8. Programmieraufgabe

Programmierparadigmen

LVA-Nr. 194.023
2024/2025 W
TU Wien

Themen:

Funktionale Programmierung

Ausgabe:

09. 12. 2024

Abgabe (Deadline):

13. 01. 2025, 14:00 Uhr

Abgabeverzeichnis:

`Aufgabe8`

Programmaufruf:

`java Test`

Grundlage:

Skriptum, Schwerpunkt auf Abschnitt 5.1 und 5.2

Kontext

Wir wollen ein Optimierungsproblem lösen, bei dem es darum geht, eine gute Struktur für ein Gebäude oder einen Gebäudekomplex zu finden. Dabei werden oberirdische Gebäudeteile symbolisch durch Würfel (alle gleich groß) dargestellt, die in einem würfelförmigen Raster (Gitterabstand entspricht der Würfelgröße, Ausrichtung nach Himmelsrichtungen) angeordnet sind. Konkret sollen n Würfel so angeordnet werden, dass sie eine zulässige Struktur bilden, die durch eine vorgegebene Kennzahlberechnung möglichst gut bewertet wird. Bei größerem n ist es nicht möglich, innerhalb vernünftiger Zeit alle möglichen Anordnungen auszuprobieren. Stattdessen wird rekursiv eine nicht notwendigerweise optimale Lösung berechnet: Sind n Würfel anzuordnen, wird zunächst eine Anordnung für $n - 1$ Würfel gesucht, danach werden alle möglichen Varianten zum Hinzufügen des letzten Würfels bewertet und verglichen.

Zu Beginn ($n = 1$) liegt ein Würfel am Grund des Rasters. Schrittweise werden weitere Würfel angefügt. Dabei gelten diese Bedingungen:

- Jeder Würfel wird vollflächig auf die freie Oberseite eines vorhandenen Würfels oder einen freien Platz am Grund des Rasters gesetzt. Kein Würfel kann „in der Luft hängen“.
- Bei $n > 1$ muss mindestens eine Fläche jeden Würfels vollflächig an eine Fläche eines anderen Würfels anschließen. Da Würfel einzeln hinzugefügt werden, ergibt sich eine zusammenhängende Struktur.
- An mindestens eine seitliche Fläche jedes Würfels darf kein anderer Würfel anschließen. An der freien Seite können Fenster liegen.
- Es dürfen nicht mehr als eine vorgegebene maximale Zahl m Würfel übereinander gestapelt sein.

Jedes so konstruierte Gebilde wird durch eine Kennzahl (größer ist besser) bewertet, die der Summe der Bewertungen aller Würfel entspricht. Die Bewertung eines Würfels ist die Summe folgender Zahlen:

- Die *thermische Oberflächenqualität* ist die Summe der sechs pro Fläche des betrachteten Würfels folgendermaßen ermittelten Zahlen:
  - 1 wenn die Fläche direkt an einen anderen Würfel grenzt;
  - sonst 0,2 wenn es sich um eine besonnte Ostseite handelt, wobei eine Ostseite als besonnt gilt, wenn im gesamten Bereich östlich bis südlich der betrachteten Würfelfläche[1] kein anderer Würfel auf gleicher Höhe liegt;
  - sonst 0,1 wenn es sich um eine besonnte Westseite handelt, wobei eine Westseite als besonnt gilt, wenn im gesamten Bereich westlich bis südlich der betrachteten Würfelfläche kein anderer Würfel auf gleicher Höhe liegt;

[1] Das heißt, südlich des betrachteten Würfels dürfen schon andere Würfel liegen, aber nicht südöstlich davon, da nur die östliche Fläche des Würfels betrachtet wird.

  - sonst 0,5 wenn es sich um eine besonnte Südseite handelt, wobei eine Südseite als besonnt gilt, wenn sich südlich der betrachteten Würfelfläche (von West bis Ost) kein Würfel um z Einheiten (Würfelkantenlängen) höher als (bzw. mit $z = 0$ auf gleicher Höhe wie) der betrachtete Würfel in kürzerem Nord-Süd-Abstand als $5 \cdot (z + 1)$ befindet[2];
  - sonst 0 (obere, untere, nördliche oder beschattete Seite).

