ÜBUNGSBLATT 5

Algorithmen mit Matrizen

AUFGABE 5.1: MATRIZENRECHNER (3.5 PUNKTE)

1. Implementieren Sie eine Funktion, die eine Matriz der Dimension (M, N) mit Zufallszahlen initialisiert; **(0.5 Punkte)**
2. Implementieren Sie folgende Matrizenoperationen als Funktion (siehe https://de.wikipedia.org/wiki/Matrix_(Mathematik)):
   a. Skalarmultiplikation: Multiplikation einer Matrix mit einer skalaren Zahl; **(0.5 Punkte)**
   b. Matrizenaddition: Addition von zwei Matrizen; **(0.5 Punkte)**
   c. Matrizenmultiplikation: Multiplikation von zwei Maztrizen; **(0.5 Punkte)**
   d. Matrixtransposition: Transposition einer Matrix. **(0.5 Punkte)**
3. Verwenden Sie diese Funktionen um einen Matrizenrechner zu implementieren. **(0.5 Punkte)**
4. Bestimmen Sie die Laufzeit- und Speicherkomplexität aller Funktionen. **(0.5 Punkte)**

AUFGABE 5.2: GRAPH ALS ADJAZENZMATRIX (3.5 PUNKTE)

Eine Adjazenzmatrix *A* (auch Nachbarschaftsmatrix genannt) eines kantengewichteten gerichteten Graphen ist eine Matrix, die speichert, welche Knoten des Graphen durch eine Kante verbunden sind und die Länge dieser Kanten. Sie besitzt für jeden Knoten eine Zeile und eine Spalte, woraus sich für *N* Knoten eine $N \times N$ Matrix ergibt. Der Eintrag A_{ij} der Adjazenzmatrix repräsentiert die Kante zwischen Knoten *i* und Knoten *j*. Ein Wert größer 0 gibt an, dass die Kante existiert, und ist gleichzeitig die Distanz der Kante.

Ein Graph mit *N* Knoten und *E* Kanten nennt man informell dünnbesetzt wenn *E* viel kleiner als N^2 ist $(E \ll N^2)$.

Die Anzahl der Knoten *N* und Kanten *E* in einem gerichteten Graph werden eingelesen.

1. Implementieren sie eine Funktion, die einen gerichteten Graph mit *N* Knoten und *E* Kanten als Adjazenzmatrix mit Zufallskanten und Zufallsdistanzen initialisiert; **(1 Punkt)**
2. Implementieren Sie eine Funktion, die überprüft, ob ein gerichteter Graph dünnbesetzt ist, wobei die Operation „viel kleiner als ≪“ als Funktion freiwillig zu definieren ist; **(0.5 Punkte)**
3. Implementieren Sie eine zweite Funktion die überprüft, ob zwei eingegebene Knoten in einem gerichteten Graph direkt mit einer Kante miteinander verbunden sind. Die Funktion soll **„`wahr`“** zurückgeben falls die Knoten verbunden sind und ansonsten **„`falsch`“**; **(0.5 Punkte)**
4. Implementieren Sie eine dritte Funktion die überprüft, ob zwei eingegebene Knoten in einem gerichteten Graph indirekt mit einem Weg durch mehrere Kanten miteinander verbunden sind. Die Funktion soll **„`wahr`“** zurückgeben falls die Knoten verbunden sind und ansonsten **„`falsch`“**; **(1 Punkt)**
5. Bestimmen Sie die Laufzeit- und Speicherkomplexität der drei Funktionen. **(0.5 Punkte)**