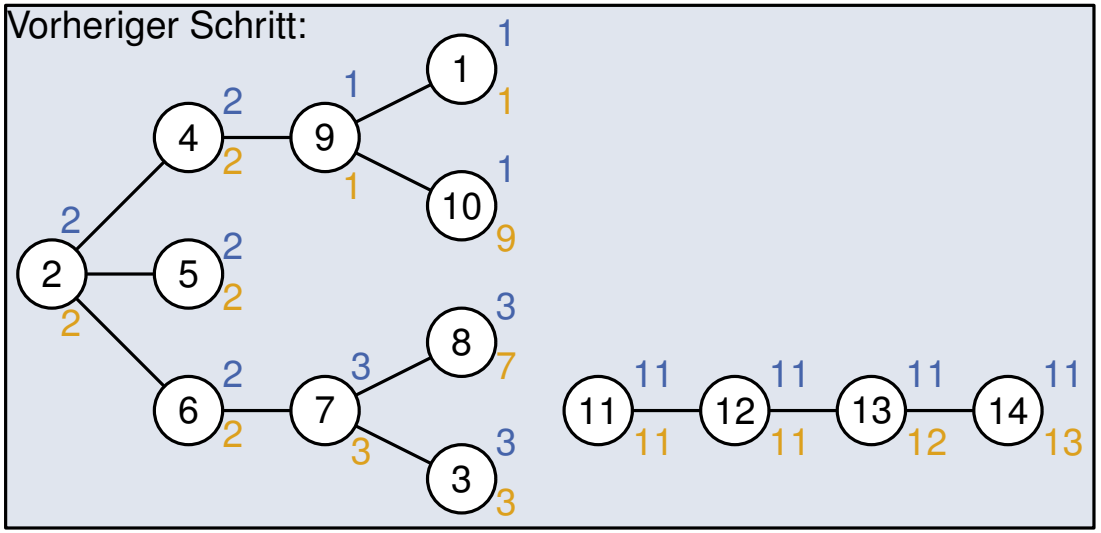
setze $N[i] := \min\{K[j] : \{i,j\} \in E \text{ und } K[i] \neq K[j]\}$

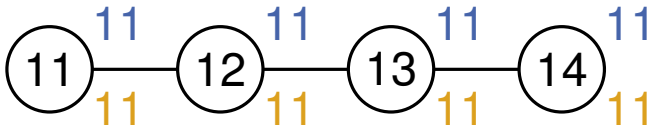
sonst N[i] = K[i]

$\ell = 2$



Vorheriger Schritt:





Legende: i $K[i]$ $N[i]$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

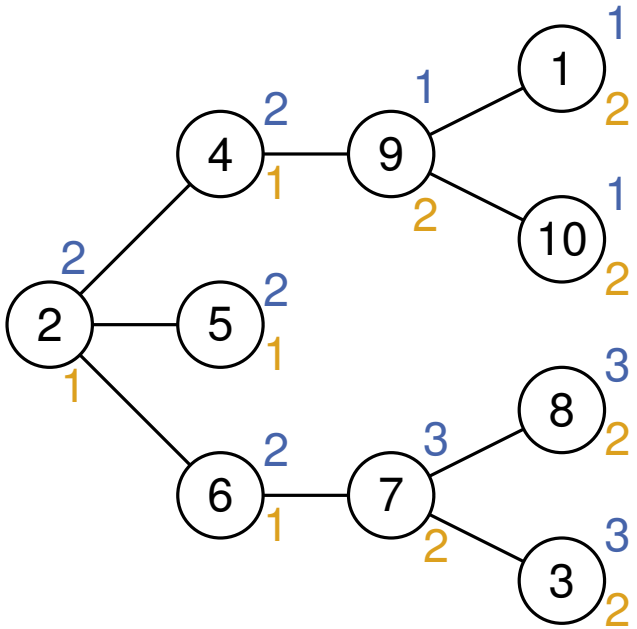
Zusammenhangskomponenten

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 2. Schritt:

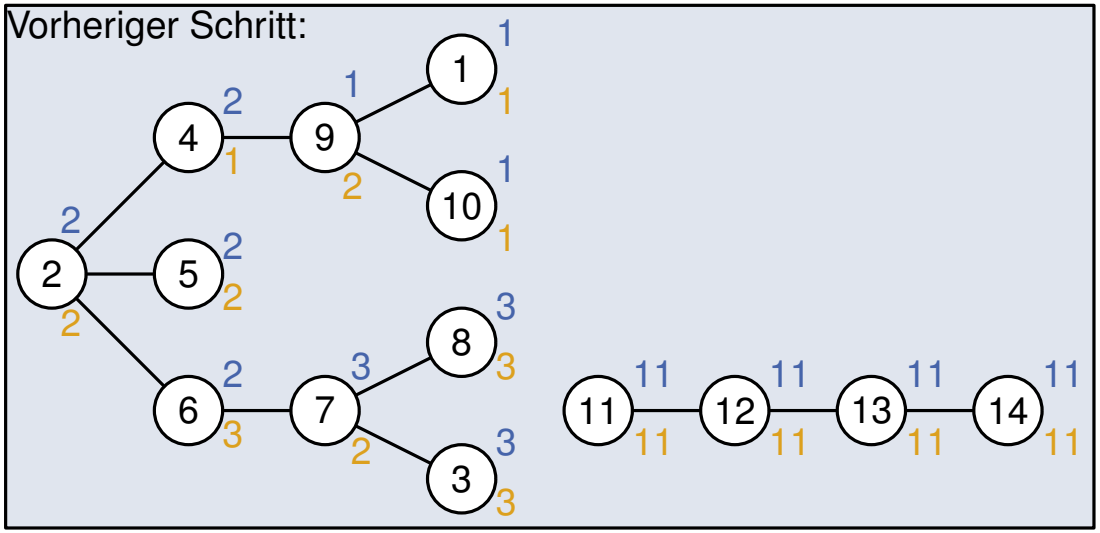
Für alle i : $1 \leq i \leq n$ **führe parallel aus**

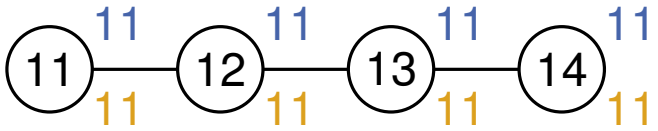
$N[i] := \min\{N[j] : K[i] = K[j], N[j] \neq K[j]\}$

$\ell = 2$



Vorheriger Schritt:





Legende: i $K[i]$ $N[i]$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

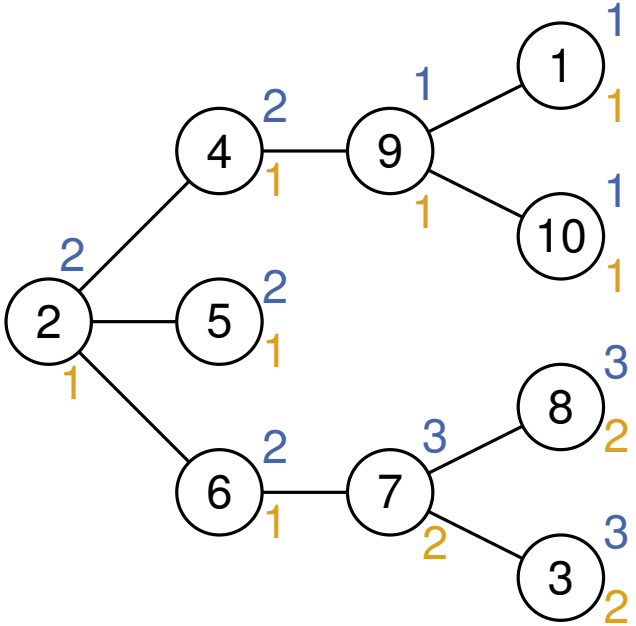
Zusammenhangskomponenten

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 3. Schritt:

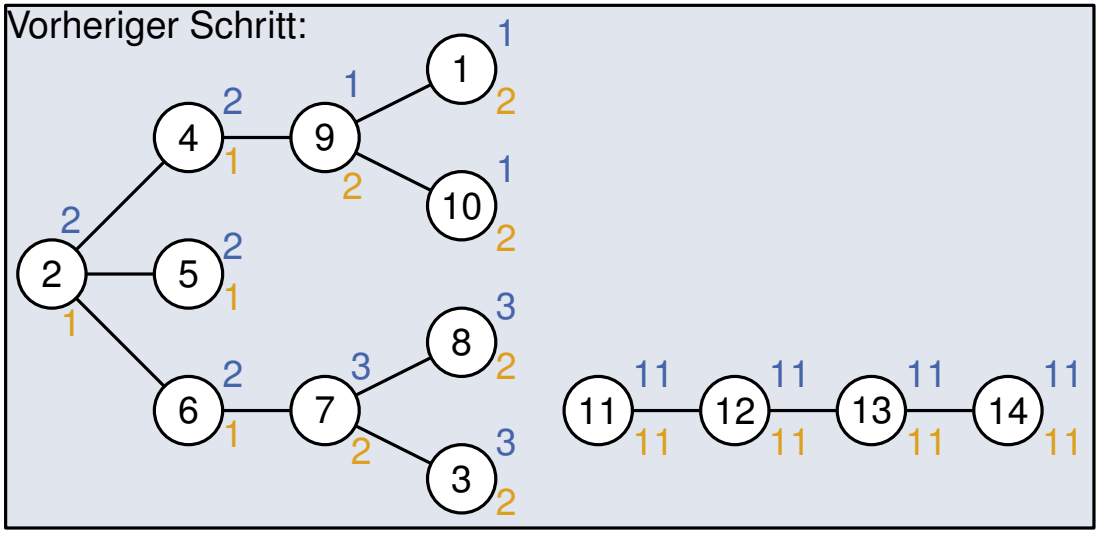
Für alle i : $1 \leq i \leq n$ **führe parallel aus**

