Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. Peter Sanders
Moritz Laupichler
Hans-Peter Lehmann

5. Übungsblatt zu Algorithmen II im WS 2022/2023

http://algo2.iti.kit.edu/AlgorithmenII_WS22.php
{sanders, moritz.laupichler, hans-peter.lehmann}@kit.edu

Aufgabe 1 (*Kleinaufgaben: Parallele Algorithmen*)

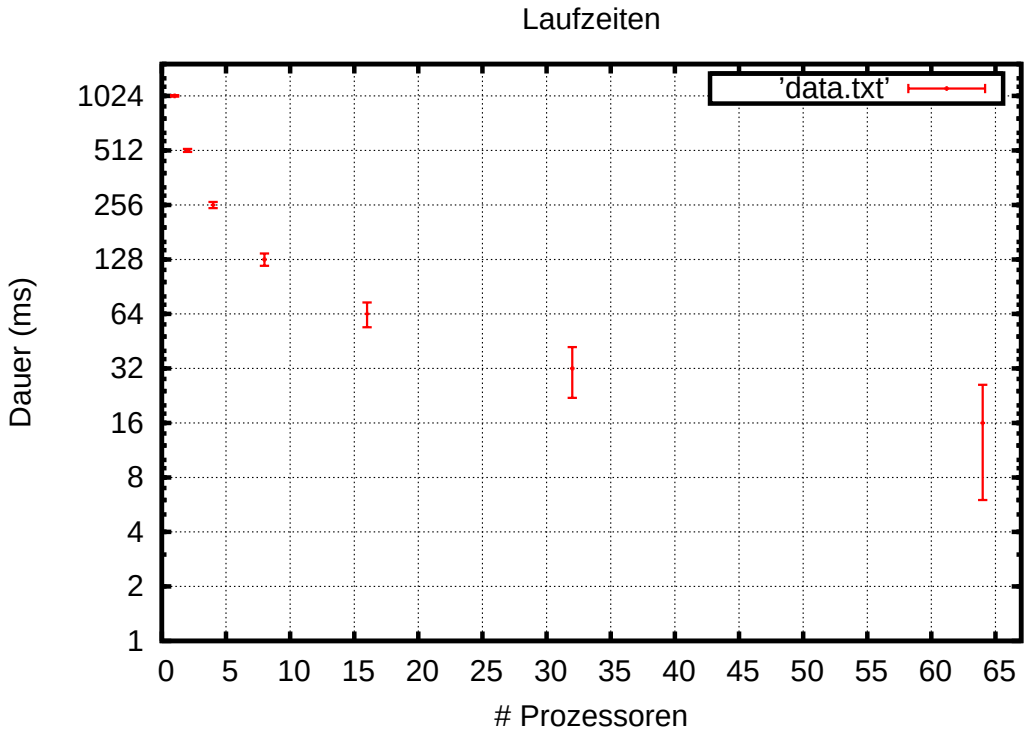
a) Gegeben sei ein paralleler vergleichsbasierter Sortieralgorithmus zum Sortieren von n komplexen Objekten auf p Prozessoren mit einer Laufzeit von

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

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 2 (*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 mindestens 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.

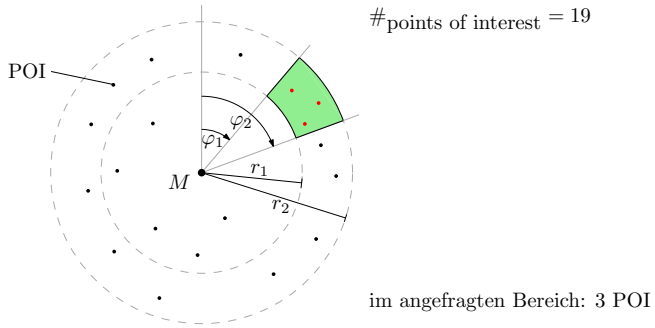
Aufgabe 3 (*Kleinaufgaben: Geometrie-Entwurf*)

Entwerfen Sie einen Algorithmus, der ...

a) in Zeit $O(n \log n)$ ein geschlossenes, kreuzungsfreies Polygon aus n Punkten $P \in \mathbb{R}^2$ konstruiert.

b) in Zeit $O(n)$ für eine Menge von n Filialen einen Standort für ein Zentrallager berechnet, so dass der maximale Abstand (in Luftlinie) zwischen Zentrallager und allen Filialen minimiert wird.

c) (*) in Zeit $O(\log n)$ die Anzahl an *points of interest* (POI) um einen fixen Mittelpunkt M in einem Winkelbereich $[\varphi_1, \varphi_2]$ und einem Entfernungsbereich $[r_1, r_2]$ bestimmt.



Bei n POI ist eine Vorverabeitungszeit von $O(n \log n)$ und ein Platzverbrauch von $O(n)$ erlaubt.

