Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. Peter Sanders
Moritz Laupichler
Nikolai Maas

6. Übungsblatt zu Algorithmen II im WS 2023/2024

https://algo2.iti.kit.edu/AlgorithmenII_WS23.php
{sanders, moritz.laupichler, nikolai.maas}@kit.edu

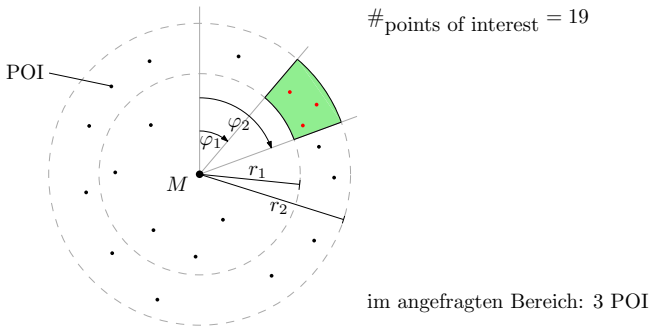
Aufgabe 1 (*Kleinaufgaben: Geometrie-Entwurf*)

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 2 (*Analyse+Entwurf: Überdeckungsproblem*)

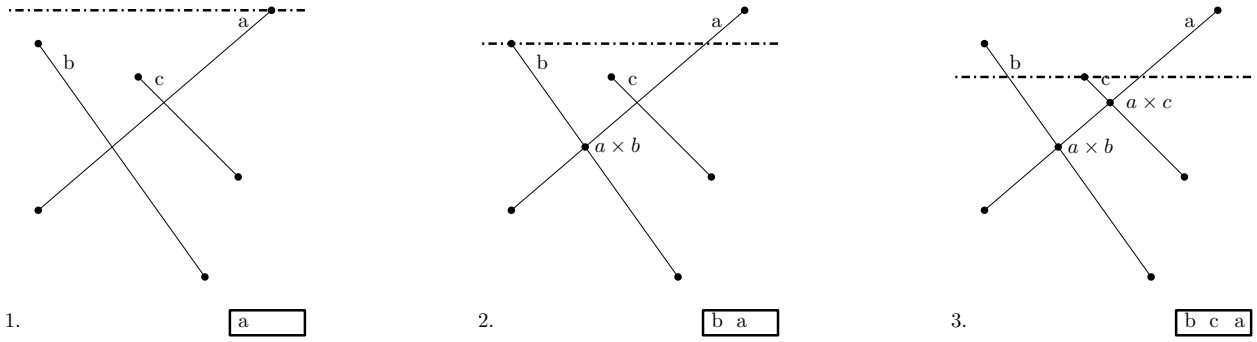
Zur Gebietsüberwachung wurde in einem weitläufigen Gelände ein Sensornetz aus mehreren Millionen Knoten ausgelegt. Die Positionsdaten der Knoten wurden per Funkübertragung an einer zentralen Stelle gesammelt. Jeder Sensorknoten überwacht ein kreisförmiges Gebiet mit Radius r. Durch Fehler in der Ausbringung 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 und die Ausdehnung des betrachteten Gebiets für n Knoten durch $\mathcal{O}(rn)$ beschränkt ist. Dabei darf zusätzlich angenommen werden, dass in jedem Kreis mit Radius r nur eine konstante Anzahl an Sensorknoten liegt.

Aufgabe 3 (*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 4 (*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.

