

4. Übungsblatt zu Algorithmen II im WS 2021/2022

http://algo2.iti.kit.edu/AlgorithmenII_WS21.php
 {sanders, hans-peter.lehmann, daniel.seemaier}@kit.edu

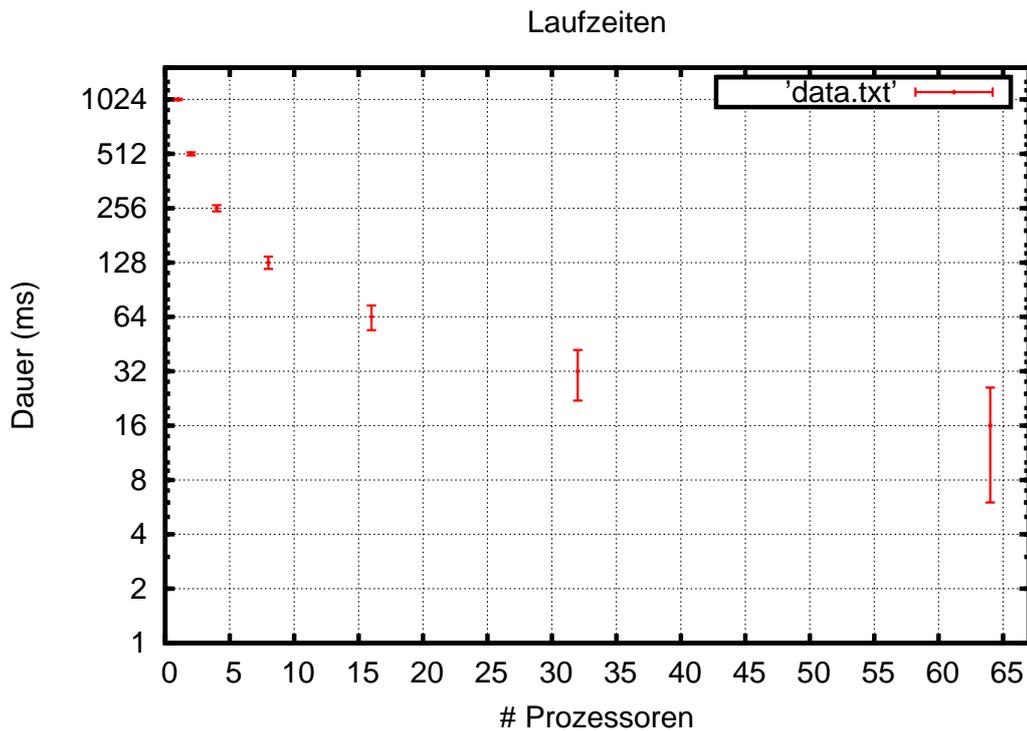
Aufgabe 1 (Kleinaufgaben: Parallele Algorithmen)

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

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

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 2 (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.: $\text{findif}(\{1,4,9,7,3\}, \text{is_bigger_than_3}) = \{4,9,7,1,3\}$

- Beschreiben Sie einen Algorithmus, der eine parallele **findif** Anweisung auf $a[\cdot]$ möglichst schnell ausführt. Sie haben $p = n$ Prozessoren zur Verfügung.
- 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.
- Wie verhalten sich die Laufzeiten, wenn Sie nur noch $p < n$ Prozessoren zur Verfügung haben?

Aufgabe 3 (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.

- 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.

- Beschreiben Sie einen schnellen parallelen Algorithmus, der $\odot_{i=1}^n a_i$ berechnet und geben Sie dessen Laufzeit $T(n, p)$ an.
- Nun sei \odot wie folgt definiert: X beschreibe die Menge an möglichen Zeichenketten über dem Alphabet $\{(\cdot,)\}$. Die Operation $x \odot y$ verknüpfe beide Zeichenketten und schiebe alle öffnenden Klammern nach links, alle schließenden Klammern nach rechts (Bsp.: $()(()) \odot (()) = (((((())))))$). 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 4 (Kleinaufgaben: Eigenschaften von Flüssen)

- Nach Vorlesung ist eine gültige Distanzfunktion $d(\cdot)$ für *Dinitz Algorithmus* gegeben durch:

- $d(t) = 0$
- $d(u) \leq d(v) + 1 \quad \forall (u, v) \in G_f$

Zeigen Sie, falls $d(s) \geq n$, existiert kein *augmentierender Pfad*.

- In der Vorlesung wurde gezeigt, dass die Laufzeit von *Dinitz Algorithmus* für Graphen mit Kantengewichten gleich 1 (*unit edgeweights*) in $O((n+m)\sqrt{m})$ liegt. Vergleichen Sie diese Laufzeit zum *Ford Fulkerson Algorithmus*. Für welche Graphen mit *unit edgeweights* ist welcher der beiden Algorithmen schneller?
- Sei $G = (V, E)$ ein gerichteter Graph, in dem maximale Flüsse berechnet werden sollen. Sei $e = (i, j) \in E$ ebenso wie $e' = (j, i) \in E$, d.h. G besitzt ein Paar entgegengesetzter Kanten. Außerdem sei $c(e) \geq c(e')$. Widerlegen Sie durch ein Gegenbeispiel:
Entfernt man e' aus E und reduziert $c(e) := c(e) - c(e')$, ändert sich der maximale Fluss nicht, d.h. man kann entgegengesetzte Kanten a-priori (für beliebige s und t) gegeneinander aufrechnen.

