Untersuchung der Skalierbarkeit von parallelem Sortieren auf einem Multicore-Prozessor

Verteidigung der Bachelorarbeit

Leon Zoerner

Informatik

15. Februar 2026

Agenda

1. Einleitung
2. Theoretische Grundlagen
3. Methodik und Versuchsaufbau
4. Ergebnisse und Analyse
5. Diskussion und Fazit
6. Binärbaum
7. Formeln
8. Herleitung T(n,p)

Einleitung: Motivation

Motivation: Die Motivation dieser Arbeit war es, herauszufinden, wie sehr man mit Threads an die theoretisch erwartete Laufzeitverbesserung herankommen kann und welche Strategie dafür am besten geeignet ist. Da sich für diese Untersuchungen ein geeigneter, leicht verständlicher und programmierbarer Anwendungsfall anbietet, werden Sortieralgorithmen betrachtet, die sich zudem sehr gut parallelisieren lassen. Dazu kommt, dass ich Recherche nach Möglichkeit vermeiden wollte. Daher war es naheliegend, selbst Code zu schreiben und diesen zu analysieren, da man hierfür weniger recherchieren muss.

Einleitung: Zielsetzung und Forschungsfrage

Zielsetzung und Forschungsfrage: Ziel dieser Bachelorarbeit ist die systematische Analyse der Laufzeitentwicklung paralleler Sortierverfahren. Dabei soll untersucht werden, wie sich parallele Implementierungen von Quicksort und Mergesort im Vergleich zu ihren sequentiellen Varianten verhalten. Im Fokus stehen insbesondere folgende Punkte:

- der Einfluss verschiedener Threadingstrategien auf die Laufzeit,
- die Frage, ab welcher Eingabegröße und bei welcher Anzahl von Threads ein messbarer Geschwindigkeitsvorteil entsteht,
- sowie die Identifikation von Thread-Management-Techniken, die für Sortieralgorithmen die besten Laufzeiten erzielen.

Aus diesen Aspekten ergibt sich die zentrale Forschungsfrage dieser Arbeit:
Unter welchen Bedingungen liefern parallele Sortieralgorithmen anhand von Quicksort und Mergesort einen signifikanten Laufzeitvorteil gegenüber der sequentiellen Ausführung, und welche Threadingstrategien führen dabei zur besten Laufzeit?

Grundlagen der Parallelisierung

Messumgebung und Implementierung

Implementierungsvarianten Code

Laufzeitmessungen (sequenziell)

