Programmieren 2 (INF)
Übungsblatt 3

Ziel dieses Übungsblattes ist es, Ihre Kenntnisse der Sprache Java weiter aufzufrischen. Am Beispiel der Traveling Salesman Problems (TSP) sollen Entwurf von Datenstrukturen und Algorithmen sowie deren Implementierung geübt werden. Welche Sprachmittel Sie dazu einsetzen ist bei dieser Übung Ihnen überlassen.

Aufgabe 1

Machen Sie sich anhand der Vorlesungsfolien sowie ggf. weiterer Internetquellen vertraut mit dem TSP. Wir konzentrieren uns hier auf Probleme, bei denen die Position der Orte auf einer Ebene über Koordinaten spezifiziert wird und die Reisekosten über die Euklidische Metrik ermittelt werden.

Aufgabe 2

Entwerfen Sie geeignete Klassen (nicht notwendigerweise 3!)

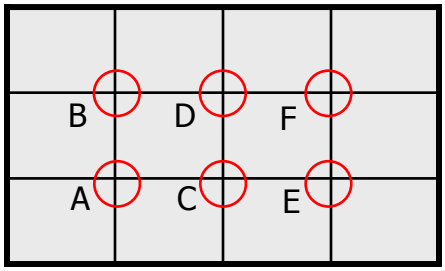
- zur Repräsentation eines Ortes (charakterisiert durch einen Namen und 2 Koordinaten),
- zur Repräsentation einer Rundreise und
- zur Repräsentation eines TSPs.

Überlegen Sie, welche Operationen auf Objekten dieser Klassen durchgeführt werden müssen, um ein TSP zu definieren und dann zu lösen. Legen Sie entsprechende Dummy-Methoden an.

Aufgabe 3

Ihr Programm soll dem Benutzer später die Möglichkeit bieten, zwischen drei TSPs auszuwählen. Sie sind wie folgt definiert:

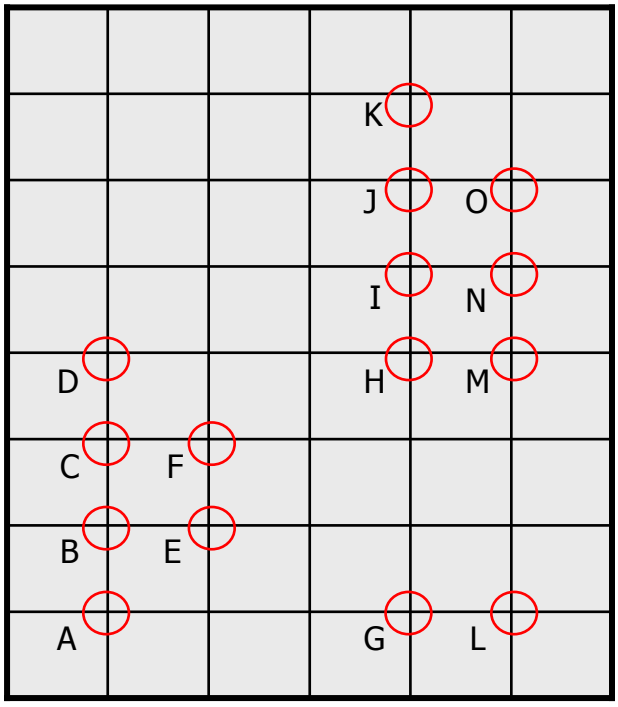
Leiterplattenbohrproblem 1:



Das Leiterplattenbohrproblem 2 enthält die Zielpunkte A-K,
das Leiterplattenbohrproblem 3 enthält zusätzlich die Zielpunkte L-O.

Bemerkung: Die Berechnung einer exakten Lösung für Problem 3 ist schwer!

Leiterplattenbohrproblem 2/3:



Schreiben Sie 3 Methoden, die Ihnen die in Aufgabe 2 implementierten Datenstrukturen mit den Daten der Beispiel-TSPs füllen. Schreiben Sie auch eine `toString()`-Methode, die Ihnen beim Debuggen der TSPs hilft.

Aufgabe 4

Implementieren Sie eine Methode zur Orts-Abstandsberechnung (Kostenfunktion). Verwenden Sie dabei die Euklidische Metrik. Sie benötigen dazu die Wurzelfunktion (`Math.sqrt(…)`).

Aufgabe 5

Nun kommen wir zum schwierigsten Teil. Wählen Sie ein Lösungsverfahren aus (z.B. den Greedy-Algorithmus aus der Vorlesung) und implementieren Sie es. Mindestanforderung ist, dass Ihr Algorithmus in der Lage ist, irgendeine Route für die Probleme 1 und 2 zu berechnen. Natürlich wäre es schön, wenn die Touren möglichst kurz ausfallen würden…

Aufgabe 6

Schreiben Sie eine Main-Methode mit einer kleinen Benutzerführung (ggf. in einer neuen Klasse), die es dem Benutzer erlaubt, eines der drei TSPs auszuwählen und den Lösungsalgorithmus zu starten. Als Ergebnis sollte die berechnete Rundreise (in irgendeiner Form) ausgegeben werden, sowie die Länge der Reise.

Aufgabe 7 (optional)

Es bietet sich beim TSP an, verschiedene Lösungsverfahren/Heuristiken auszuprobieren. Wenn Sie Zeit haben, nutzen Sie die Gelegenheit für ein paar Experimente. Neben dem Vergleich der Qualität der berechneten Rundreisen (deren Länge) ist insbesondere ein Vergleich der Rechenzeiten interessant.

Tipps

Es gibt einige Stellen in der TSP-Software bei der sich Exception-Handling sinnvoll einsetzten lässt. Es bietet sich also an, auch diese sprachlichen Möglichkeiten von Java zu nutzen und somit den Umgang damit zu üben. Beispielsweise wäre das (versehentliche) Entfernen eines Ortes aus einer leeren Liste von Orten eine Situation, die abgefangen werden könnte und sollte.

In den Algorithmen wird gelegentlich eine Kopie einer Liste von Orten benötigt. Dies könnte durch Überladen der `Object`-Methode `clone()` erfolgen. Dabei sind verschiedene Dinge zu beachten (tiefe Kopie, `Cloneable`-Interface implementieren, `CloneNotSupportedException` abfangen etc.), siehe z.B. unter

`http://docs.oracle.com/javase/8/docs/api/java/lang/Object.html#clone`

Viel Spaß beim Experimentieren!