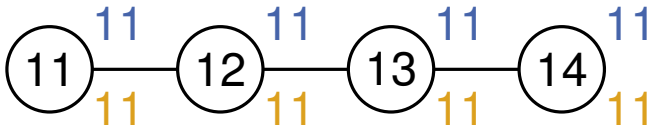
$N[i] := \min\{N[i], K[i]\}$

$\ell = 2$



Vorheriger Schritt:





Legende: i $K[i]$ $N[i]$

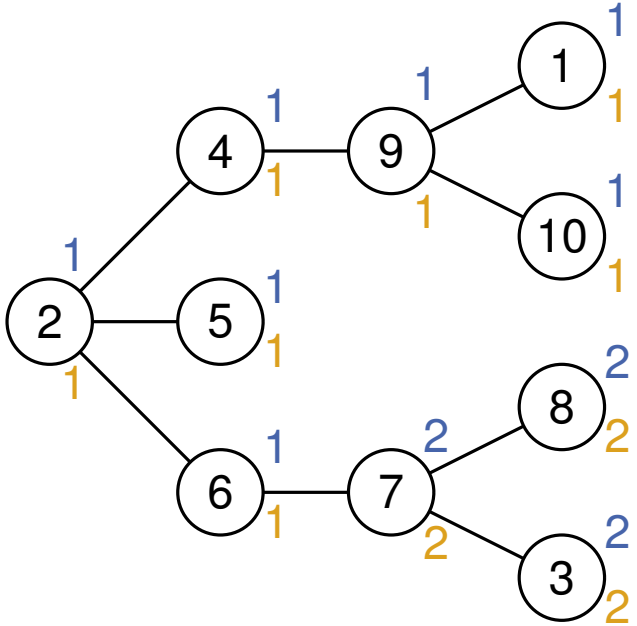
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Zusammenhangskomponenten

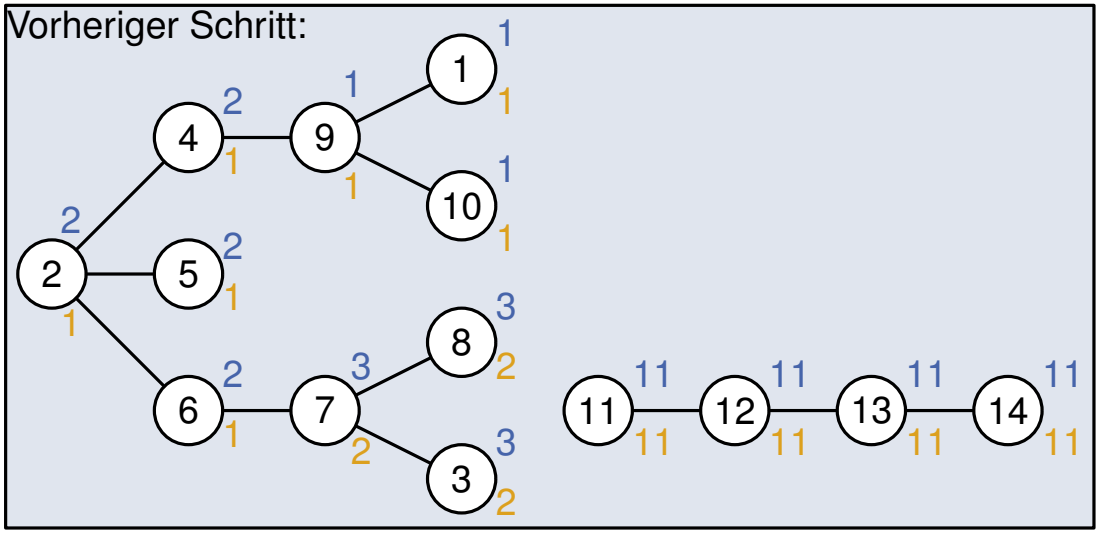
Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 4. Schritt:

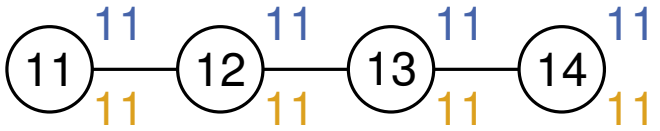
Für alle i : $1 \leq i \leq n$ **führe parallel aus**
$K[i] := N[i]$

$\ell = 2$



Vorheriger Schritt:





Legende: i $K[i]$ $N[i]$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

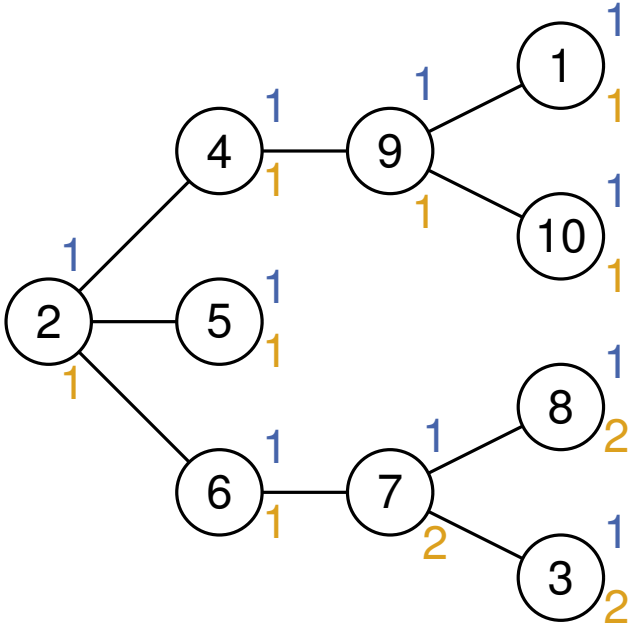
Zusammenhangskomponenten

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 5. Schritt:

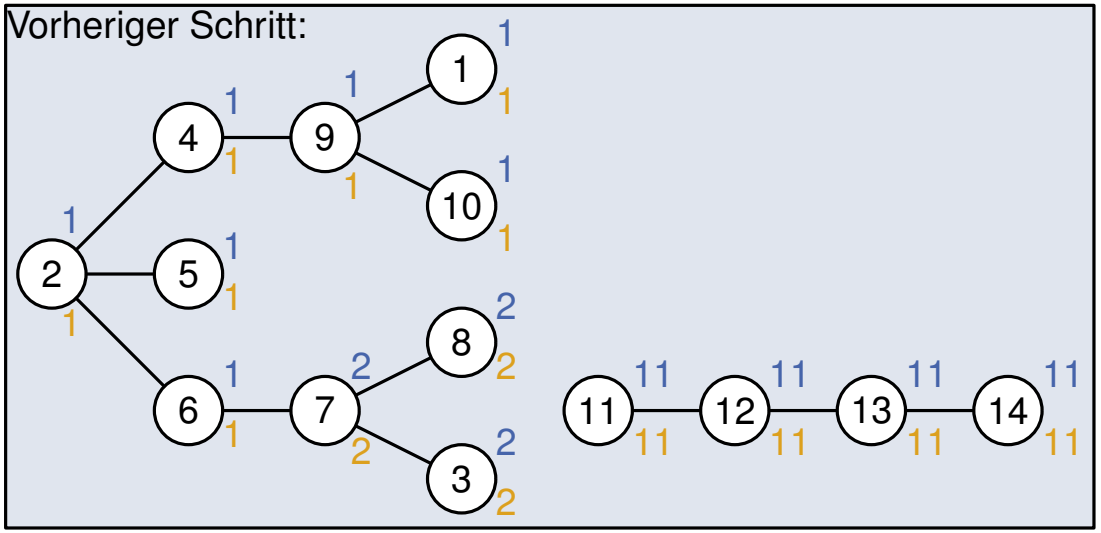
für $m = 1$ **bis** $\lceil \log n \rceil$ **tue**
Für alle $i : \ 1 \leq i \leq n$ **führe parallel aus**
$K[i] := K[K[i]]$

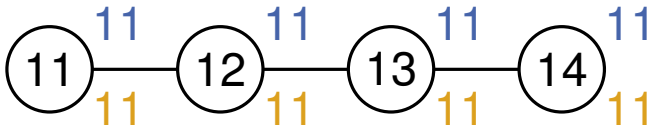
} LISTRANK auf Wurzelbäumen

$\ell = 2$



Vorheriger Schritt:





Legende: i $K[i]$ $N[i]$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Zusammenhangskomponenten

Initialisierung $O(1)$ n

Für alle i : $1 \leq i \leq n$ **führe parallel aus** $O(1)$ n

$K[i] := i$

Phase ℓ **mit** $1 \leq \ell \leq \lceil \log n \rceil$ $O(\log^2 n)$ n^2

Für alle i : $1 \leq i \leq n$ **führe parallel aus** $O(\log n)$ n^2

wenn *es Nachbarn* j *von* i *mit* $K[i] \neq K[j]$ *gibt* **dann**

setze $N[i] := \min\{K[j] : \{i,j\} \in E \text{ und } K[i] \neq K[j]\}$ $O(\log n)$ n

sonst N[i] = K[i]

Für alle i : $1 \leq i \leq n$ **führe parallel aus** $O(\log n)$ n^2

$N[i] := \min\{N[j] : K[i] = K[j], N[j] \neq K[j]\}$ $O(\log n)$ n

Für alle i : $1 \leq i \leq n$ **führe parallel aus** $O(1)$ n

$N[i] := \min\{N[i], K[i]\}$

Für alle i : $1 \leq i \leq n$ **führe parallel aus** $O(1)$ n

$K[i] := N[i]$

für $m = 1$ **bis** $\lceil \log n \rceil$ **tue** $O(\log n)$ n

Für alle i : $1 \leq i \leq n$ **führe parallel aus**

$K[i] := K[K[i]]$

Laufzeit
Prozessorenanzahl

Gesamte Laufzeit.