Analyse der Threading-Methoden

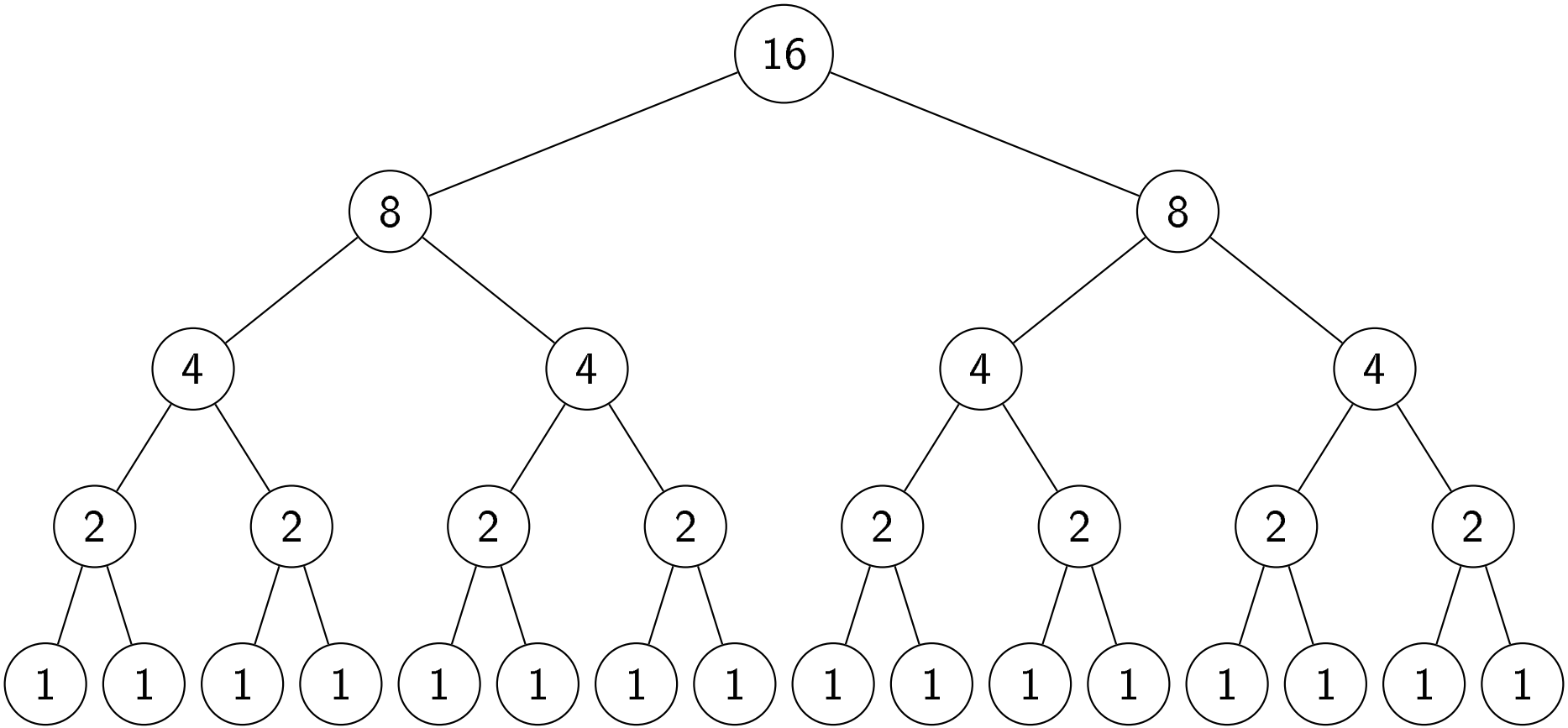
Beantwortung der Forschungsfrage

Zusammenfassung und Ausblick

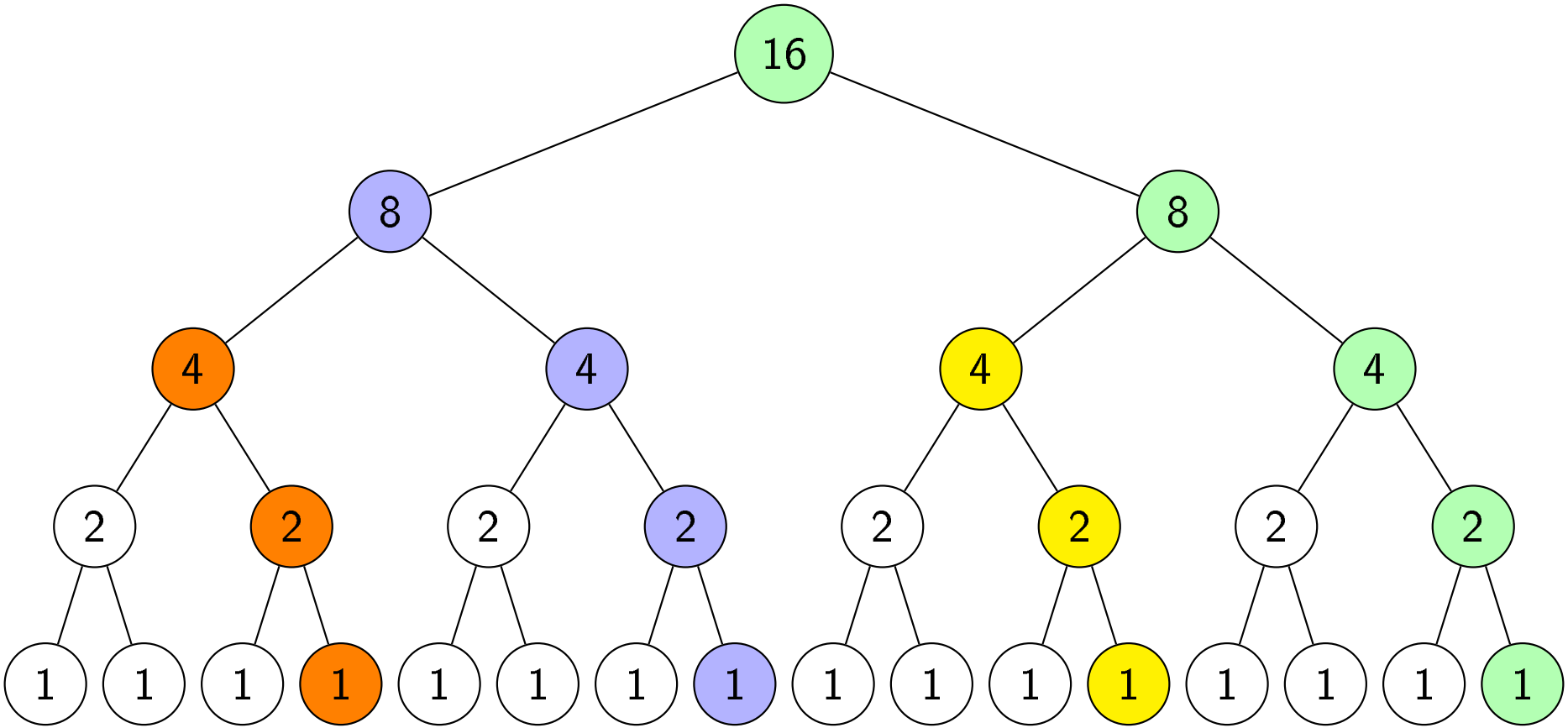
Vielen Dank für Ihre Aufmerksamkeit!

