Fachbereich Informatik
Wesley Terpstra
Dr.-Ing. Ilia Petrov
Prof. Alejandro Buchmann, Ph. D.
Sommersemester 2009

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Grundlagen der Informatik II

1. Praktikumsaufgabe

Um eine Mission vorzubereiten muss die Dienststelle die Abdeckung des Zielgebiets durch Spionagesatelliten optimieren. Die Verwendung eines Satelliten ist kostspielig, und daher sollten so wenig Satelliten wie möglich eingesetzt werden. Außerdem besitzt jeder Satellit einen eigenen Orbit. Manche operieren in geringer Höhe und bewegen sich sehr schnell; sie sind also nur für kurze Zeit über dem Zielgebiet. Andere bewegen sich langsamer in großer Höhe und sind folglich länger aber dafür weniger oft über dem Zielgebiet. Also sind Satelliten zu unterschiedlichen Zeitpunkten für unterschiedliche Zeitlängen nutzbar. Viele Satelliten überfliegen das Zielgebiet sogar mehrfach.

Um den Missionserfolg zu sichern, müssen jederzeit während der Mission mindestens drei Satelliten über dem Zielgebiet zur Verfügung stehen. Dies können wechselnde Satelliten sein; ein späterer Satellit kann einen früheren im Missionsverlauf ersetzen. Durch die begrenzte Anzahl von Satelliten im Orbit sind nicht jederzeit drei Satelliten über dem Missionsgebiet. Ihre erste Aufgabe ist es, die längste mögliche Missionszeit zu bestimmen, in der sich jederzeit mindestens drei Satelliten über dem Gebiet befinden. Unsere Strategen werden dann eine Missionszeit bestimmen, und Ihre Aufgabe ist es dann, die Anzahl der tatsächlich benutzten Satelliten zu minimieren und dennoch die notwendige Abdeckung zu erreichen. Dies erfordert mindestens drei Satelliten, aber da einzelne Satelliten das Zielgebiet vor Missionsabschluss verlassen könnten, kann ihre Ersetzung notwendig sein, was zusätzliche Satelliten erfordert.

Glücklicherweise verfügen wir über andere Techniker, die in den Details von Satellitenflugbahnen geschult sind. Diese Techniker haben alle Satelliten identifiziert, welche das Zielgebiet überfliegen. Für jeden dieser Satelliten haben Sie eine Liste der Zeitintervalle erstellt, in denen sich der Satellit über dem Zielgebiet befinden wird. Um die tatsächlichen Missionszeiten geheim zu halten, werden sie nur Zeiten in Integer-Format statt tatsächliche Zeiten erhalten. Wenn z.B. der Satellit „RadioHunter“ das Zielgebiet überfliegt, erhalten Sie eine Liste { { 5, 9 }, { 12, 15 } } um anzuzeigen, dass er zu den Zeiten 5-9 und 12-15 nutzbar ist.

Sie sollten wie immer Ihre Undercoverkontakte in der GDI2-Vorlesung nutzen, um die Aufgabe zu diskutieren. Aus Sicherheitsgründen müssen Sie darauf achten, dass Sie die Lösung vollständig verstanden und selbst implementiert haben.

Zu implementieren: Surveillance.java für die Satellitenauswahl!

Die Lösung ist bis zum **10. Mai 2009 23:59** einzureichen!

Achtung: Diese Anweisungen sind mit besonderer Sorgfalt zu behandeln und werden sich automatisch selbst zerstören!

Schritt 0: Laden Sie die Sourcecode-Vorgaben von `https://www.dvs.tu-darmstadt.de/teaching/inf2/2009-de/Problem1.zip` herunter.

Schritt 1a: Implementieren Sie Surveillance.Event.compareTo.

Schritt 1b: Konvertieren Sie die Satellitenabdeckungsintervalle in Enter- und Leave-Ereignisse.

Hinweis: Dokumentation zum Collections-Framework finden Sie unter `http://java.sun.com/javase/6/docs/technotes/guides/collections/reference.html`.

Hinweis: Benutzen Sie Collections.sort, um die Ausgabeliste zu sortieren.

Schritt 2a: Bestimmen Sie alle Intervalle mit ausreichender Satellitenabdeckung (satCoverage).

Hinweis: Gehen Sie die zeitsortierte Eventliste durch und zählen Sie die jeweils sichtbaren Satelliten.

Schritt 2b: Bestimmen Sie den längsten möglichen Missionszeitraum (longestSatCoverage).

Schritt 3: Bestimmen Sie bei gegebener Missionszeit die minimale Anzahl benötigter Satelliten (fewestSatellites).

Hinweis: Benutzen Sie einen klassischen Backtracking Algorithmus.

Hinweis: Fangen Sie mit allen Satelliten an, und entfernen Sie diese dann recursiv.

Hinweis: Benutzen Sie intervalCovered, um zu überprüfen, ob eine Satellitenabdeckung für die Mission noch gegeben ist.

Hinweis: Sie können einen Satelliten wieder in die Menge aufnehmen (*.put()*), nachdem er aus die Menge entfernet (*.remove()*) wurde.

Denkaufgabe: Der hier vorgeschlagene Backtracking-Algorithmus ist sehr langsam. Wie könnte seine Performance verbessert werden? Wenn das Problem so geändert würde, dass jeder Satellit nur für genau einen Intervall zur Verfügung stünde, wäre die Lösung sehr viel einfacher. Mit einer Vorgehensweise (namens Greed), die wir später kennen lernen werden, wäre eine Lösung in $O(n)$ Schritten möglich (plus $O(n \log n)$ Sortierungskosten). Was an den multiplen Intervallen macht das Problem so viel aufwendiger zu lösen?

Zusätzliche Informationen für diese Aufgabe finden Sie unter:

a) `http://en.wikipedia.org/wiki/Back_tracking?oldid=281593259`

b) `http://java.sun.com/docs/books/tutorial/java/generics/index.html`