1. Prozessorenzahl kann durch Rescheudling auf $\left\lceil \frac{n^2}{\log n} \right\rceil$ reduziert werden.
2. Somit sind die Kosten $O(n^2 \log n)$, was nicht kostenoptimal ist.

Minimaler Spannbaum (MST)

Problemstellung
gegeben: Graph $G = (V, E)$, $V = \{1, \ldots, n\}$ und Kantengewichte $c_{i,j}$ für $\{i, j\} \in 2^V$, wobei $c_{i,j} = \infty$ wenn $\{i, j\} \notin E$.
gesucht: Spannbaum in G mit minimalen Gewicht.

Idee: Passe Algorithmus zur Bestimmung der Zusammenhangskomponenten an.

Datenstrukturen:

- Array K der Länge n: K[i] = Index der aktuellen Komponente in der i liegt.

  Also: Für alle Knoten $i_1, \ldots, i_\ell$ der selben Komponente gilt: $K[i_1] = \ldots = K[i_\ell] = k$. Insbesondere wird gelten: $k = \min\{i_1, \ldots, i_\ell\}$. Knoten k ist *Repräsentant der Komponente*.

- Array T enthält die Kanten, die zum MST gehören.
- Array S der Länge n: $S[i]$= Index desjenigen Knoten, der in der selben Komponente wie i ist und eine Kante minimalen Gewichts zu einem Knoten einer anderen Komponente hat.
- Array N der Länge n: N[i] = Index desjenigen Knotens, zu dem es von i aus eine Kante minimalen Gewichts gibt und der in einer anderen Komponente als i liegt.

Minimaler Spannbaum (MST)

Problemstellung

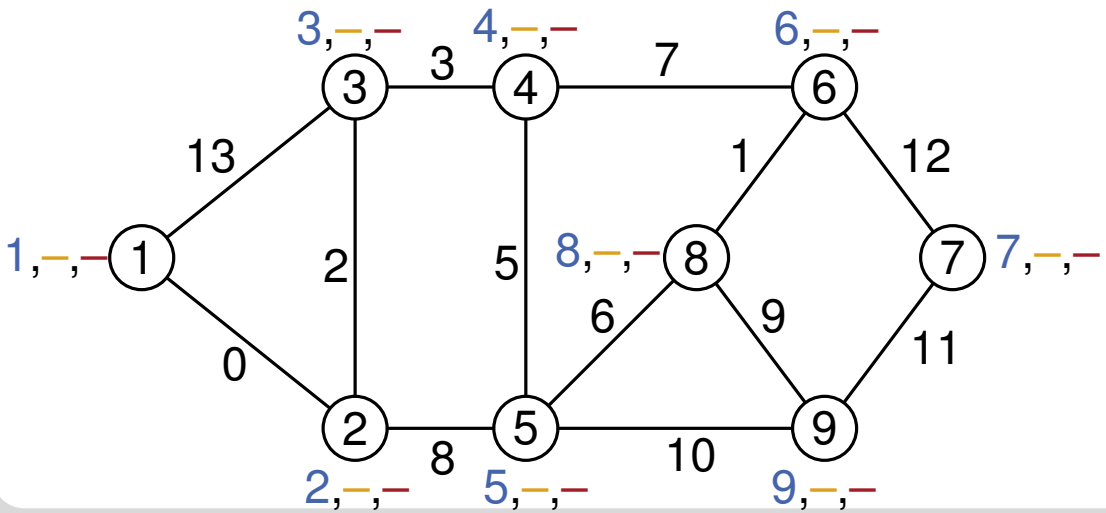
gegeben: Graph $G = (V, E)$, $V = \{1, \ldots, n\}$ und Kantengewichte $c_{i,j}$ für $\{i, j\} \in 2^V$, wobei $c_{i,j} = \infty$ wenn $\{i, j\} \notin E$.

gesucht: Spannbaum in G mit minimalen Gewicht.

Initialisierung

Für alle i : $1 \leq i \leq n$ **führe parallel aus**

$K[i] := i$



Vorheriger Schritt:

Legende: $K[i]$,$N[i]$,$S[i]$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

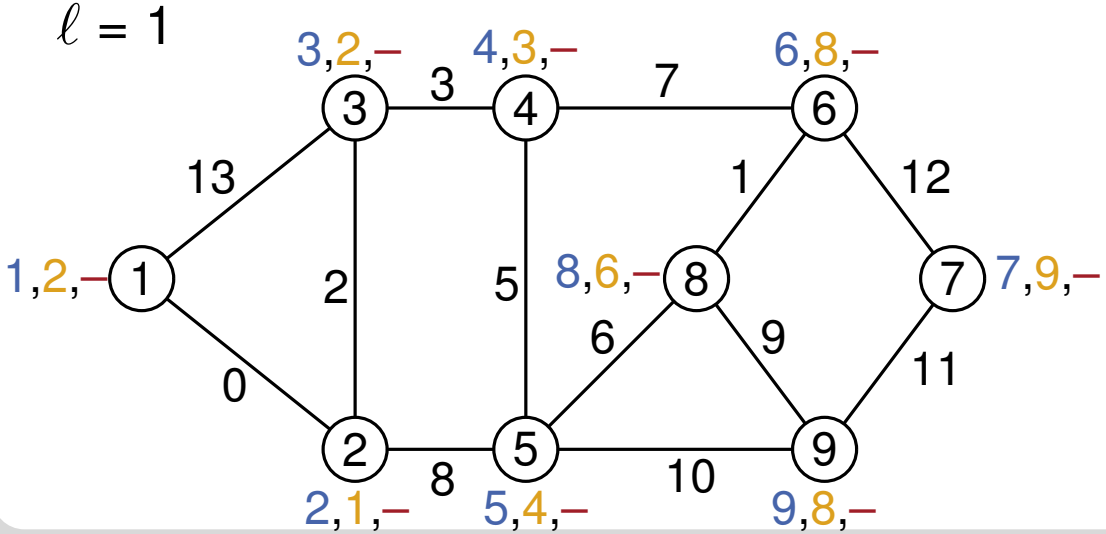
Minimaler Spannbaum (MST)

Problemstellung
gegeben: Graph $G = (V, E)$, $V = \{1, \ldots, n\}$ und Kantengewichte $c_{i,j}$ für $\{i, j\} \in 2^V$, wobei $c_{i,j} = \infty$ wenn $\{i, j\} \notin E$.
gesucht: Spannbaum in G mit minimalen Gewicht.

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 1. Schritt:

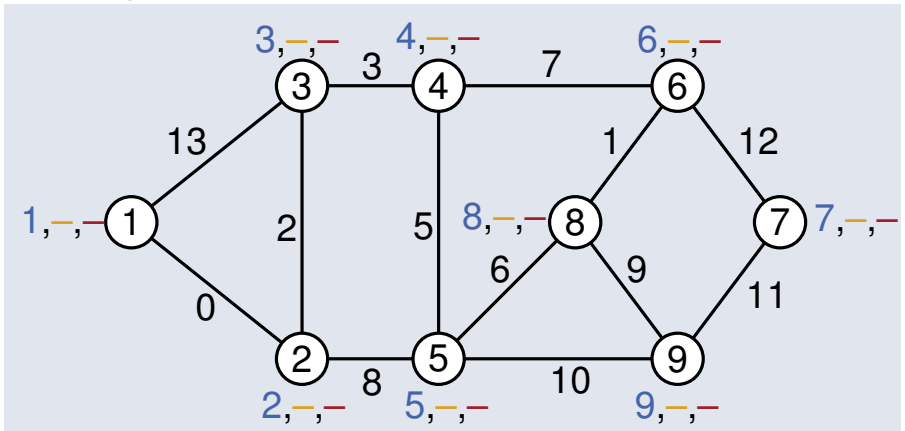
Für alle i : $1 \leq i \leq n$ **führe parallel aus**
Finde $k \in V$ mit $K[i] \neq K[k]$ und $c_{ik} = \min\{c_{ij} : 1 \leq j \leq n, K[i] \neq K[j]\}$
$N[i] := k$

$\ell = 1$



Vorheriger Schritt:

Legende: $K[i]$,$N[i]$,$S[i]$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Minimaler Spannbaum (MST)

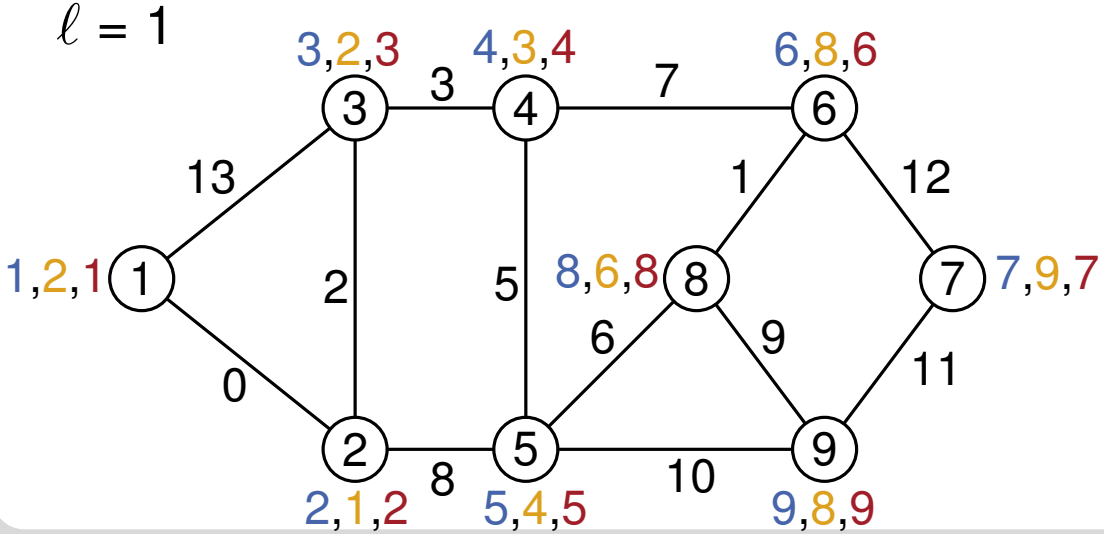
Problemstellung
gegeben: Graph $G = (V, E)$, $V = \{1, \ldots, n\}$ und Kantengewichte $c_{i,j}$ für $\{i, j\} \in 2^V$, wobei $c_{i,j} = \infty$ wenn $\{i, j\} \notin E$.
gesucht: Spannbaum in G mit minimalen Gewicht.

