Algorithmen und Datenstrukturen

Sommersemester 2022

Prof. Dr. Philipp Jenke

HAW HAMBURG | Department Informatik

Aufgabe 1

Dieses Aufgabenblatt baut auf den Vorlesungen auf:

- Abstrakte Datentypen,
- Komplexität.

Deliverables:

- PDF mit Lösungen zur Theorie-Aufgaben + Mess- und Analyseergebnisse zu Ansteckungen
- Code für Praxisaufgabe (Code-Konventionen!)

1 Theorie

1.1 Komplexität

Beweisen Sie:

a) $23 \cdot x + 2 \cdot \sqrt{x}$ ist $\mathcal{O}(x)$

b) $\sqrt{n} \notin \mathcal{O}(\ln n)$

c) $n \cdot \ln n = \mathcal{O}(n^2)$

1.2 Abstrakter Datentyp

Spezifizieren Sie den Datentyp `Schiff` für eine Handelssimulation. Ein Schiff hat eine Beschreibung (Text) und eine Menge von durchnummerierten Stellplätzen für Kisten.

Ein Schiff hat die folgenden Operationen:

- Beladen: Übergabe einer Liste von Kisten, belegen von freien Stellplätzen
- Entladen: Rückgabe der Kiste an einer bestimmten Stelle

2 Praxis

Im Vorgabeprojekt finden Sie Code für die Simulation zur Ausbreitung eines Virus. Dazu betrachten wir Personen (Klasse `Person`) in einem geschlossenen rechteckigen Raum. Jede Person hat eine Position im 2D (Klasse `Vector2i`) und einen Gesundheitszustand: gesund, krank, immun. Wir simulieren die Ausbreitung des Virus bei den Personen über die Zeit: $t_0, t_1, t_2, \ldots$. Alle gesunden Personen stecken sich bei kranken Personen mit dem Virus an, die weniger als ϵ entfernt sind. Erkrankt eine Person, dann ist sie für t_{krank} Zeitschritte krank. Im Anschluss ist sie immun und kann nicht mehr erkranken. In jedem Zeitschritt bewegt sich jede Person um ein zufälliges kleines 2D-Offset. Die Simulation wird mit einer einfachen JavaFX-GUI dargestellt. Dort sind außerdem die aktuellen Fallzahlen der Gesundheitszustände visualisiert. Alle Parameter der Simulation werden in Form von Konstanten in der Klasse `Constants` gesetzt. Machen Sie sich zunächst mit der Vorgabe vertraut.

2.1 Bewegung

Wir betrachten zunächst die Bewegung der Personen. Die Person soll dabei nicht in jedem Schritt in eine beliebige Richtung laufen. Vielmehr soll jede Person eine aktuelle Ausrichtung haben, in die sie sich bewegt. Die Schrittweite ist dabei vorgegeben (`MOVE_DISTANCE`). Dazu hat jede Person immer eine aktuelle Ausrichtung, repräsentiert durch einen Winkel α. Den Winkel können Sie immer in einen normierten 2D Richtungsvektur $d = (\cos(\alpha), \sin(\alpha))$ umrechnen. In jedem Schritt wird der Winkel dann ein wenig zufällig verändert.

Dabei können die Personen den Raum aber nicht verlassen. Erreicht eine Person den Rand des Simulationsgebiets, so dreht sie sich um 180 Grad. Das Feld `simulationArea` der Klasse `Simulation` gibt die Breite und Höhe des Feldes an.

Überlegen Sie sich einen Algorithmus für die Bewegung, formulieren Sie ihn in Pseudocode und implementieren sie ihn.

2.2 Ansteckung

Erweitern Sie nun die Simulation, sodass kranke Personen gesunde anstecken. Finden Sie dazu für alle Personen jeweils alle anderen Personen, die weniger als `INFECTION_RADIUS` entfernt sind. Ist die Person krank, dann werden automatisch alle anderen gesunden Personen in diesem Umkreis ebenfalls krank.

Hinweis: Man könnte die Nachbarn auch nur für kranke Personen suchen; diese Optimierung soll hier aber nicht durchgeführt werden.

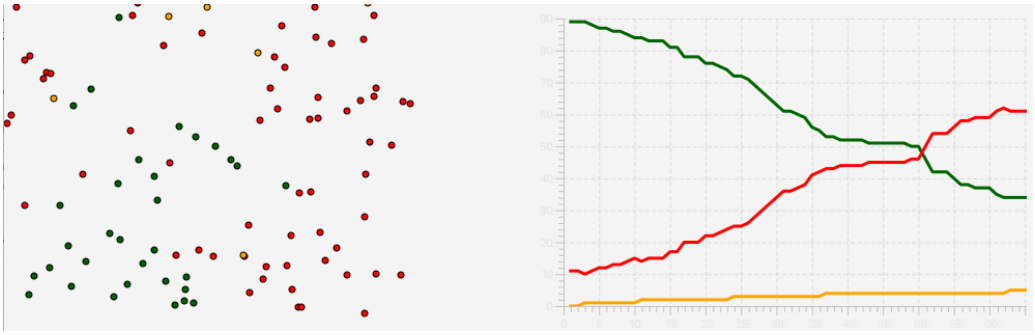

Abbildung 1: Screenshot aus der laufenden Anwendung mit Punkt-Darstellung der Personen (links) und Verteilung der Gesundheitszustände (rechts).

Messen Sie den Aufwand zum Finden aller ϵ-Nachbarn aller `Individuen` mit unterschiedlichen Populationsgrößen (z.B. 1000, 2000, 3000, 4000, ...) - entweder durch eine Zeitmessung oder durch Zählen der zentralen Operationen. Achten Sie darauf, dass Sie nur das Finden der Nachbarn messen, nicht die Verarbeitung des Ergebnisses. Stellen Sie die Ergebnisse in einem Graphen (lineare Achsen-Skalierung) dar und interpretieren Sie das Ergebnis. Damit die Ergebnisse verwendbar sind, müssen Sie mehrere Messungen mit der gleichen Populationsgröße durchführen und das Ergebnis mitteln.