Aufgabe 5 (*Analyse: ADAC Mitgliedschaft*)

Der "**A**utomobil **D**urch **A**lgorithmiker **C**lub" (ADAC) leistet auf Autobahnen Pannenhilfe. Ein Autofahrer hat in seiner Zeit als Verkehrsteilnehmer n Pannen, $n \in \mathbb{N}_{\geq 0}$, für die er die Hilfe des ADAC in Anspruch nehmen muss. Für jede geleistete Pannenhilfe verlangt der Club eine Aufwandsentschädigung abhängig von der Schwere der Panne. Mitglieder beim ADAC müssen lediglich ein Viertel dieser Kosten bezahlen. Eine lebenslange Mitgliedschaft kann man sich durch eine Einmalzahlung in Höhe von 1000 DM (**D**ijkstra **M**ark) sichern.
Da nicht schon mit Erwerb des Führerscheins klar ist, wie viele Pannen man in seinem Leben haben wird und wie schwerwiegend diese sein werden, stellt sich die Frage, ab wann es sich lohnt eine Mitgliedschaft beim ADAC zu erwerben.

a) Geben Sie eine Strategie an, die einen kompetitiven Faktor (competitive ratio) von ∞ erreicht. Begründen Sie kurz.

b) Wie gut ist die Strategie, sich nie eine Mitgliedschaft beim ADAC zu sichern? Begründen Sie.

c) Zeigen Sie, dass folgende Strategie einen kompetitiven Faktor $c = 3$ hat. Die Strategie ist, sich beim Pannenhelfer eine Mitgliedschaft zu kaufen, wenn die momentan von ihm bearbeitete Panne die Gesamtausgaben für Pannen (ohne Mitgliedschaft) auf über 500 DM anheben würde.

Hinweis: Verwenden Sie die summierten Gesamtkosten K über alle Pannen (ohne Mitgliederrabatt).

Aufgabe 6 (*Analyse: Online-gaming-Algorithmen*)

Ein Angestellter in einem Rechenzentrum hat einen recht eintönigen Job: Er ist für das Scheduling einer Maschine im Rechenzentrum zuständig. Dazu muss der Mitarbeiter die laufenden Jobs auf der Maschine überwachen und einen neuen Job starten, falls ein Job fertig geworden ist. Da sich der Angestellte damit oft langweilt, spielt er viel lieber sein neues Computerspiel, das er zu Weihnachten geschenkt bekommen hat.
Eine neue Dienstanweisung besagt jedoch, dass die teuren Großrechner zu mindestens 50% ausgelastet sein müssen. So muss der Mitarbeiter häufiger die Maschine überprüfen und wird ständig beim Spielen gestört!!! Helfen Sie also dem Mitarbeiter, zu minimieren, wie oft er die Maschine überprüfen muss.
Es gibt genau eine Maschine, welche immer genau einen Job gleichzeitig ausführen kann. Dabei können Jobs nicht unterbrochen werden. Die Laufzeit jedes Jobs ist unbekannt, jedoch braucht jeder Job mindestens 5 Minuten. Die Zeit zum Starten eines Jobs kann für die Bestimmung der Auslastung vernachlässigt werden. Es gibt keine automatischen Benachrichtigungen über das Ende eines Jobs, sondern der Mitarbeiter muss selbst periodisch überprüfen, ob ein Job fertig geworden ist. Solange bis der letzte Job abgeschlossen ist, stehe nach Beenden eines Jobs immer ein nächster Job bereit.

a) Entwerfen Sie einen *online scheduling* Algorithmus, der unter Einhaltung der Nebenbedingungen minimiert, wie oft der Mitarbeiter überprüfen muss, ob ein Job fertig geworden ist.

b) Zeigen Sie, dass Ihr Algorithmus ein optimaler Online-Algorithmus bzgl. der Anzahl Überprüfungen ist. (Das heißt in diesem Kontext, dass der Algorithmus unter allen Algorithmen, die die Einhaltung der Nebenbedingungen garantieren, eine minimale Anzahl von Überprüfungen erreicht.)

   Hinweis: Zeigen Sie, dass für jeden Online-Algorithmus, der eine geringere Anzahl an Überprüfungen ermöglicht, eine Folge von Jobs existiert, sodass die Nebenbedingungen nicht eingehalten werden.

Ausgabe: 30.01.2024
Besprechung: 13.02.2024