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 2. Schritt:

Finde $t \in V$ mit $K[i] = K[t]$ und $c_{tN[t]} = \min\{c_{jN[j]} : 1 \leq j \leq n, K[i] = K[j]\}$

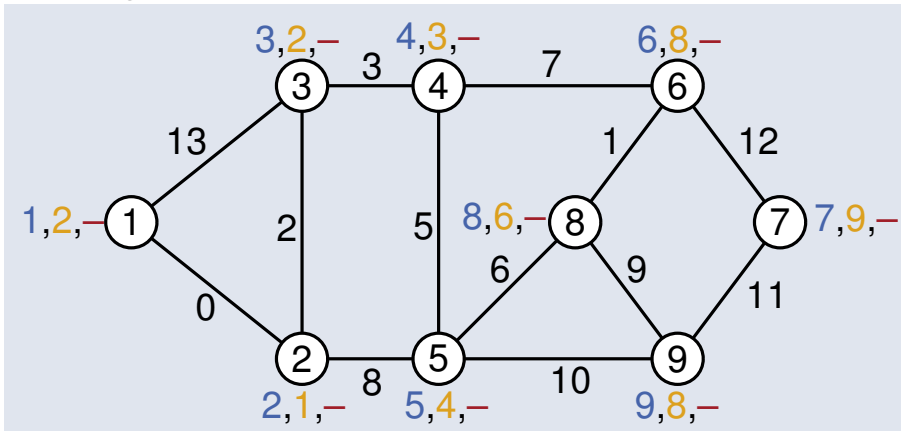
setze $N[i] := N[t]$ und $S[i] := t$

$\ell = 1$



Vorheriger Schritt:

Legende: $K[i]$,$N[i]$,$S[i]$



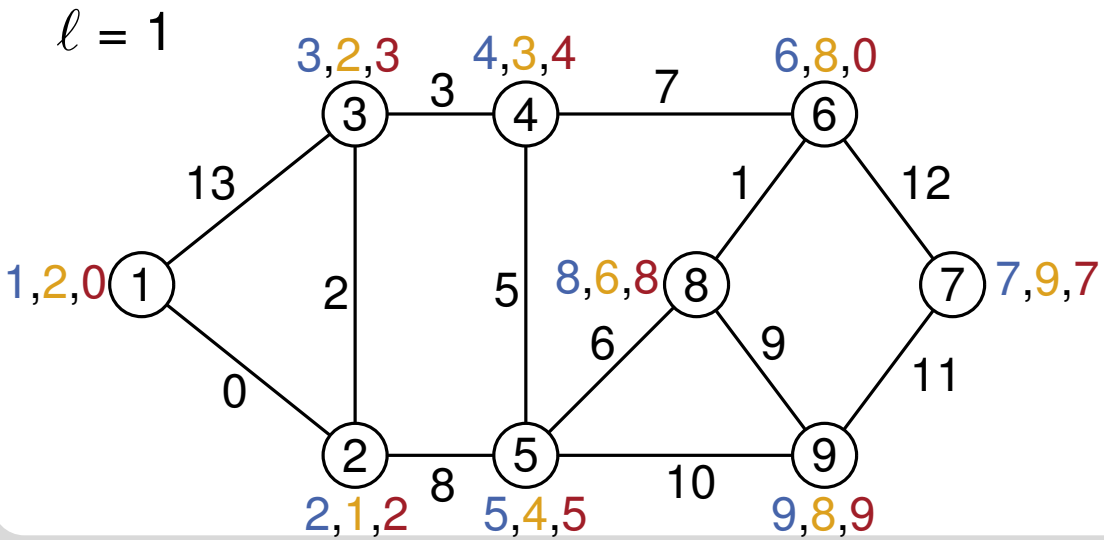
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Minimaler Spannbaum (MST)

Problemstellung
gegeben: Graph $G = (V, E)$, $V = \{1, \ldots, n\}$ und Kantengewichte $c_{i,j}$ für $\{i, j\} \in 2^V$, wobei $c_{i,j} = \infty$ wenn $\{i, j\} \notin E$.
gesucht: Spannbaum in G mit minimalen Gewicht.

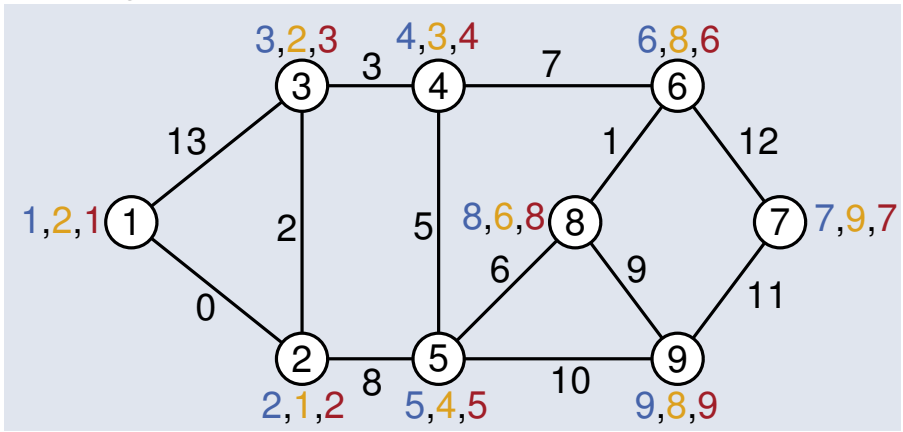
Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 3. Schritt:

Für alle i : $1 \leq i \leq n$ **führe parallel aus**
wenn $N[N[i]] = S[i]$ *und* $K[i] < K[N[i]]$ **dann**
setze $S[i] := 0$



Vorheriger Schritt:

Legende: $K[i], N[i], S[i]$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Minimaler Spannbaum (MST)

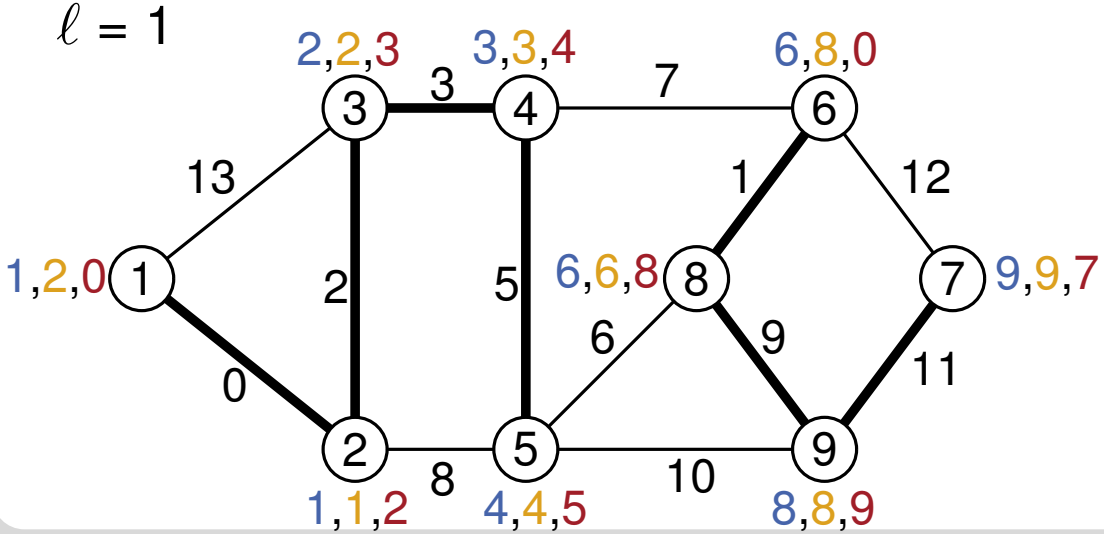
Problemstellung
gegeben: Graph $G = (V, E)$, $V = \{1, \ldots, n\}$ und Kantengewichte $c_{i,j}$ für $\{i, j\} \in 2^V$, wobei $c_{i,j} = \infty$ wenn $\{i, j\} \notin E$.
gesucht: Spannbaum in G mit minimalen Gewicht.