Aufgabe 5 (Rechnen: Segmentierung mit Flüssen)

Wir betrachten einen einfachen Fall für Bildbearbeitung. Die Vorder-/Hintergrundsegmentierung. Das Ziel dieses Prozesses ist es, ein Bild in Vorder und Hintergrund zu zerlegen. Die Transformation dafür weißt jedem Pixel $p_{i,j}$ des Bildes einen Knoten $v_{i,j}$ im Graphen zu. Für jedes Paar von benachbarten Pixeln $p_{i,j}$ und $p_{k,l}$ ($|i - k| + |j - l| = 1$) fügen wir eine ungerichtete Kante $(v_{i,j}, v_{k,l})$ ein. Zusätzlich fügen wir je einen Knoten s für Vordergrund (Quelle) und einen Knoten t für Hintergrund (Senke) ein. Von Knoten s existiert eine gerichtete Kante zu jedem Knoten $p_{i,j}$ und von jedem Knoten $p_{i,j}$ existiert eine gerichtete Kante zu Knoten t . Wir definieren darüber hinaus folgende Kantengewichte:

$$c(e = (u, v)) = \begin{cases} p_v(v) & u = s \\ p_h(u) & v = t \\ f(u, v) & \text{sonst} \end{cases}$$

Wobei mit $p_v(x)$ die Wahrscheinlichkeit gegeben ist, dass x Vordergrundknoten ist, mit $p_h(x)$ die Wahrscheinlichkeit für einen Hintergrundknoten und mit $f(x, y)$ eine Penaltyfunktion für das Trennen der beiden Knoten x und y . Für ein Graustufenbild B definieren wir

$$p_v(x, y) = B[x, y]^2, \quad p_h(x, y) = (4 - B[x, y])^2 \quad \text{sowie} \quad f((x_1, y_1), (x_2, y_2)) = (4 - |B[x_1, y_1] - B[x_2, y_2]|)^2.$$

Hinweis: Diese Modellierung ist nur ein Beispiel und keine allgemeingültige Modellierung. Sie soll nur verdeutlichen, wie Flow-Algorithmen für andere Probleme eingesetzt werden können.

- Geben Sie den Flussgraphen für das unten angegebene Graustufenbild an.
- Führen Sie einen augmenting Path Algorithmus auf dem entstandenen Graphen aus.
- Wie würde die Segmentierung in Vorder- und Hintergrund im Bild als Ergebnis aussehen?

4	4	1
4	2	0
0	0	0

Aufgabe 6 (Analyse+Entwurf+Rechnen: Grenzüberwachung)

Eine (eindimensionale) Grenzlinie soll durch ein Sensornetz überwacht werden. Zu diesem Zweck wurde eine große Anzahl an Sensorknoten unregelmäßig an der Grenze ausgebracht. Jeder Knoten kann einen Bereich der Grenze für eine gewisse Zeit proportional zu seiner Batteriekapazität überwachen. Die Grenze gilt als vollständig gesichert, wenn jeder Abschnitt der Grenzlinie von mindestens einem Sensorknoten abgedeckt ist. Aufgrund der großen Menge an Knoten sind ihre Überwachungsbereiche stark überlappend. Daher müssen nicht immer alle Knoten aktiv sein, um eine vollständige Sicherung der Grenze zu gewährleisten. So kann Energie gespart werden und die maximale Dauer der Grenzsicherung erhöht werden.

Durch die unregelmäßige Anbringung der Knoten und durch große Fertigungstoleranzen in der Batteriekapazität und dem Überwachungsbereich (*man hat unbedingt beim billigsten Hersteller einkaufen müssen...*) ist zunächst nicht klar, wie lange die Grenze maximal vollständig gesichert werden kann. Glücklicherweise wurden die Positionen der Knoten und ihre jeweiligen Kapazitäten und Detektionsbereiche protokolliert und können verwendet werden, um diese Frage zu beantworten.

- a) In der Vorlesung haben Sie Flussprobleme mit beschränkten Kantenkapazitäten $c(e)$ kennengelernt. Ebenso können Flussprobleme mit beschränkten Knotenkapazitäten $c(v)$ sinnvoll sein. In diesem Fall darf für einen gültigen Fluss die Summe der in den Knoten ankommenden bzw. ausgehenden Flüsse die Kapazität des Knotens nicht überschreiten. Außerdem muss wie bisher für jeden Knoten (außer der Quelle und Senke) die Summe der ankommenden Flüsse gleich der Summe der ausgehenden Flüsse sein.

Erklären Sie, wie maximale Flüsse mit Knotenkapazitäten berechnet werden können. Begründen Sie kurz, warum Ihr Ansatz einen zulässigen und optimalen Fluss berechnet.

- b) Konstruieren Sie ein Flussnetzwerk, das das oben beschriebene Problem der Bestimmung einer maximalen Dauer für die vollständige Grenzüberwachung lösen kann.

Hinweis: Jeder Knoten entspricht einem Sensorknoten. Batteriekapazität kann als äquivalent zur Flussmenge betrachtet werden.

- c) Erstellen Sie ein Flussnetz, das dem folgenden Sensornetz entspricht. Wie lange kann dieses Netz die Grenze im Bereich $[0, 13]$ überwachen? Welche Sensorknoten müssen wann aktiv sein?

Format der Angaben: $x_{nodeID} = \{[begin_range, end_range], capacity\}$

$$\begin{aligned}x_1 &= \{[0, 5], 4\} \\x_2 &= \{[0, 7], 3\} \\x_3 &= \{[4, 9], 2\} \\x_4 &= \{[3, 8], 5\} \\x_5 &= \{[8, 13], 5\} \\x_6 &= \{[7, 11], 3\} \\x_7 &= \{[11, 15], 2\}\end{aligned}$$

Hinweis: Bevor Sie langwierig einen maximalen Fluss berechnen, versuchen Sie ihn durch *scharfes Hinschauen* zu bestimmen.

Ausgabe: 06.12.2021

Abgabe: keine Abgabe, keine Korrektur