Fragen und Diskussion

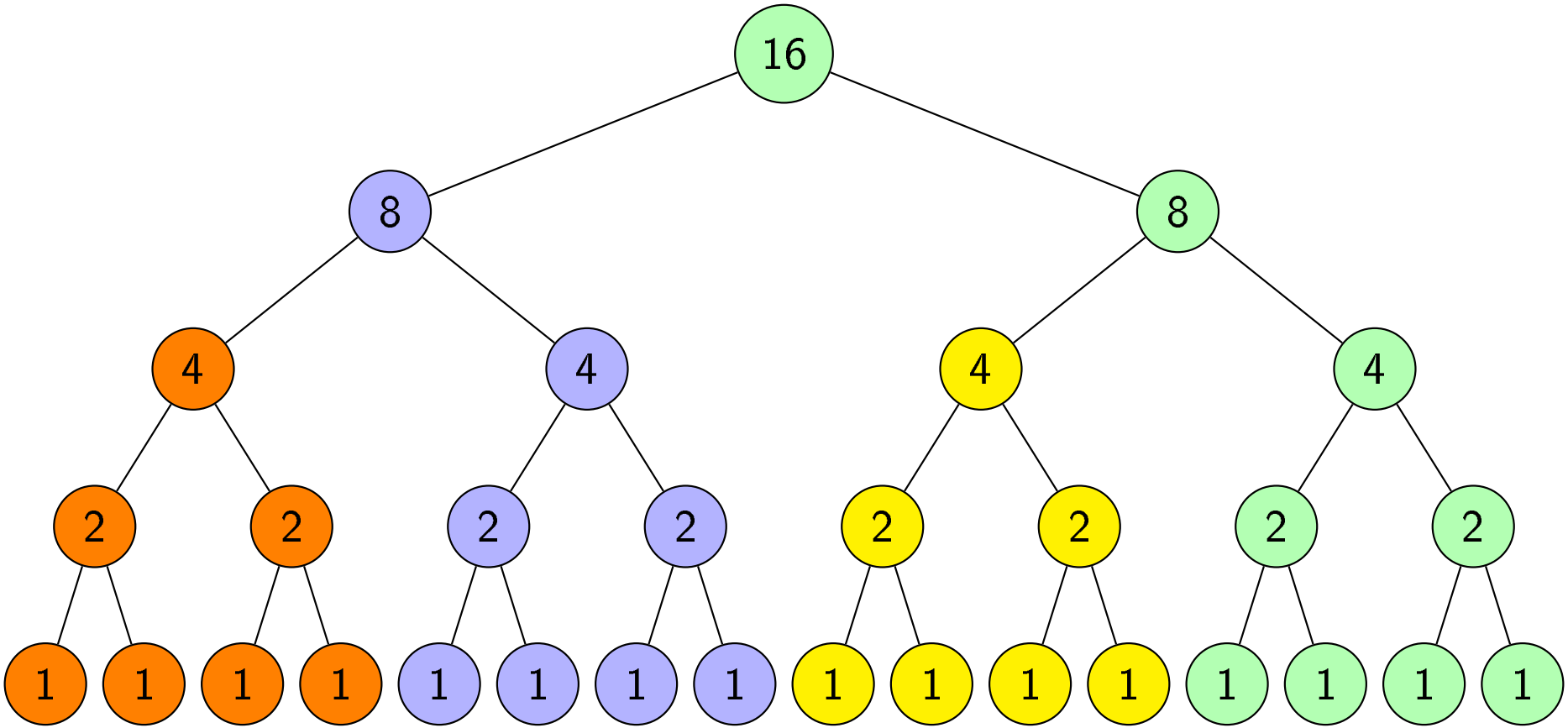
Balancierter Binärbaum für $n = 16$



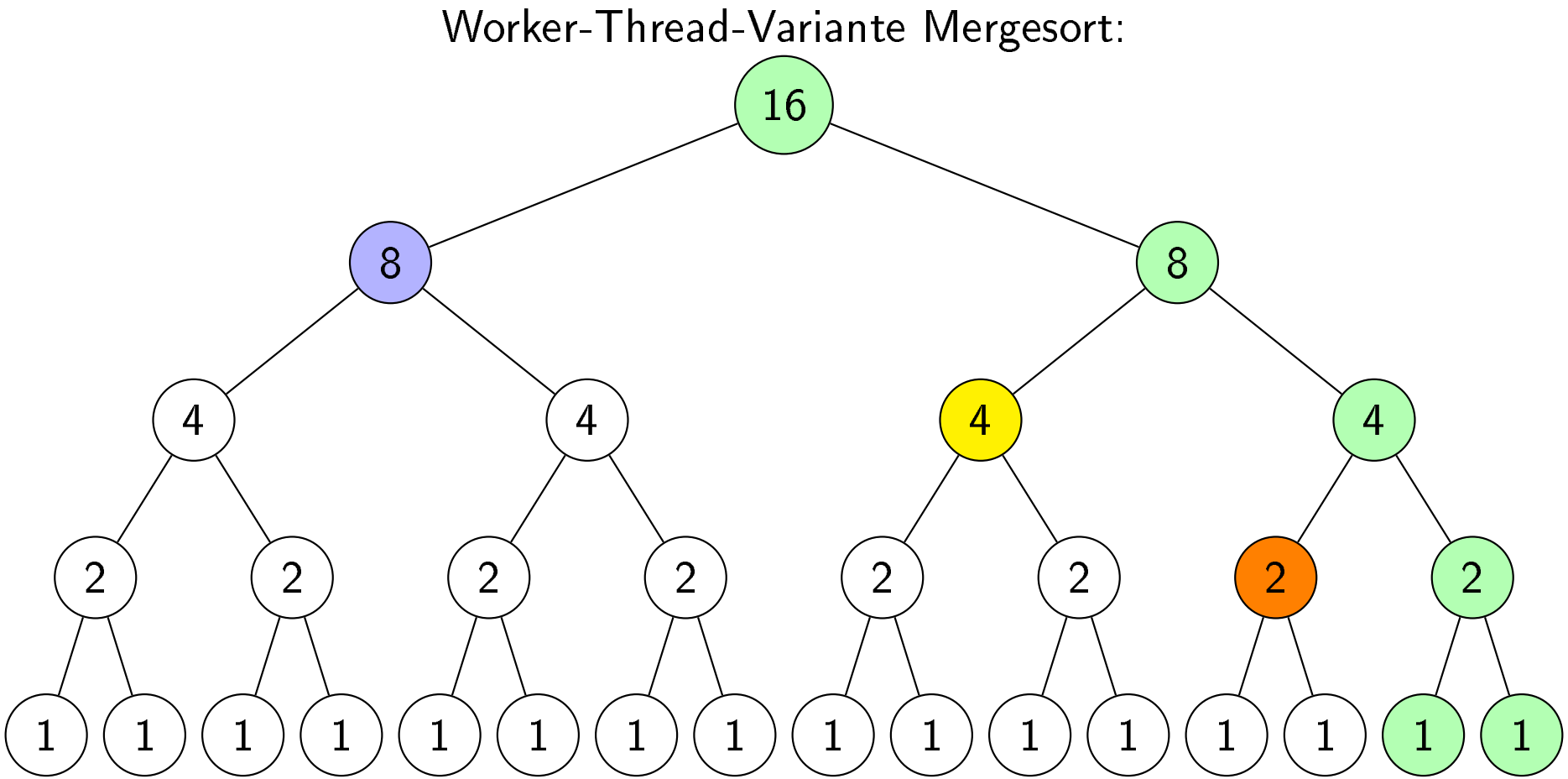
Balancierter Binärbaum für $n = 16$, $p = 4$, $e = 2$