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 4. Schritt:

Für alle i : $1 \leq i \leq n$ **führe parallel aus**
wenn $S[i] \neq 0$ *und* $K[i] = i$ *und* $c_{N[i],S[i]} \neq \infty$ **dann**
setze $T := T \cup \{N[i], S[i]\}$

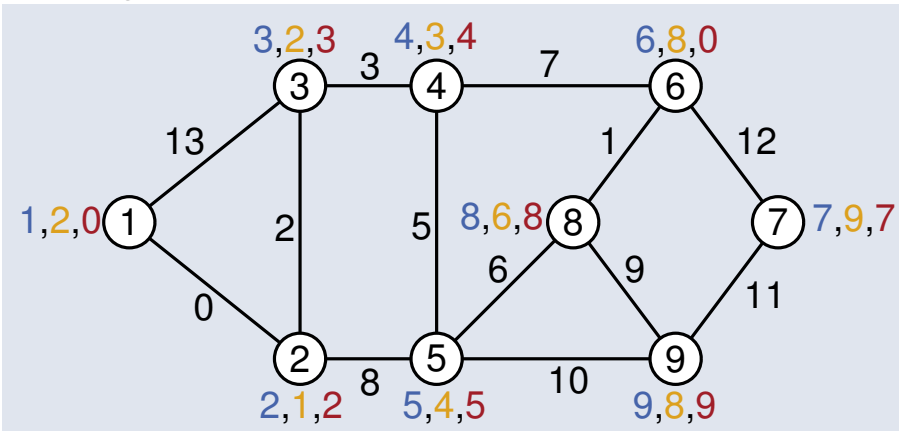
wenn $S[i] \neq 0$ **dann**
setze $K[i] = K[N[i]]$

$\ell = 1$



Vorheriger Schritt:

Legende: $K[i]$,$N[i]$,$S[i]$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Minimaler Spannbaum (MST)

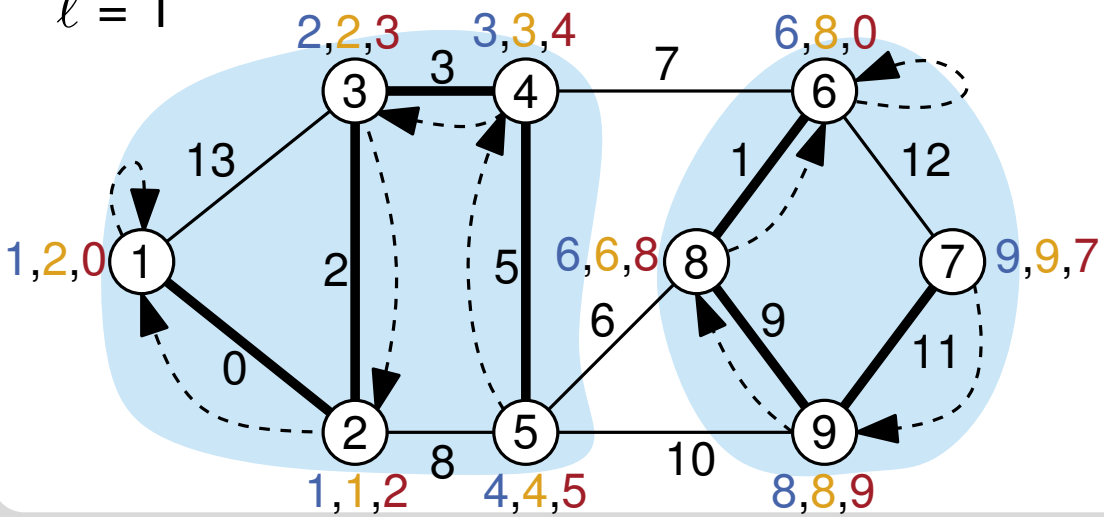
Problemstellung
gegeben: Graph $G = (V, E)$, $V = \{1, \ldots, n\}$ und Kantengewichte $c_{i,j}$ für $\{i, j\} \in 2^V$, wobei $c_{i,j} = \infty$ wenn $\{i, j\} \notin E$.
gesucht: Spannbaum in G mit minimalen Gewicht.

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 5. Schritt:

für $m = 1$ *bis* $\lceil \log n \rceil$ **tue**
Für alle i : $1 \leq i \leq n$ **führe parallel aus**
$K[i] := K[K[i]]$

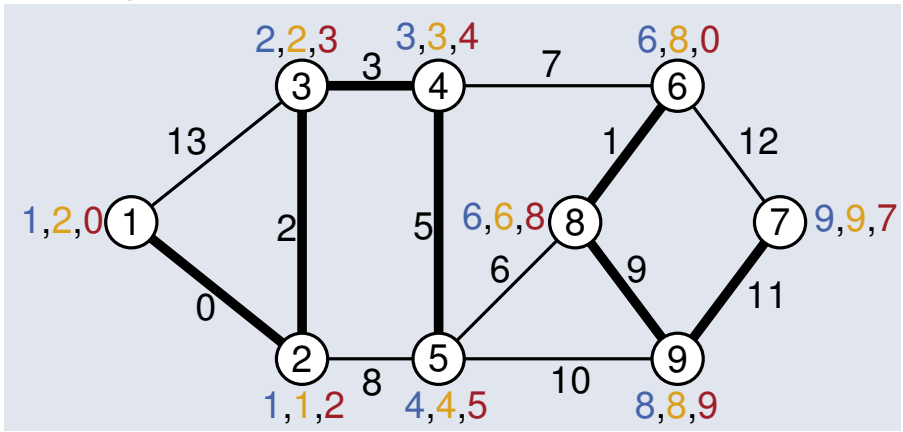
} LISTRANK auf Wurzelbäumen

$\ell = 1$



Vorheriger Schritt:

Legende: $K[i], N[i], S[i]$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Minimaler Spannbaum (MST)

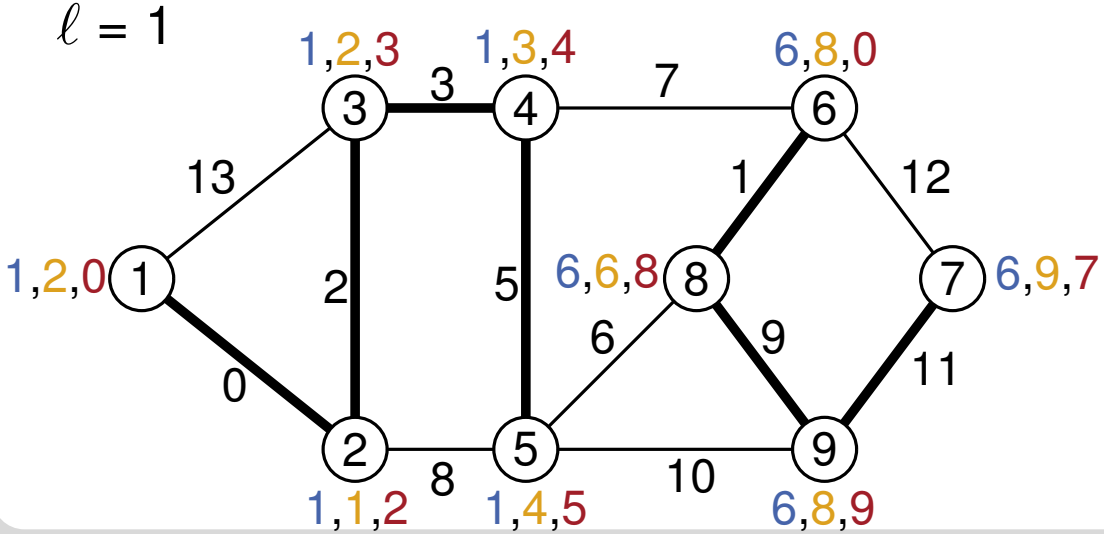
Problemstellung
gegeben: Graph $G = (V, E)$, $V = \{1, \ldots, n\}$ und Kantengewichte $c_{i,j}$ für $\{i, j\} \in 2^V$, wobei $c_{i,j} = \infty$ wenn $\{i, j\} \notin E$.
gesucht: Spannbaum in G mit minimalen Gewicht.

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 5. Schritt:

für $m = 1$ *bis* $\lceil \log n \rceil$ **tue**
Für alle i : $1 \leq i \leq n$ **führe parallel aus**
$K[i] := K[K[i]]$

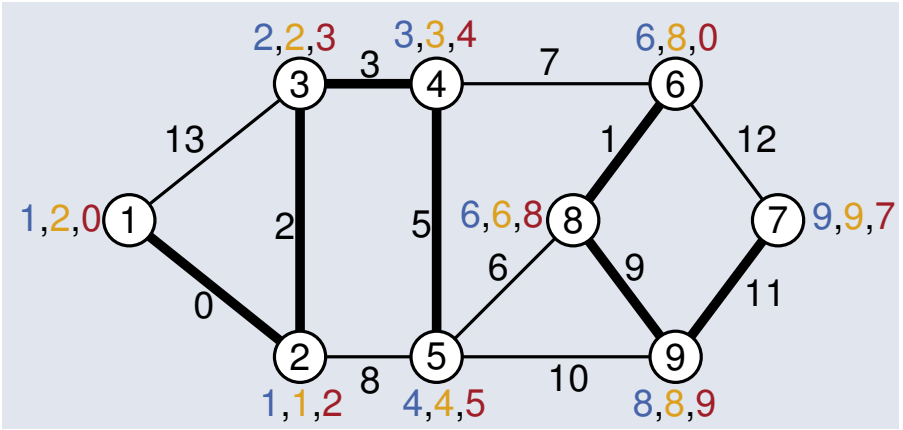
} ListRank auf Wurzelbäumen

$\ell = 1$



Vorheriger Schritt:

Legende: $K[i], N[i], S[i]$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

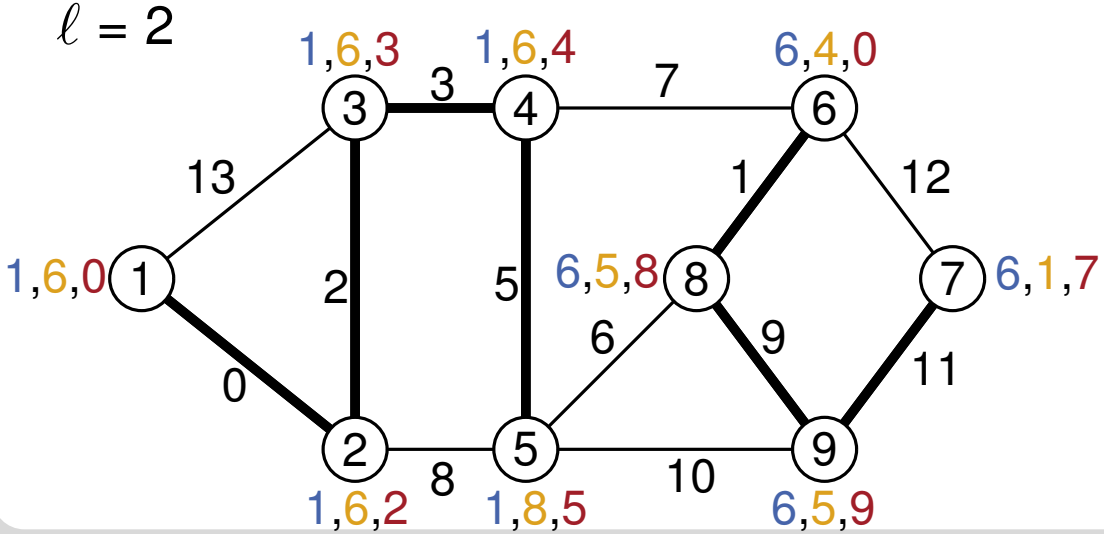
Minimaler Spannbaum (MST)

Problemstellung
gegeben: Graph $G = (V, E)$, $V = \{1, \ldots, n\}$ und Kantengewichte $c_{i,j}$ für $\{i, j\} \in 2^V$, wobei $c_{i,j} = \infty$ wenn $\{i, j\} \notin E$.
gesucht: Spannbaum in G mit minimalen Gewicht.

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 1. Schritt:

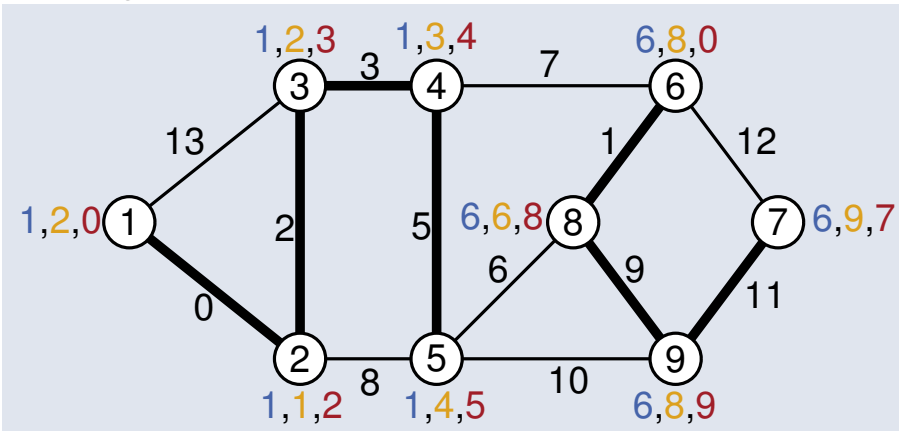
Für alle i : $1 \leq i \leq n$ **führe parallel aus**
Finde $k \in V$ mit $K[i] \neq K[k]$ und $c_{ik} = \min\{c_{ij} : 1 \leq j \leq n, K[i] \neq K[j]\}$
$N[i] := k$

$\ell = 2$



Vorheriger Schritt:

Legende: $K[i], N[i], S[i]$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Minimaler Spannbaum (MST)

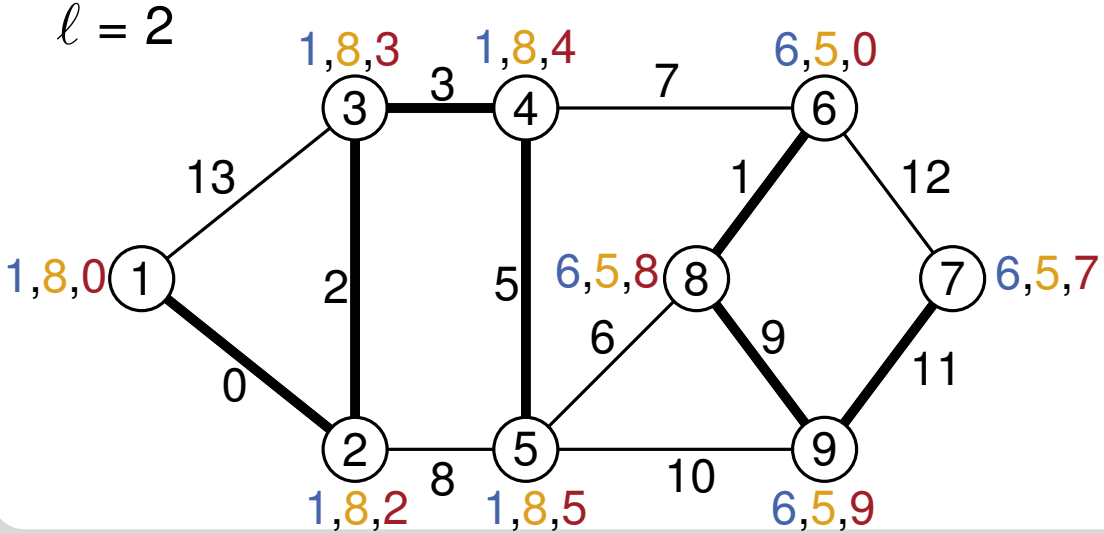
Problemstellung
gegeben: Graph $G = (V, E)$, $V = \{1, \ldots, n\}$ und Kantengewichte $c_{i,j}$ für $\{i, j\} \in 2^V$, wobei $c_{i,j} = \infty$ wenn $\{i, j\} \notin E$.
gesucht: Spannbaum in G mit minimalen Gewicht.

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 2. Schritt:

Finde $t \in V$ mit $K[i] = K[t]$ und $c_{tN[t]} = \min\{c_{jN[j]} : 1 \leq j \leq n, K[i] = K[j]\}$

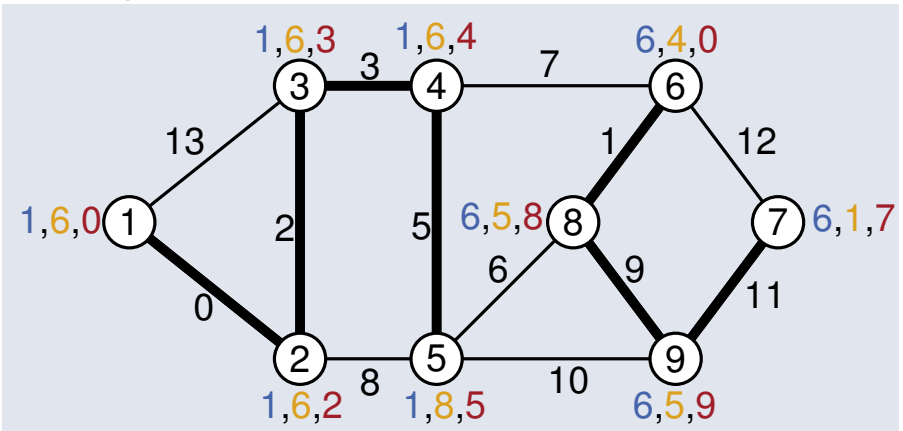
setze $N[i] := N[t]$ und $S[i] := t$

$\ell = 2$



Vorheriger Schritt:

Legende: $K[i], N[i], S[i]$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

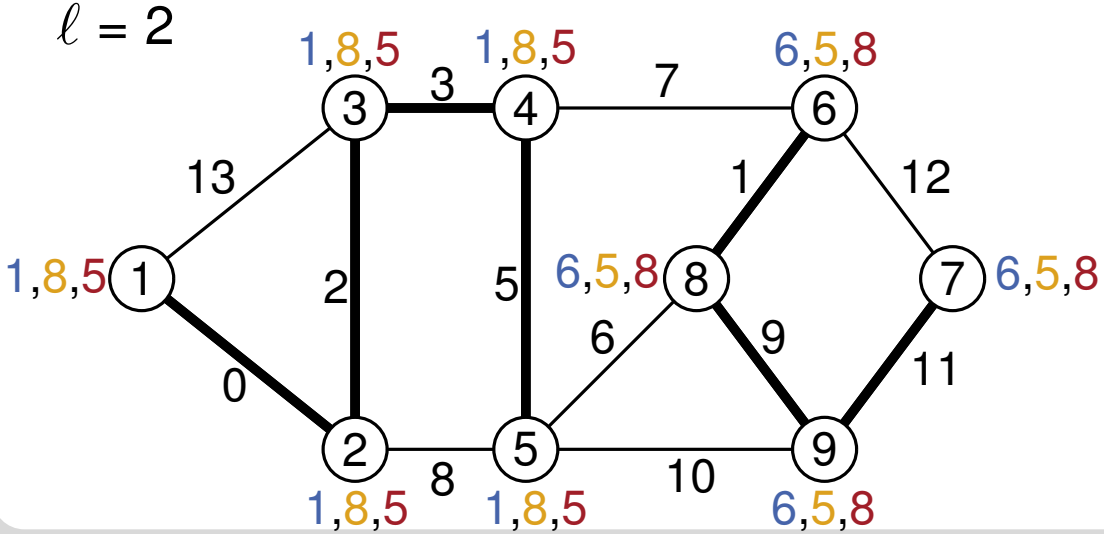
Minimaler Spannbaum (MST)

Problemstellung
gegeben: Graph $G = (V, E)$, $V = \{1, \ldots, n\}$ und Kantengewichte $c_{i,j}$ für $\{i, j\} \in 2^V$, wobei $c_{i,j} = \infty$ wenn $\{i, j\} \notin E$.
gesucht: Spannbaum in G mit minimalen Gewicht.