- Die *Qualität der Aussicht auf die Umgebung* ist die Summe der vier pro seitlicher Fläche ermittelten Zahlen, die sich jeweils aus dem Produkt der auf folgende Weise ermittelten Zahlen ergeben:

  - 0 wenn die Fläche an einen anderen Würfel grenzt, sonst 1;
  - $a/25$ wenn sich mit Abstand $a < 25$ Einheiten vor der Fläche ein anderer Würfel befindet, sonst 1;
  - für die Kanten unten, links und rechts der betrachteten Fläche: 1/2 wenn an einer dieser Kanten das Sichtfeld eingeschränkt ist, das heißt, ein anderer Würfel vorsteht oder diese Kante am Grund des Rasters liegt (unterster Würfel), 1/4 wenn an zwei Kanten das Sichtfeld eingeschränkt ist, 1/8 wenn an allen drei Kanten das Sichtfeld eingeschränkt ist, sonst 1.

Mit folgendem Algorithmus wird nach einer gut geeigneten Anordnung der Würfel gesucht, wobei „Lösung“ für eine zulässige Anordnung der Würfel steht, die alle oben genannte Bedingungen einhält:

Als Parameter gegeben: n (Anzahl der Würfel),
m (maximale Höhe),
k (Anzahl der zurückzugebenden Lösungen).
Wenn $n = 1$, dann
    erstelle eine Lösung mit nur einem Würfel und gib sie zurück;
sonst ($n > 1$)
    führe den Algorithmus rekursiv mit $n - 1$, m und k aus;
    zurück kommen bis zu k Lösungen für $n - 1$;
    erstelle aus allen (bis zu k) bekannten Lösungen für $n - 1$
        die Menge aller möglichen Lösungen für n durch
        Hinzufügen eines Würfels an je einer Stelle, wo das zulässig ist;
    bewerte jede dieser Lösungen;
    entferne Lösungen außer einer bestbewerteten, bis max. k übrig;
    gib die verbleibenden Lösungen zurück.

Der Algorithmus gibt also in den meisten Fällen k Lösungen zurück, von denen nur eine die beste Bewertung im jeweiligen Durchführungsschritt haben muss. Die anderen Lösungen werden zufällig gewählt. Dabei sollen möglichst unterschiedliche Lösungen zum Zug kommen. Lösungen können so gewählt werden, dass besser bewertete Lösungen mit höherer Wahrscheinlichkeit zurückgegeben werden als schlecht bewertete. Zurückgegebenen Lösungen sollen unterschiedliche Bewertungen haben, weil andernfalls die Gefahr besteht, dass die Lösungen fast gleich sind.

[2] Weiter entfernte Würfel blockieren Strahlen der hoch stehenden Sonne nicht.

Welche Aufgabe zu lösen ist

Java-8-Streams und Funktionen höherer Ordnung

Algorithmus und Bewertung als Funktionen

3 Optimierungsaufgaben

Verwenden Sie zur Implementierung der unter „Kontext“ beschriebenen Aufgabe in Java einen funktionalen Ansatz, der sowohl Java-8-Streams, als auch selbst geschriebene funktionale Formen verwendet. Wesentliche Programmteile (Funktionen) sind so auszulegen, dass Argumente (z. B. für n, m und k) frei wählbar sind. Auch die für die Bewertung verwendete Funktion soll eigenständig und austauschbar gestaltet sein. Beim Testen sind konkrete Werte einzusetzen. Wählen Sie Werte so, dass sich ein Programmlauf in etwa 20 bis 60 Sekunden ausgeht. In dieser Zeit sind drei Optimierungsaufgaben zur Anordnung von Würfeln mit unterschiedlichen Parametern auszuführen. Zwei Optimierungen sollen die unter „Kontext“ beschriebene Vorgehensweise zur Bewertung von Anordnungen und unterschiedliche Werte für n, m und k verwenden (mit möglichst großen Werten für n und k), eine dritte Optimierung soll eine andere, frei von Ihnen festlegbare Funktion zur Bewertung verwenden.