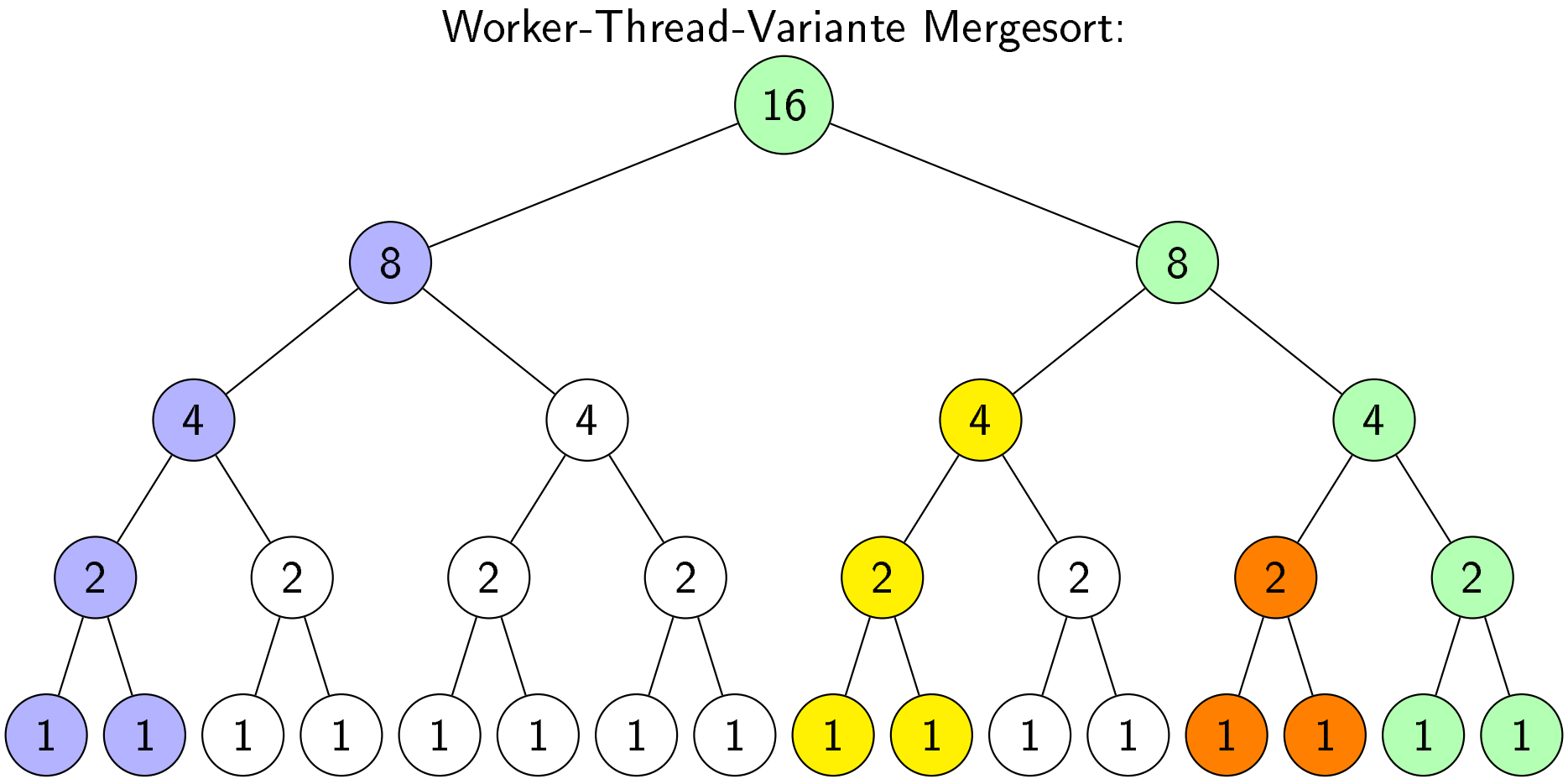
Balancierter Binärbaum für $n = 16$, $p = 4$, $e = 2$



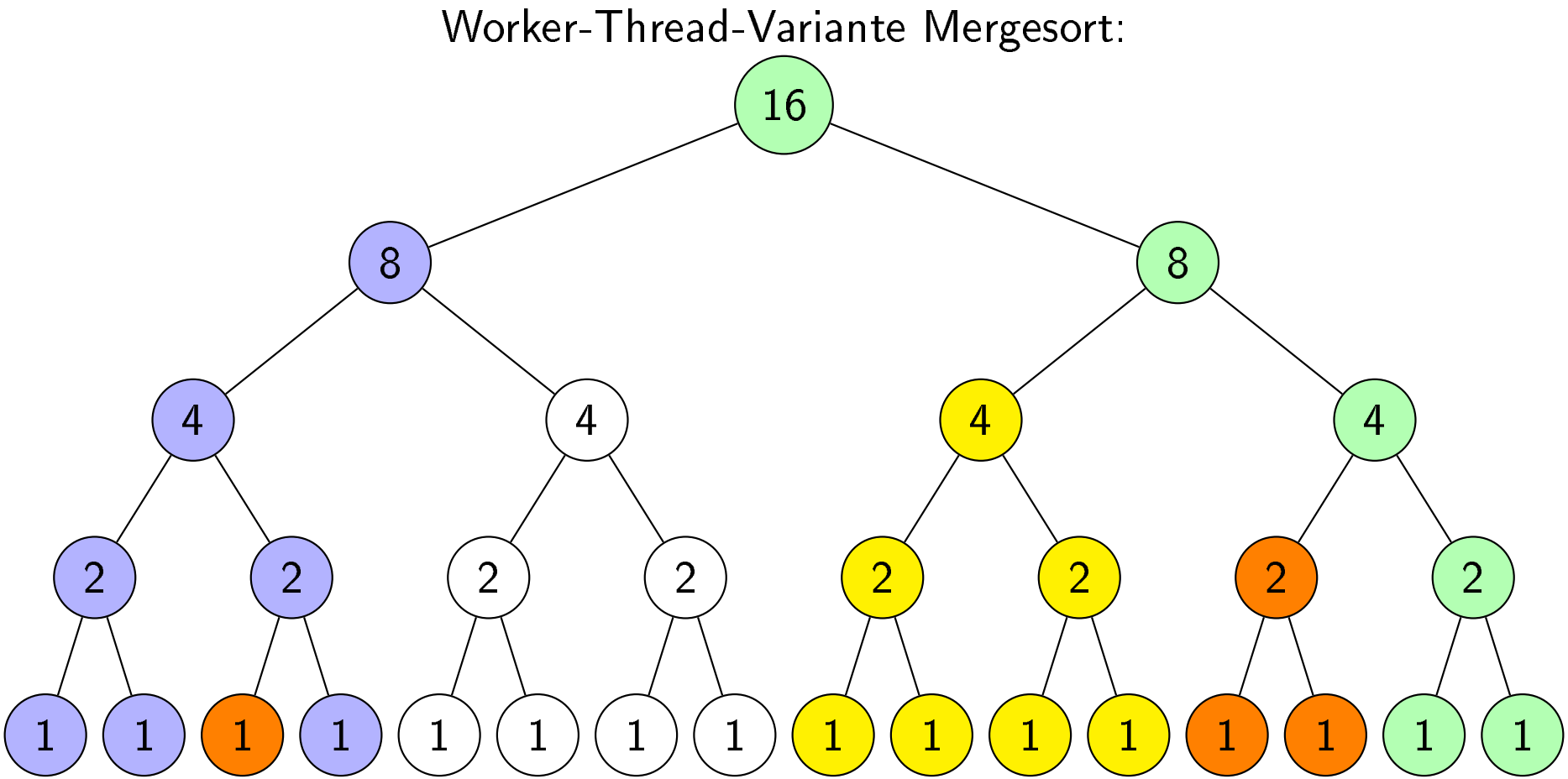
Balancierter Binärbaum für $n = 16$, $p = 4$