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 3. Schritt:

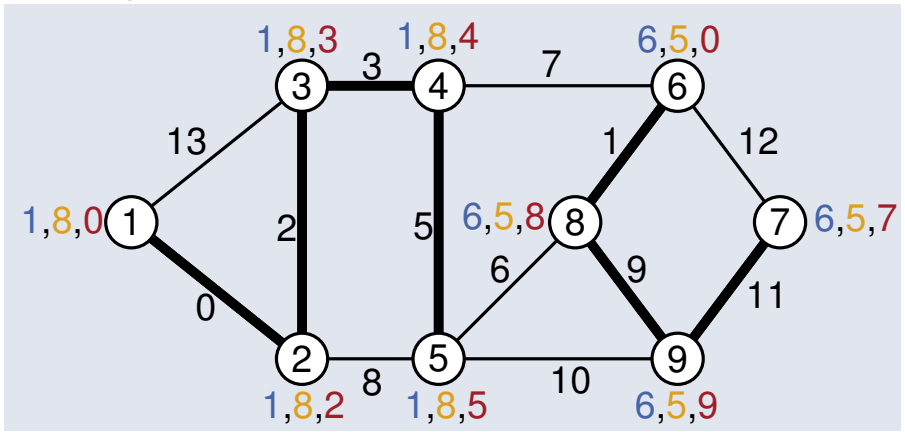
Für alle i : $1 \leq i \leq n$ **führe parallel aus**
wenn $N[N[i]] = S[i]$ *und* $K[i] < K[N[i]]$ **dann**
setze $S[i] := 0$

$\ell = 2$



Vorheriger Schritt:

Legende: $K[i]$,$N[i]$,$S[i]$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Minimaler Spannbaum (MST)

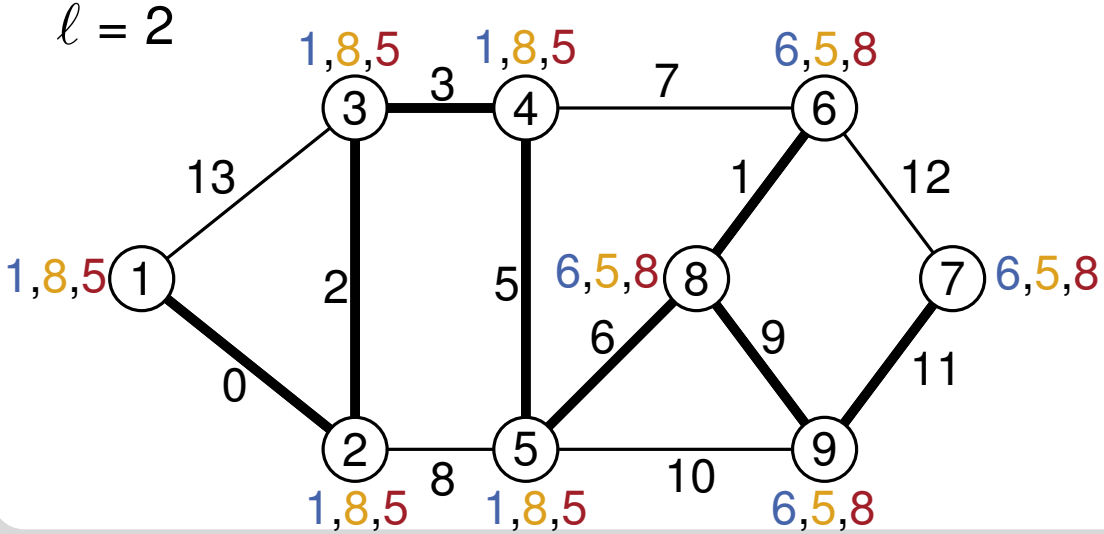
Problemstellung
gegeben: Graph $G = (V, E)$, $V = \{1, \ldots, n\}$ und Kantengewichte $c_{i,j}$ für $\{i, j\} \in 2^V$, wobei $c_{i,j} = \infty$ wenn $\{i, j\} \notin E$.
gesucht: Spannbaum in G mit minimalen Gewicht.

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 4. Schritt:

Für alle i : $1 \leq i \leq n$ **führe parallel aus**
wenn $S[i] \neq 0$ *und* $K[i] = i$ *und* $c_{N[i],S[i]} \neq \infty$ **dann**
setze $T := T \cup \{N[i], S[i]\}$

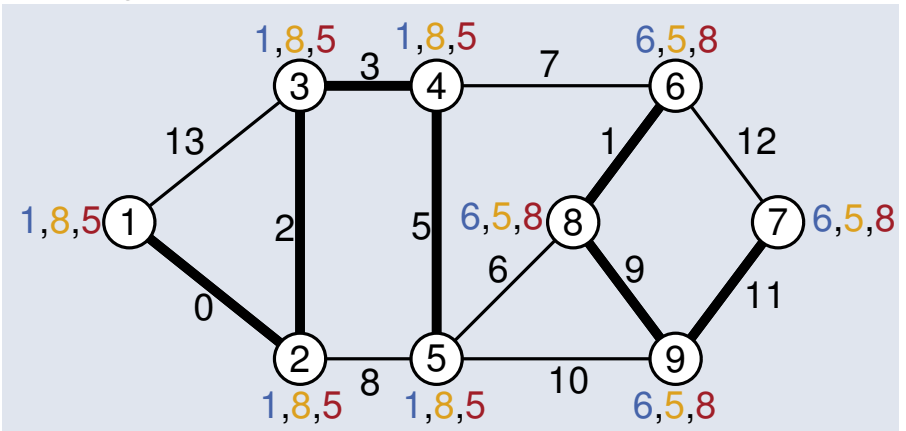
wenn $S[i] \neq 0$ **dann**
setze $K[i] = K[N[i]]$

$\ell = 2$



Vorheriger Schritt:

Legende: $K[i]$,$N[i]$,$S[i]$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Minimaler Spannbaum (MST)

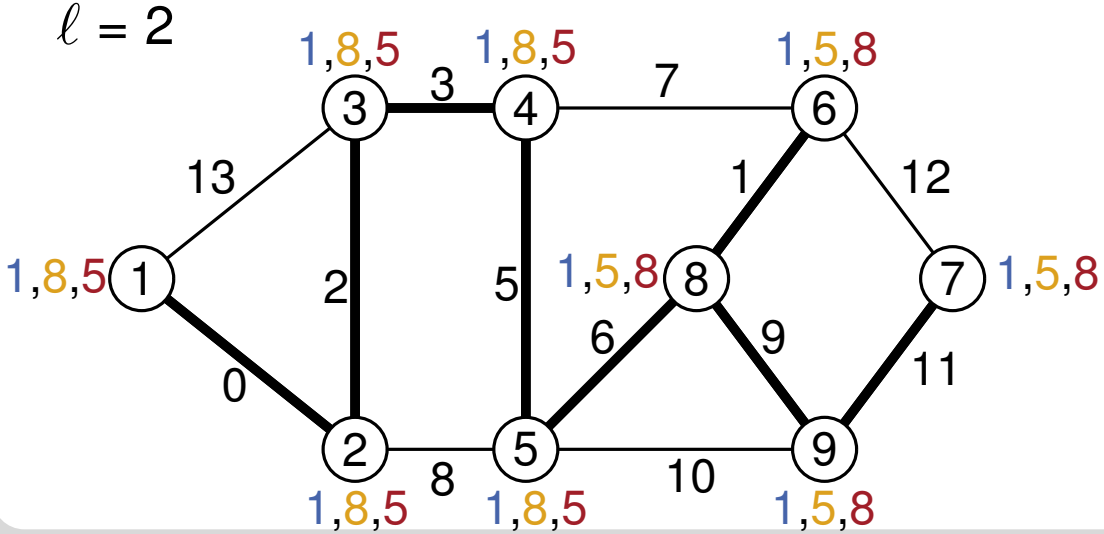
Problemstellung
gegeben: Graph $G = (V, E)$, $V = \{1, \ldots, n\}$ und Kantengewichte $c_{i,j}$ für $\{i, j\} \in 2^V$, wobei $c_{i,j} = \infty$ wenn $\{i, j\} \notin E$.
gesucht: Spannbaum in G mit minimalen Gewicht.

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 5. Schritt:

für $m = 1$ *bis* $\lceil \log n \rceil$ **tue**
Für alle i : $1 \leq i \leq n$ **führe parallel aus**
$K[i] := K[K[i]]$

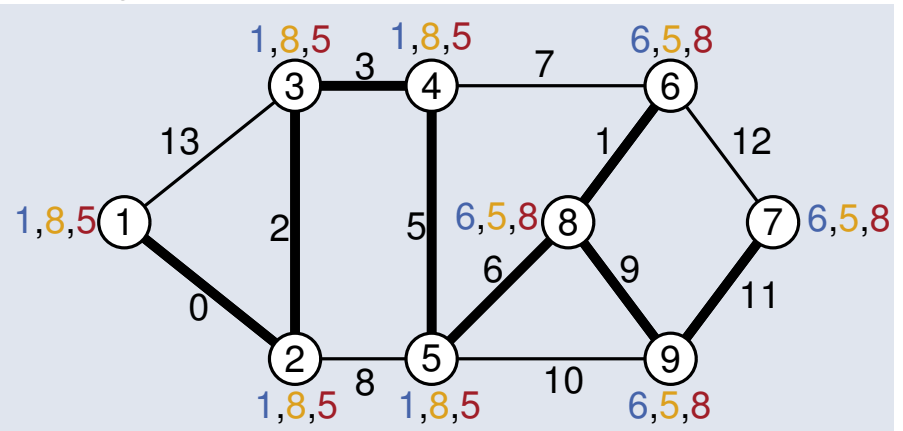
} ListRank auf Wurzelbäumen

$\ell = 2$



Vorheriger Schritt:

Legende: $K[i], N[i], S[i]$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Gesamter Algorithmus für MST

Für alle $i: 1 \leq i \leq n$ **führe parallel aus**
  $K[i] := i$

für $\ell = 1$ **bis** $\lceil \log n \rceil$ **tue**

  Für alle $i: 1 \leq i \leq n$ **führe parallel aus** — Schritt 1
    Finde $k \in V$ mit $K[i] \neq K[k]$ und $c_{ik} = \min\{c_{ij} : 1 \leq j \leq n, K[i] \neq K[j]\}$
    $N[i] := k$

  Für alle $i: 1 \leq i \leq n$ **führe parallel aus** — Schritt 2
    Finde $t \in V$ mit $K[i] = K[t]$ und $c_{tN[t]} = \min\{c_{jN[j]} : 1 \leq j \leq n, K[i] = K[j]\}$
    setze $N[i] := N[t]$ und $S[i] := t$

  Für alle $i: 1 \leq i \leq n$ **führe parallel aus** — Schritt 3
    wenn $N[N[i]] = S[i]$ *und* $K[i] < K[N[i]]$ **dann**
      setze $S[i] := 0$

  Für alle $i: 1 \leq i \leq n$ **führe parallel aus** — Schritt 4
    wenn $S[i] \neq 0$ *und* $K[i] = i$ *und* $c_{N[i],S[i]} \neq \infty$ **dann**
      setze $T := T \cup \{N[i], S[i]\}$
    wenn $S[i] \neq 0$ **dann**
      setze $K[i] = K[N[i]]$

  für $m = 1$ *bis* $\lceil \log n \rceil$ **tue** — Schritt 5
    Für alle $i: 1 \leq i \leq n$ **führe parallel aus**
      $K[i] := K[K[i]]$

1. Laufzeitanalyse wie für Zusammenhangskomponenten: $O(\log^2(n))$ Zeit und n^2 Prozessoren.
2. Prozessorenzahl kann durch Rescheudling auf $\left\lceil \frac{n^2}{\log n} \right\rceil$ reduziert werden.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Reduzierung der Prozessorenanzahl

Anzahl Prozessoren kann sogar auf $\lceil \frac{n^2}{\log^2 n} \rceil$ reduziert werden.

für $\ell = 1$ **bis** $\lceil \log n \rceil$ **tue**

Für alle $i: 1 \leq i \leq n$ **führe parallel aus** — Schritt 1
- Finde $k \in V$ mit $K[i] \neq K[k]$ und $c_{ik} = \min\{c_{ij} : 1 \leq j \leq n, K[i] \neq K[j]\}$
- $N[i] := k$

Für alle $i: 1 \leq i \leq n$ **führe parallel aus** — Schritt 2
- Finde $t \in V$ mit $K[i] = K[t]$ und $c_{tN[t]} = \min\{c_{jN[j]} : 1 \leq j \leq n, K[i] = K[j]\}$
- setze $N[i] := N[t]$ und $S[i] := t$

Für alle $i: 1 \leq i \leq n$ **führe parallel aus** — Schritt 3
- **wenn** $N[N[i]] = S[i]$ *und* $K[i] < K[N[i]]$ **dann**
  - setze $S[i] := 0$

Für alle $i: 1 \leq i \leq n$ **führe parallel aus** — Schritt 4
- **wenn** $S[i] \neq 0$ *und* $K[i] = i$ *und* $c_{N[i],S[i]} \neq \infty$ **dann**
  - setze $T := T \cup \{N[i], S[i]\}$
- **wenn** $S[i] \neq 0$ **dann**
  - setze $K[i] = K[N[i]]$

für $m = 1$ *bis* $\lceil \log n \rceil$ **tue** — Schritt 5
- **Für alle** $i: 1 \leq i \leq n$ **führe parallel aus**
  - $K[i] := K[K[i]]$

Idee: Betrachte nicht alle Knoten, sondern nur Repräsentanten von Komponenten.
Vorteil: Anzahl der Komponenten halbiert sich mindestens in jeder Phase.

Vorgehen:

1. Führe Minimumsbildung in Schritt 1 und Schritt 2 mit $\lceil \frac{n}{\log^2 n} \rceil$ Prozessoren aus.
2. Nach jeder Phase wird die Adjazenzmatrix des Graphen neu aufgestellt, d.h. für alle Knoten i und j wird die Kante $\{i, j\}$ (falls vorhanden) auf die Repräsentanten der Komponenten von i bzw. j übertragen.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Reduzierung der Prozessorenanzahl

1. Führe Minimumsbildung in Schritt 1 und Schritt 2 mit $\lceil \frac{n}{\log^2 n} \rceil$ Prozessoren aus.

Lemma: Gegeben seien n Werte $a_1, \ldots, a_n$. Die Summe (bzw. Produkt, Minimum, etc) aus $a_1, \ldots, a_n$ kann mit K Prozessoren in Zeit $O(t(n))$ bestimmt werden, wobei

$$t(n) \leq \begin{cases} \lceil n/K \rceil - 1 + \log K, & \text{falls } \lceil n/2 \rceil > K \\ \log n, & \text{sonst.} \end{cases}$$

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 1. Schritt:

Für alle i : $1 \leq i \leq n$ **führe parallel aus**

Finde $k \in V$ mit $K[i] \neq K[k]$ und $c_{ik} = \min\{c_{ij} : 1 \leq j \leq n, K[i] \neq K[j]\}$

$N[i] := k$

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 2. Schritt:

Finde $t \in V$ mit $K[i] = K[t]$ und $c_{tN[t]} = \min\{c_{jN[j]} : 1 \leq j \leq n, K[i] = K[j]\}$

setze $N[i] := N[t]$ und $S[i] := t$

Reduzierung der Prozessorenanzahl

1. Führe Minimumsbildung in Schritt 1 und Schritt 2 mit $\lceil \frac{n}{\log^2 n} \rceil$ Prozessoren aus.

Lemma: Gegeben seien n Werte $a_1, \ldots, a_n$. Die Summe (bzw. Produkt, Minimum, etc) aus $a_1, \ldots, a_n$ kann mit K Prozessoren in Zeit $O(t(n))$ bestimmt werden, wobei

$$t(n) \leq \begin{cases} \lceil n/K \rceil - 1 + \log K, & \text{falls } \lceil n/2 \rceil > K \\ \log n, & \text{sonst.} \end{cases}$$

Lemma: Schritt 1 und Schritt 2 können mithilfe von $n \cdot K$ Prozessoren in Gesamtlaufzeit (über alle Phasen)

$$O(\frac{n}{K} + \log n \cdot \log K)$$

ausgeführt werden.

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 1. Schritt:

Für alle i : $1 \leq i \leq n$ **führe parallel aus**

Finde $k \in V$ mit $K[i] \neq K[k]$ und $c_{ik} = \min\{c_{ij} : 1 \leq j \leq n, K[i] \neq K[j]\}$

$N[i] := k$

Phase ℓ mit $1 \leq \ell \leq \lceil \log n \rceil$, 2. Schritt:

Finde $t \in V$ mit $K[i] = K[t]$ und $c_{tN[t]} = \min\{c_{jN[j]} : 1 \leq j \leq n, K[i] = K[j]\}$

setze $N[i] := N[t]$ und $S[i] := t$

Reduzierung der Prozessorenanzahl

2. Nach jeder Phase wird die Adjazenzmatrix des Graphen neu aufgestellt, d.h. für alle Knoten i und j wird die Kante $\{i, j\}$ (falls vorhanden) auf die Repräsentanten der Komponenten von i bzw. j übertragen.

Lemma: Die Gesamtlaufzeit (über alle Phasen) der Neuberechnung der Adjazenzmatrix beträgt

$$O(\frac{n}{K} + \log n \cdot \log K)$$

Zeit, wenn $n \cdot K$ Prozessoren verwendet werden.

Beweisskizze:

- Ordne die Knoten der Komponenten entsprechend des Index ihrer Repräsentanten in Gruppen an.
- Schreibe an Anfang jeder solchen Knotengruppe den Repräsentanten.
- Zum Ermitteln der Einträge für alle Repräsentantenpaare muss jeweils fur jede Knotengruppe die ODER-Verbindung der entsprechenden Einträge berechnet werden.
- Ordne jeder Komponente $K = \lceil \frac{n}{\log^2 n} \rceil$ Prozessoren zu, um die ODER-Verbindungen für alle Adjazenzen von Knoten der Komponenten zu bilden.
- Adjazenzen können für Knotengruppe in $O(\lceil \frac{n}{K} \rceil + \log K)$ Zeit berechnet werden. (Binärverbinungen ein partitionierten Menge.)

Zusammenfassung

Lemma: Schritt 1 und Schritt 2 können mithilfe von $n \cdot K$ Prozessoren in Gesamtlaufzeit (über alle Phasen)

$$O(\frac{n}{K} + \log n \cdot \log K)$$

ausgeführt werden.

Lemma: Die Gesamtlaufzeit (über alle Phasen) der Neuberechnung der Adjazenzmatrix beträgt

$$O(\frac{n}{K} + \log n \cdot \log K)$$

Zeit wenn $n \cdot K$ Prozessoren verwendet werden.

Für $\frac{n^2}{\log^2 n}$ Prozessoren ergibt sich damit eine Gesamtlaufzeit von $O(\log^2 n)$ für die Bestimmung eines MST. Damit betragen die Kosten $O(n^2)$:

- bzgl. sequentiellen Algorithmen für MST, die auf Adjazenzmatrix des Graphen arbeiten, ist dies kostenoptimal.
- bzgl. beliebigen sequentiellen Algorithmen für MST ist dies nicht kostenoptimal.