Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. Peter Sanders
Dr. Thomas Worsch, Dr. Simon Gog
Demian Hespe, Yaroslav Akhremstev

5. Übungsblatt zu Algorithmen II im WS 2017/2018

http://algo2.iti.kit.edu/AlgorithmenII_WS17.php
{hespe,sanders,simon.gog,worsch,yaroslav.akhremtsev}@kit.edu

Weihnachtsblatt

Aufgabe 1 (*Pflichtaufgabe (*)*)

a) Machen Sie Ihren Übungsleitern ein schönes Weihnachtsgeschenk.

Aufgabe 2 (*Kleinaufgaben: Parallele Algorithmen*)

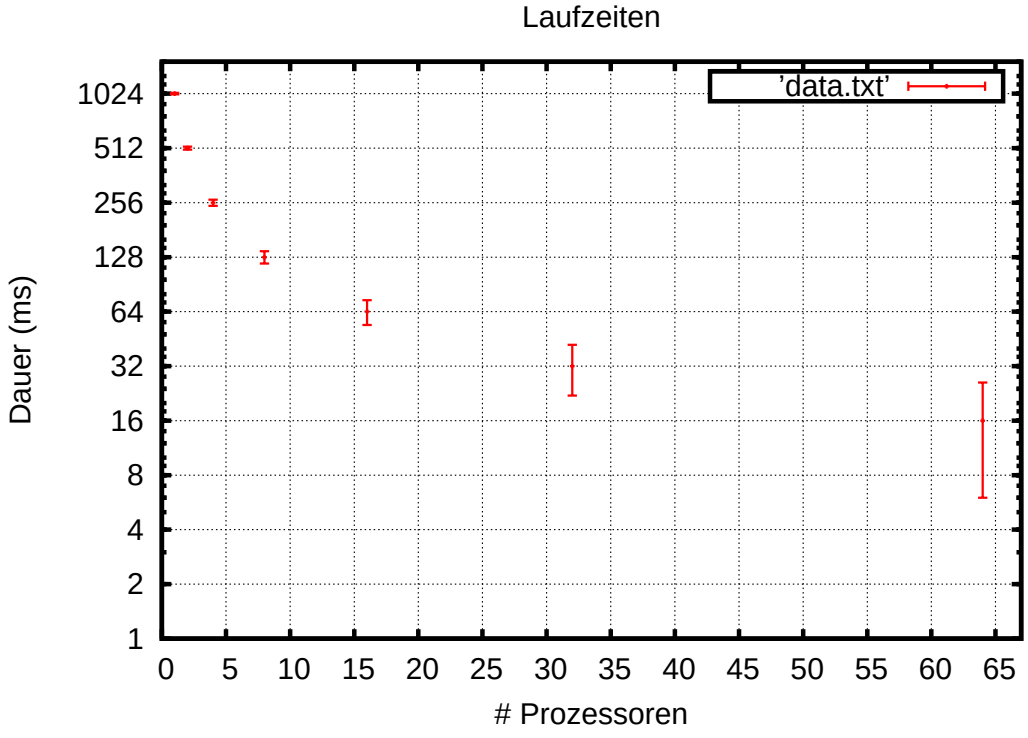
a) Gegeben sei ein paralleler vergleichsbasierter Sortieralgorithmus zum Sortieren von n Objekten auf p Prozessoren mit einer Laufzeit von

$$T(p) := \Theta(\frac{n^2 \log^2 n}{p^2}).$$

Geben Sie den absoluten *speed-up* und die *efficiency* an.

b) Wie muss in der vorherigen Teilaufgabe die Prozessorzahl p mit der Eingabegröße n wachsen, damit der absolute *speed-up* konstant bleibt?

c) Sie haben für einen parallelen Algorithmus folgende Laufzeiten bei unterschiedlicher Prozessorzahl gemessen. Was können Sie über die Skalierung dieses Algorithmus aussagen?