Aufgabe 4 (*Analyse+Entwurf: Überdeckungsproblem*)

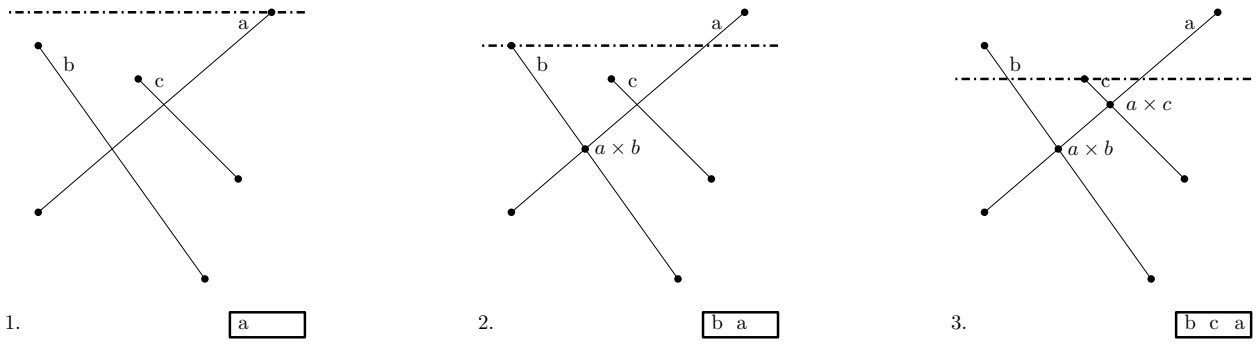
Zur Gebietsüberwachung wurde in einem weitläufigen Gelände ein Sensornetz aus mehreren Millionen Knoten ausgelegt. Die Positionsdaten der Knoten wurden per Funkübertragung an einer zentralen Stelle gesammelt. Jeder Sensorknoten überwacht ein kreisförmiges Gebiet mit Radius r. Durch Fehler in der Ausbringung können dabei starke Überlappungen der von den Knoten überwachten Gebiete entstanden sein.
Um diese Information zu einem späteren Zeitpunkt ggf. nutzen zu können, haben die Betreiber beschlossen, dass alle Knotenpaare bestimmt werden sollen, deren Gebiete sich teilweise überlappen.

a) Zeigen Sie, dass ein *Sweepline*-Algorithmus, der einfach alle (im Rahmen der Algorithmenausführung) aktiven Sensorknoten auf Überlappung prüft, in quadratischer Laufzeit resultieren kann, auch wenn die Ausgabekomplexität (Anzahl überlappender Knotenpaare) linear ist.

b) Überlegen Sie, wie Sie dennoch einen *Sweepline*-Algorithmus verwenden können, um das Problem in Linearzeit zu lösen (exklusive das Sortieren am Anfang), falls die Ausgabekomplexität linear ist.

Aufgabe 5 (*Entwurf: Platzeffizienter Linienschnitt*)

In der Vorlesung wurde ein *Sweepline*-Algorithmus zur Bestimmung der Schnitte zwischen n Liniensegmenten behandelt. Der Algorithmus arbeitet eine Liste an Ereignispunkten (Schnittpunkte, Linienanfänge und Linienenden), in Form einer nach der y-Position sortierten *Queue*, ab. Gleichzeitig führt er eine Liste aktiver Kanten in sortierter Reihenfolge.
Das betrachtete Problem lässt sich in seiner Laufzeitkomplexität nie besser lösen als durch die Anzahl Schnittpunkte vorgegeben. Liegen z.B. $O(n^2)$ Schnittpunkte vor, so kann bestenfalls eine quadratische Laufzeitkomplexität erreicht werden.
Für den Algorithmus aus der Vorlesung gilt die gleiche Einschränkung auch für den Platzbedarf. Die *Queue* muss bis zu $O(n + k)$ Ereignispunkte gleichzeitig halten, bei insgesamt k Linienschnitten. Dies liegt unter anderem daran, dass einmal erkannte Schnittpunkte auch zwischen Liniensegmenten existieren können, die in der sortierten Liste nicht (mehr) benachbart sind. Dies sei durch folgendes Beispiel illustriert:



Im Beispiel wird zuerst der Schnittpunkt $a \times b$ zwischen den Linien a und b bestimmt. Nach der Aktivierung von Linie c ist der Schnittpunkt weiterhin in der *Queue*, die Linien a und b sind allerdings nicht mehr benachbart.

Modifizieren Sie den Algorithmus so, dass er nur noch $O(n)$ Platz für die Ereignispunkte benötigt.

Aufgabe 6 (*Analyse+Entwurf:* Graham's Scan)

Betrachten Sie den *Graham's Scan* Algorithmus zur Bestimmung der konvexen Hülle einer Punktemenge $P \in \mathbb{R}^2$ mit $|P| = n$. In der Vorlesung wurde eine lexikographische Sortierung der Punkte vorgeschlagen, mit der Folge dass die obere und untere Hülle getrennt berechnet werden mussten.

a) Geben Sie eine geeignetere Sortierung der Punkte an, so dass die gesamte konvexe Hülle in einem Durchlauf berechnet werden kann.

b) Zeigen Sie, dass der *Graham's Scan* Algorithmus nicht für jede beliebige Sortierung der Punkte eine korrekte konvexe Hülle liefert (Randfälle ausgenommen).

c) Zeigen Sie anhand eines Beispiels, dass es möglich ist, dass der *Graham's Scan* Algorithmus p schon zur Hülle hinzugefügte Punkte hintereinander verwirft für beliebig großes p.

d) Eine Schneidemaschine bringt Stoffe anhand eines Schnittmusters in die gewünschte Form. Ein Schnittmuster ist dabei durch ein Polygon aus n Ecken definiert, das selbst nicht notwendigerweise konvex ist. Die Schneidemaschine kann beliebige konvexe Stoffe bearbeiten.

   Entwerfen Sie einen Algorithmus, der den minimal möglichen Verschnitt für ein gegebenes Stoffmuster in Linearzeit berechnet. Die Ausgabe des Algorithmus soll also die Gesamtfläche des Verschnitts sein, welcher anfällt, wenn ein optimales konvexes Stoffstück zum gegebenen Schnittmuster verarbeitet wird. Sie können davon ausgehen, dass die Ecken des Polygons als geschlossener Kantenzug vorliegen.

Ausgabe: 17.01.2023
Abgabe: keine Abgabe, keine Korrektur