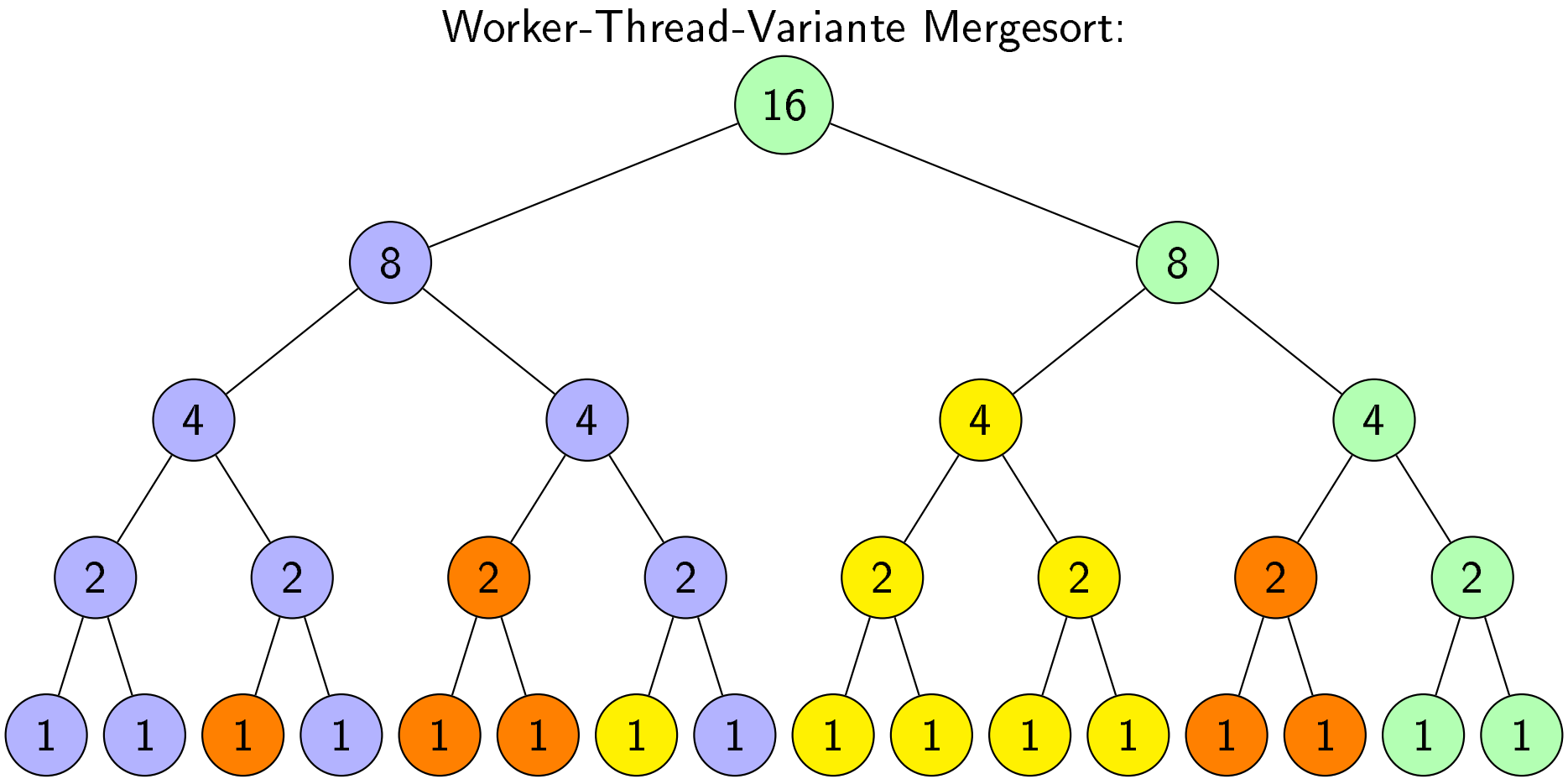
Balancierter Binärbaum für $n = 16$, $p = 4$