Wie üblich soll das Programm durch einen Aufruf von `java Test` vom Abgabeverzeichnis `Aufgab8` aus ausführbar sein. Für jeden Optimierungslauf sind die gewählten Parameterwerte und die vom Algorithmus zurückgegebenen Würfelanordnungen mit der besten und schlechtesten Bewertung (zusammen mit den Bewertungen) auszugeben. Die Ausgabe einer Würfelanordnung auf dem Bildschirm soll ähnlich einer aus Ziffern und Buchstaben zusammengesetzten Landkarte gestaltet sein, wobei jedes Zeichen für eine bestimmte Anzahl von übereinander liegenden Würfeln steht (Leerzeichen für 0, `1` für 1, `2` für 2, ..., `A` für 10, `B` für 11, ...). Das kann beispielsweise so aussehen:

```
35741
 252
```

Dabei liegen die meisten Würfel im Norden, die drei aufeinanderliegenden Würfel im Westen und so weiter.

Aufgabenaufteilung beschreiben

Wie üblich soll die Datei `Test.java` als Kommentar eine kurze, aber verständliche Beschreibung der Aufteilung der Arbeiten auf die einzelnen Gruppenmitglieder enthalten – wer hat was gemacht.

Wie die Aufgabe zu lösen ist

funktionale Denkweise soll klar erkennbar sein

Es wird vorausgesetzt, dass eine ganz eindeutig funktionale Denkweise zum Einsatz kommt, irgendwo Java-8-Streams vorkommen, aber irgendwo auch selbst geschriebene funktionale Formen (also Funktionen, die Lambdas oder andere Objekte, die vorwiegend als Funktionen verwendet werden, als Parameter haben) vorkommen. Ob Sie vorwiegend mit Java-8-Streams oder eigenen funktionalen Formen arbeiten möchten, ist Ihnen überlassen, solange beides in einem sinnvollen Zusammenhang vorkommt. Dabei wird etwas als sinnvoll betrachtet, wenn es keine einfachere Möglichkeit gibt, das Gleiche auf simple herkömmliche Weise darzustellen. Beispielsweise sollten Sie nicht einfach nur gewöhnliche Methoden durch Lambdas ersetzen, die Sie an Variablen zuweisen und über die Variablen wie normale Methoden verwenden. Wichtig ist, dass durch die Lösung ein Verständnis der Ziele hinter der funktionalen Programmierung zum

Ausdruck kommt. Es kommt sowohl auf Programmiereffizienz als auch eine effiziente Programmausführung an. Idealerweise werden vorgefertigte Programmteile aus Standardbibliotheken eingesetzt.

Sie sollen Seiteneffekte so gut es geht vermeiden. Ganz vermeidbar sind Seiteneffekte in Java nicht, etwa bei der Ausgabe. Allerdings wird dabei die Ebene der funktionalen Programmierung verlassen und eine imperative (vermutlich prozedulare) Ebene betreten. Die Schnittstelle zwischen diesen Ebenen muss so klar wie möglich gestaltet sein, etwa durch eine deutliche Abgrenzung (z. B. Auslagerung in eine andere Klasse) und deutliche Hinweise in Form von Kommentaren. Bedenken Sie auch, welche Strukturierung einer Zusammenarbeit zwischen prozeduralen und funktionalen Teilen in der Regel unproblematisch ist und versuchen Sie, eine solche Strukturierung herzustellen.

klare, sichere Schnittstelle zwischen funktionalen und nichtfunktionalen Teilen

Kommentare in der funktionalen Programmierung unterscheiden sich deutlich von solchen in der objektorientierten Programmierung. Darin spiegelt sich eine andere Form der Abstraktion wider. Versuchen Sie bitte bewusst, Kommentare so einzusetzen, wie das in der funktionalen Programmierung üblich ist. Konzepte wie Design-by-Contract sind in diesem Zusammenhang nicht von Bedeutung.