Aufgabe 3 (*Entwurf+Analyse: CRCW und CREW Modelle*)

Gegeben sei ein Array $a[\cdot]$ im gemeinsamen Speicher, der n Zahlen hält.

a) Beschreiben Sie einen möglichst schnellen parallelen Algorithmus, der überprüft, ob eine der Zahlen durch 7 teilbar ist. Gehen Sie von $p = n$ Prozessoren und dem CRCW *common* Modell aus. Außerdem sei die Teilbarkeit in $O(1)$ prüfbar.

b) Wie ändert sich die Laufzeit, wenn Sie nur noch $p < n$ Prozessoren zur Verfügung haben?

c) Wieviel Zeit würde Ihr Algorithmus im CREW Modell benötigen (wieder $p = n$ Prozessoren)?

d) Geben Sie den absoluten *speed-up* und die *efficiency* für die vorherigen Teilaufgaben an.

e) Im Folgenden soll berechnet werden, wieviele der Zahlen in $a[\cdot]$ durch eine andere Zahl in $a[\cdot]$ (nicht durch sich selbst!) teilbar sind. Sie haben das CRCW Modell und $p = n^2$ Prozessoren zur Verfügung.

Aufgabe 4 (*Entwurf+Analyse:* `findif`*-Anweisung*)

Gegeben sei ein Array $a[\cdot]$ im verteilten Speicher der n Objekte hält. Gesucht ist ein Algorithmus, der eine parallele `findif` Anweisung auf $a[\cdot]$ ausführt. Die Anweisung sortiert die Elemente von $a[\cdot]$ anhand eines Prädikats $pred(\cdot)$, so dass Elemente, die das Prädikat erfüllen, vorne stehen. Die relative Ordnung der Elemente untereinander soll dabei erhalten bleiben.

Bsp.: `findif( {1,4,9,7,3},` *is_bigger_than_*`3 ) = {4,9,7,1,3}`

a) Beschreiben Sie einen Algorithmus, der eine parallele `findif` Anweisung auf $a[\cdot]$ möglichst schnell ausführt. Sie haben $p = n$ Prozessoren zur Verfügung.

b) Untersuchen Sie die Laufzeit der Anweisung für den Fall, dass $p = n$ Prozessoren zur Verfügung stehen und das Prädikat in $T(n) = O(1),\ O(\log n)$ bzw. $O(n)$ ausgewertet werden kann.

c) Wie verhalten sich die Laufzeiten, wenn Sie nur noch $p < n$ Prozessoren zur Verfügung haben?

Aufgabe 5 (*Entwurf+Analyse: Assoziative Operationen*)

Gegeben sei ein Array A im gemeinsamen Speicher bestehend aus n Objekten vom Typ X. Auf den Objekten sei eine Operator $\odot$ definiert. Es sei nach dem Ergebnis von $\odot_{i=1}^{n} a_i$ gesucht.

a) Sei $X = \mathbb{R}^2$ und der Operator definiert als

$$(x_1, x_2) \odot (y_1, y_2) := (x_1 y_1, x_2 + y_2)$$

Zeigen Sie, dass der Operator $\odot$ assoziativ ist.

b) Beschreiben Sie einen schnellen parallelen Algorithmus, der $\odot_{i=1}^{n} a_i$ berechnet und geben Sie dessen Laufzeit $T(n, p)$ an.

c) Nun sei $\odot$ wie folgt definiert: X beschreibe die Menge an möglichen Zeichenketten über dem Alphabet $\{(,)\}$. Die Operation $x \odot y$ verknüpfe beide Zeichenketten und schiebe alle öffnenden Klammern nach links, alle schließenden Klammern nach rechts (Bsp.: $()(() \odot (()) = ((((()))))$. Gehen sie davon aus, dass zu Begin die länge der Zeichenketten konstant ist.

Können Sie den selben Lösungsansatz wie in der vorherigen Teilaufgabe verwenden? Falls nein, geben Sie einen neuen parallelen Algorithmus an. Wie lange dauert die Ausführung?

Aufgabe 6 (*Dijkstras Algorithmus auf dichten Graphen*)

Zeigen sie, dass Dijkstras Algorithmus mit einem binaren Heap eine erwartete Laufzeit in $O(m)$ hat, wenn $m \in \Omega(n \log n \log \log n)$.

Ausgabe: 19.12.2016
Abgabe: keine Abgabe, keine Korrektur