Balancierter Binärbaum für $n = 16$, $p = 4$

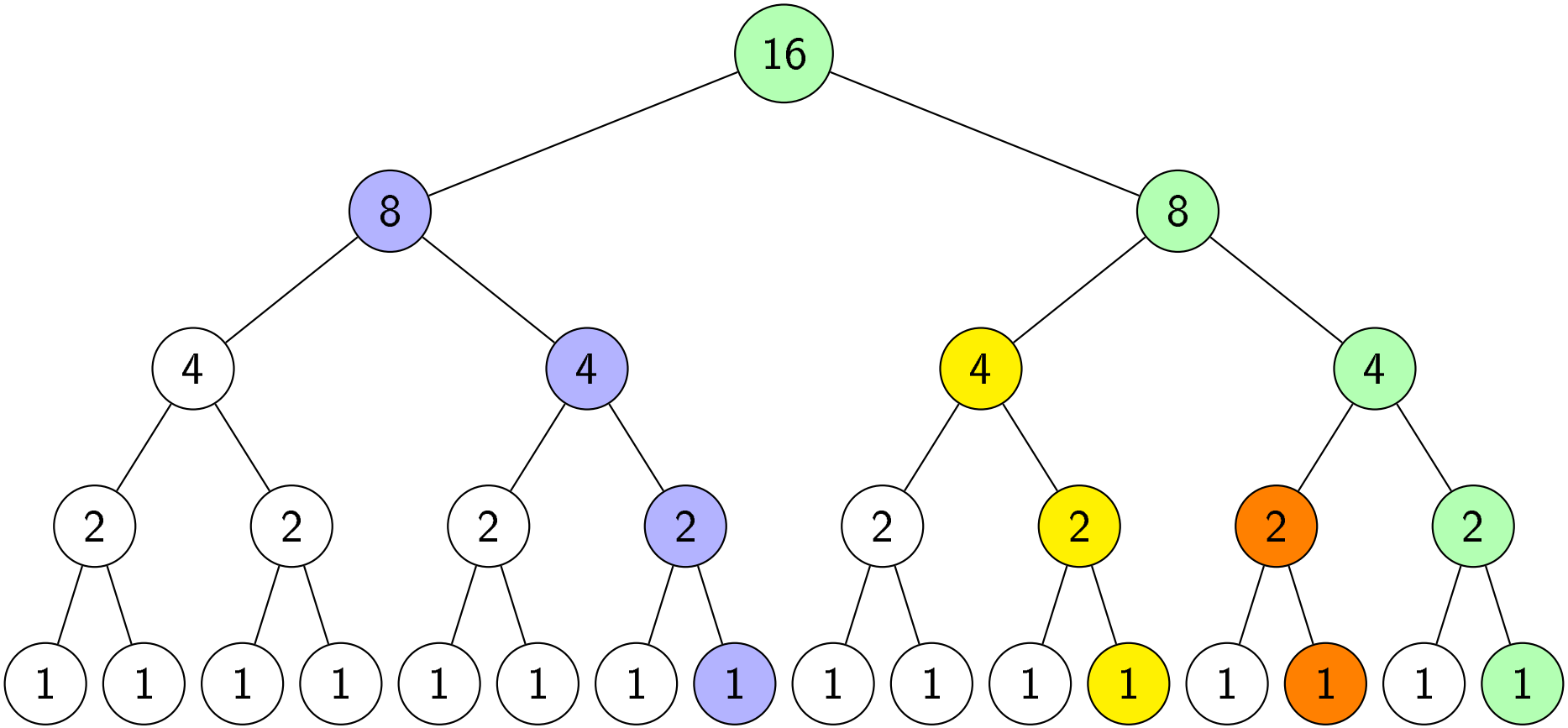


Balancierter Binärbaum für $n = 16$, $p = 4$



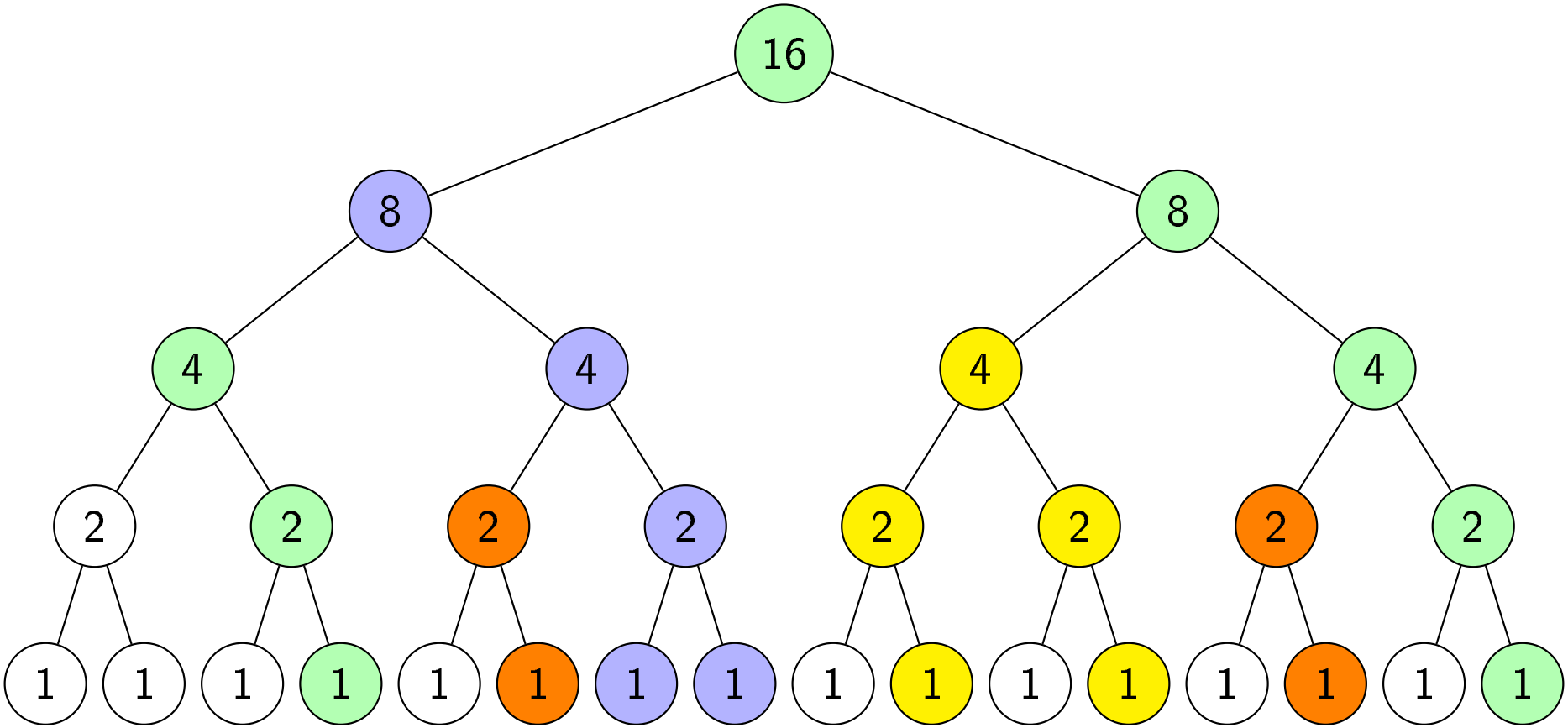
Balancierter Binärbaum für $n = 16$, $p = 4$

Worker-Thread-Variante Quicksort **Best-Case**:



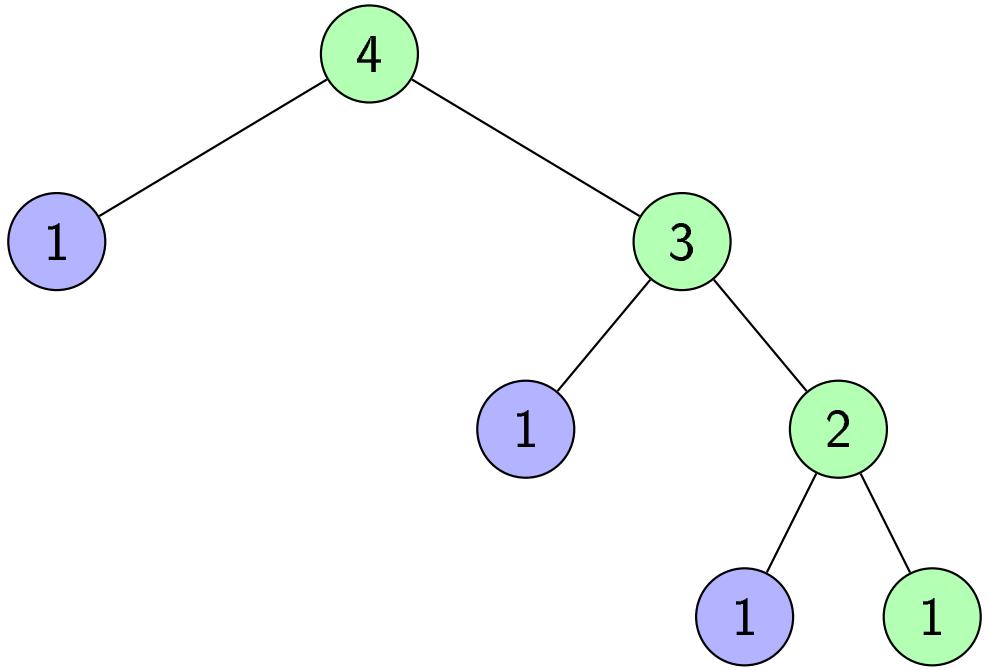
Balancierter Binärbaum für $n = 16$, $p = 4$