Kommentare entsprechend eines funktionalen Stils

Die Aufgabe gibt nicht vor, wie Datenstrukturen zu gestalten sind. Sie müssen selbst einen Weg finden, Würfelanordnungen auf sinnvolle Weise im Speicher abzubilden, wobei vor allem die in den Bewertungen benötigten Daten rasch auffindbar sein müssen. Sorgen Sie bitte dafür, dass vor allem bei Bewertungen nicht immer wieder die gleichen Berechnungen auf den gleichen Datenmengen durchgeführt werden. Das vorgegebene Raster ist zwar für die verständliche Beschreibung der Aufgabe wichtig, daraus folgt aber nicht unbedingt, dass das Raster auch in den Daten abgebildet sein muss; vor allem Raster mit begrenzter Größe könnten problematisch werden (weil vermutlich zu groß oder klein und daher ineffizient gewählt).

Was im Hinblick auf die Beurteilung wichtig ist

Die insgesamt 100 für diese Aufgabe erreichbaren Punkte sind folgendermaßen auf die zu erreichenden Ziele aufgeteilt:

Schwerpunkte berücksichtigen

- funktionale Prinzipien eingehalten — 40 Punkte
- Algorithmus und Bewertungen richtig und effizient — 40 Punkte
- für funktionale Programmierung typische Kommentare — 10 Punkte
- Lösung vollständig (entsprechend Aufgabenstellung) — 10 Punkte

Der erste Punkt bezieht sich darauf, ob eine funktionale Denkweise erkennbar ist und sowohl Java-8-Streams als auch selbst geschriebene funktionale Formen in deutlich erkennbarem Ausmaß und korrekt zum Einsatz kommen. Dieser Punkt bezieht sich aber auch darauf, ob bei gleichzeitigem Einsatz eines anderen Paradigmas (in vergleichsweise kleinem Ausmaß) die Schnittstellen zwischen den Paradigmen auf eine sichere Weise strukturiert und deutlich beschrieben sind.

Anders als in früheren Aufgaben spielt die korrekte und effiziente Umsetzung des vorgegebenen Algorithmus' eine große Rolle. Eine fehlerhafte

und ineffiziente Umsetzung kann (auch bei vergleichsweise unbedeutenden Ursachen) zu bedeutendem Punkteverlust führen.

Kommentare sind extra angeführt, um nochmals darauf hinzuweisen, dass auch hierbei ein Umdenken nötig ist. Es gibt Punkteabzüge, wenn Kommentare fehlen oder einem objektorientierten Stil folgen.

Warum die Aufgabe diese Form hat

Der Schwerpunkt liegt darauf, Techniken der funktionalen Programmierung anzuwenden. Nachdem in den letzten Aufgaben vorwiegend Techniken der objektorientierten Programmierung im Mittelpunkt standen, soll durch diese Aufgabe intuitiv klar werden, wie stark sich die funktionale Denkweise und Zielsetzung von der objektorientierten unterscheidet. Vorgegebene algorithmische Beschreibungen müssen verstanden und möglichst effizient (sowohl im Sinne der Programmiereffizienz als auch der Laufzeiteffizienz) umgesetzt werden, wobei aber die langfristige Wartbarkeit des Programms vernachlässigt werden kann. Ein Teil der Aufgabe verlangt, kreativ eine eigene Bewertungsfunktion zu schreiben. Einerseits gibt das die nötige Freiheit, um eigene Ideen in der Aufgabe zu verwirklichen, andererseits sollen Sie dabei erkennen, dass der Austausch eines Programmteils durch einen anderen in der funktionalen Programmierung anders funktioniert als in der objektorientierten.