Worker-Thread-Variante Quicksort **Best-Case**:



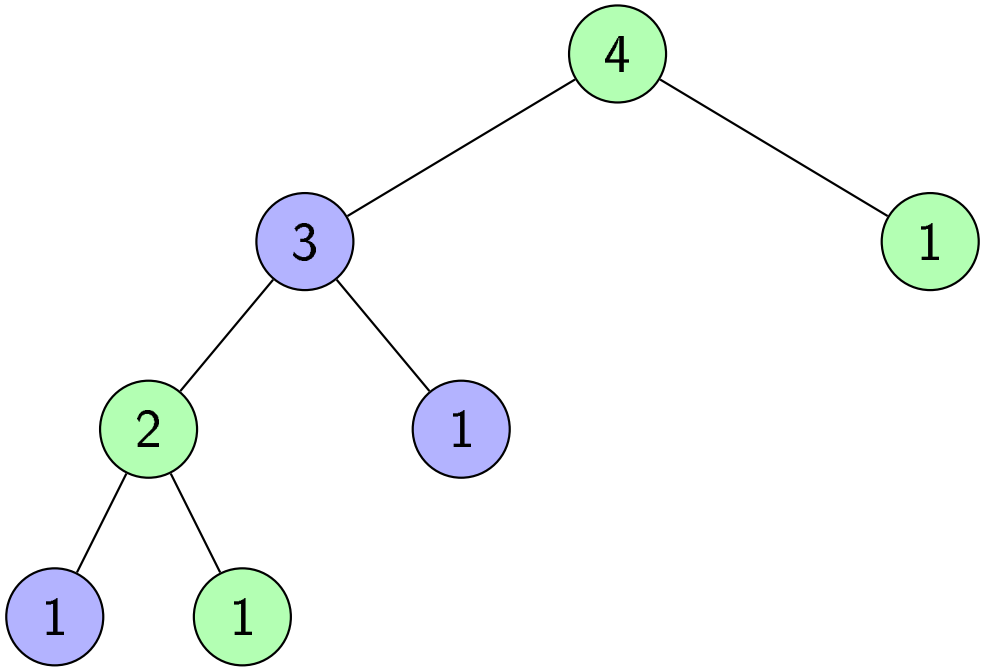
Maximal unbalancierter Binärbaum für $n = 4$, $p = 2$

Worker-Thread-Variante Quicksort **Worst-Case**:



Maximal unbalancierter Binärbaum für $n = 4$, $p = 2$

Worker-Thread-Variante Quicksort **Worst-Case**:



Formeln: Mergesort

$$
\begin{aligned}
T(n) &= m_1 + m_2 + n \\
T(n) &= T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + n \\
T(n) &= 2 \cdot T\left(\frac{n}{2}\right) + n, \\
T(n) &= n \cdot \log_2(n) + n, \\
O(T(n)) &= O(n \log n).
\end{aligned}
$$

Formeln: Quicksort

Best-Case von Quicksort:

$$
\begin{aligned}
T(n) &= q_1 + q_2 + n \\
T(n) &= T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + n \\
T(n) &= 2 \cdot T\left(\frac{n}{2}\right) + n, \\
T(n) &= n \cdot \log_2(n) + n, \\
O(T(n)) &= O(n \log n).
\end{aligned}
$$

Formeln: Quicksort

Der **Worst-Case** von Quicksort ist:

$$
\begin{aligned}
T(n) &= q_1 + q_2 + n \\
T(n) &= T(n-1) + 1 + n, \\
T(n) &= \frac{1}{2} \cdot (n^2 + n) + n, \\
O(T(n)) &= O(n^2).
\end{aligned}
$$

Der **heuristisch betrachtete Average-Case** von Quicksort ist:

$$
\begin{aligned}
T(n) &= q_1 + q_2 + n \\
q_1 &= T\left(\frac{1 + \cdots + (n-1)}{n-1}\right) = T\left(n \cdot \frac{n-1}{2} \cdot \frac{1}{n-1}\right) = T\left(\frac{n}{2}\right) = q_2 \\
T(n) &= 2 \cdot T\left(\frac{n}{2}\right) + n \\
T(n) &= n \cdot \log_2(n) + n \\
O(T(n)) &= O(n \log n).
\end{aligned}
$$

Formeln: parallel Mergesort

$$p = \text{Thread-Anzahl}$$

$$e = \log_2(p)$$

$$T(n, e) = \begin{cases} 2 \cdot T\left(\frac{n}{2}, 0\right) + n & , \text{wenn } e = 0 \\ 1 \cdot T\left(\frac{n}{2}, e-1\right) + n & , \text{wenn } e > 0 \end{cases}$$

$$T(n, p) = 2n\left(1 - \frac{1}{p}\right) + \frac{n}{p} \cdot \log_2\left(\frac{n}{p}\right) + \frac{n}{p}$$

$$O(T(n, p)) = O\left(\frac{n}{p} \cdot \log_2(n) + n\right)$$

$$e_{\max} = \log_2(n)$$

$$p_{\max} = n$$

Formeln: parallel Quicksort

Der **Best-Case** und **heuristisch betrachtete Average-Case** von Quicksort ist:

$$p = \text{Thread-Anzahl}$$

$$e = \log_2(p)$$

$$T(n, e) = \begin{cases} 2 \cdot T\left(\frac{n}{2}, 0\right) + n & , \text{wenn } e = 0 \\ 1 \cdot T\left(\frac{n}{2}, e-1\right) + n & , \text{wenn } e > 0 \end{cases}$$

$$T(n, p) = 2n\left(1 - \frac{1}{p}\right) + \frac{n}{p} \cdot \log_2\left(\frac{n}{p}\right) + \frac{n}{p}$$

$$O(T(n, p)) = O\left(\frac{n}{p} \cdot \log_2(n) + n\right)$$

$$e_{\max} = \log_2(n)$$

$$p_{\max} = n$$

Formeln: parallel Quicksort

Der **Worst-Case** von Quicksort (Worker-Thread-Variante) bei $p > 1$ ist:

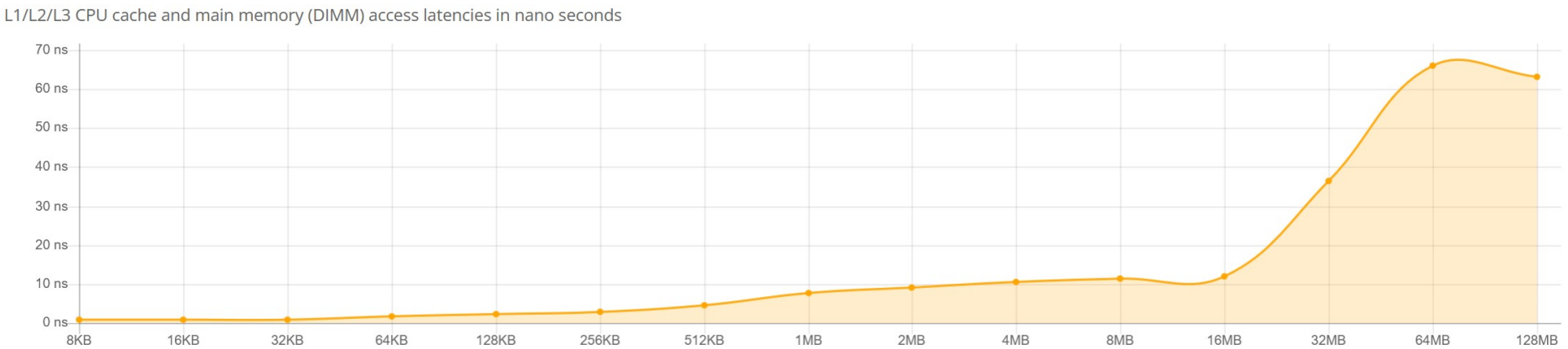
$$
\begin{aligned}
T(n) &= q_2 + n, \\
T(n) &= T(n-1) + n, \\
T(n) &= \frac{1}{2} \cdot (n^2 + n), \\
O(T(n)) &= O(n^2).
\end{aligned}
$$

Formeln: Einheiten und Skalierung

$$x \cdot T(n) = 2 \cdot T\left(\frac{n}{2}\right) + x \cdot n$$
$$x \cdot T(n) = x \cdot n \cdot \log_2(n) + x \cdot n$$
$$x \cdot T(n) = x \cdot (n \cdot \log_2(n) + n)$$

Aufgrund von Overheads wie z. B. erhöhte Speicherlatenzen durch Cache-Misses:

$$x = f(n, p)$$
$$f(n, p) \leq f(n+1, p)$$
$$f(n, p) \leq f(n, p+1)$$



Nebeneffekt des Skalierens mit dem Faktor x

Sollte die Standard-Formel

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

falsch sein und stattdessen diese Formel gelten

$$T(n) = 2T\left(\frac{n}{2}\right) + (n-1)$$

kann man diese auch schreiben als

$$T(n) = 2T\left(\frac{n}{2}\right) + \underbrace{\left(1 - \frac{1}{n}\right)}_{x} \cdot n$$

Dabei ist $\left(1 - \frac{1}{n}\right)$ implizit auch durch x repräsentiert. Durch diesen Nebeneffekt korrigiert der Faktor x auch gleich implizit die Formel. Dies macht die Formel nach dem Hochskalieren auf die sequentielle Laufzeit so präzise und sorgt damit dafür, dass die T(n,p)-Formel so exakt die untere Grenze der Laufzeitverbesserung vorhersagen kann.

Herleitung T(n,p): Ausgangsdefinition

Gegeben ist die Rekursion:

$$T(n,e) = \begin{cases} 2\,T\left(\frac{n}{2},0\right) + n & \text{falls } e = 0 \\ T\left(\frac{n}{2},e-1\right) + n & \text{falls } e > 0 \end{cases}$$

Zusätzlich gilt:

$$p = 2^e$$

Herleitung T(n,p): Entfaltung der Rekursion (1. Schritt)

Für $e > 0$ gilt:

$$T(n, e) = T\left(\frac{n}{2}, e - 1\right) + n$$

Ein Einsetzen ergibt:

$$T(n, e) = T\left(\frac{n}{4}, e - 2\right) + \frac{n}{2} + n$$

Noch ein weiteres Einsetzen:

$$T(n, e) = T\left(\frac{n}{8}, e - 3\right) + \frac{n}{4} + \frac{n}{2} + n$$

Nach e Schritten ergibt sich:

$$T(n, e) = T\left(\frac{n}{2^e}, 0\right) + n\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^{e-1}}\right)$$

Herleitung T(n,p): Auswertung der geometrischen Reihe

Die Summe

$$\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^{e-1}}\right) = \sum_{i=0}^{e-1} \frac{1}{2^i}$$

ist eine geometrische Reihe mit Quotient $\frac{1}{2}$.
Sie ergibt:

$$\sum_{i=0}^{e-1} \frac{1}{2^i} = 2\left(1 - \frac{1}{2^e}\right)$$

Damit folgt:

$$T(n, e) = T\left(\frac{n}{2^e}, 0\right) + 2n\left(1 - \frac{1}{2^e}\right)$$

Herleitung T(n,p): Einsetzen des Basisfalls

Für $e = 0$ gilt:

$$T(n, 0) = 2T\left(\frac{n}{2}, 0\right) + n$$

Dies entspricht der Rekursion von sequenziellem Mergesort.
Bekannt ist:

$$T(n, 0) = n \log_2 n + n$$

Ersetzen von n durch $\frac{n}{2^e}$ ergibt:

$$T\left(\frac{n}{2^e}, 0\right) = \frac{n}{2^e} \log_2\left(\frac{n}{2^e}\right) + \frac{n}{2^e}$$

Einsetzen in $T(n, e)$ liefert:

$$T(n, e) = \frac{n}{2^e} \log_2\left(\frac{n}{2^e}\right) + \frac{n}{2^e} + 2n\left(1 - \frac{1}{2^e}\right)$$

Herleitung T(n,p): Substitution $p = 2^e$

Da gilt:

$$p = 2^e$$

folgt:

$$\frac{n}{2^e} = \frac{n}{p}$$

Damit ergibt sich:

$$T(n,p) = \frac{n}{p}\log_2\left(\frac{n}{p}\right) + \frac{n}{p} + 2n\left(1 - \frac{1}{p}\right)$$

Umsortiert erhält man:

$$T(n,p) = 2n\left(1 - \frac{1}{p}\right) + \frac{n}{p}\log_2\left(\frac{n}{p}\right) + \frac{n}{p}$$