SES 2021-'22 - taak Backtracking - domino

Opgave

Domino is een spel waarbij je een stapel stenen krijgt met aan elke kanten van de steen een aantal ogen. De bedoeling is een kant met een bepaald aantal ogen tegen een kant van een reeds gelegde steen met hetzelfde aantal ogen te leggen. Je mag zowel links als rechts aansluiten. Wanneer je het spel alleen speelt, wil je zoveel mogelijk stenen leggen; wanneer je het spel met verschillende spelers speelt, eindigt het spel wanneer iemand alle stenen gelegd heeft of geen enkele speler nog stenen kan laten aansluiten. Daarna worden het aantal ogen van de resterende stenen per speler geteld. De speler met het minste aantal ogen wint.

Om deze opgave wat interessanter/moeilijker te maken, hebben we in deze variant elke steen een kleur gegeven. Je mag geen stenen met dezelfde kleur tegen elkaar leggen.

Doel van de basisopgave is om een volgorde voor de stenen te bepalen zodat je de volledige verzameling stenen in een cirkel kan leggen. Wanneer dit niet kan, geef je een Optional.empty() terug.
Met deze basisopgave kan je maximaal 14/20 halen.
Je kan op twee manieren drie extra punten verdienen zodat je uiteindelijk aan 20/20 kan geraken.

- Door minder stenen te gebruiken wanneer je met de volledige verzameling stenen niet aan een oplossing geraakt.
- Door àlle oplossingen terug te geven. Dit stuk is niet verteld in de les, maar het komt er op neer dat volgende aanpassingen moet doen:
    - Houd een lijst bij met daarin alle oplossingen.
    - Wanneer je een oplossing gevonden hebt, return niet onmiddellijk, maar maak een (diepe) kopie van die oplossing, voeg die toe aan de lijst van de vorige stap en doe daarna de backtracking stap.
      Het is nodig om een diepe kopie te maken omdat je anders ook de objecten in je originele lijst verandert wanneer je verder backtrackt.
      Een diepe kopie maken van *origineel* naar *kopie* doe je met volgende code:

```
ArrayList<Iets> kopie = new ArrayList<>();
for (Iets o: origineel) {
    kopie.add(o.clone());
}
```

    - Wanneer je normaal zou returnen met Optional.empty() geef dan de lijst met alle oplossingen terug.

Ontwerp en implementeer in Java een backtracking-algoritme voor deze opgave.

Concreet heb je volgende input:

- Een lijst met objecten van de klasse Steen.
- De klasse Steen krijg je in de startcode van het project en bevat:
    - Het aantal ogen op de ene kant van de steen (type **int**).
    - Het aantal ogen op de andere kant van de steen (type **int**).
    - De kleur van de steen (type **java.awt.Color**)
    - Of de steen 'geflipped' is, of m.a.w. of de kanten gewisseld zijn.

Als return-waarde van de functie geef je Optional element terug van een lijst van stenen in correcte volgorde.

Startproject

Je mag vertrekken van een startproject met daarin een aantal klassen. Enkel de klasse Algoritme moet je aanpassen. Dit project bestaat in een Maven-versie voor NetBeans en een Gradle-versie voor IntelliJ.

Je kan het project gewoon uitvoeren van de IDE, maar ook van commandline als je een jar-gegenereerd hebt. In NetBeans doe je dit met Build, in IntelliJ zit dit al in de configuratie inbegrepen.

Op de plaats waar de jar gemaakt wordt, hebben we een aantal invoerbestanden gezet.

- Geven een oplossing: stenen2.txt, stenen3.txt, stenen11.txt, stenen12.txt
- Géén oplossing: stenen4.txt, stenen8.txt, stenen9.txt

Hiervoor moet je wel de stack op (minstens 512MB) zetten.
Om dit uit te voeren, moet je eerst naar de map gaan met de jar:

- in NetBeans is dit de map **target**
- in IntelliJ is dit de map **build/libs**

Voer dan volgende opdrachtregel uit op de command prompt:

- in NetBeans:
  java -Xss512m -cp taakBacktrackingDomino-1.0-SNAPSHOT.jar be.domino.Main "stenen8.txt"
- in IntelliJ:
  java -Xss512m -jar taakBacktrackingGradle-1.0-SNAPSHOT.jar "stenen12.txt"

Hierbij hebben we de stack-grootte op 512MB gezet met de vlag -Xss512m.

Unit testing

Een belangrijk deel van de evaluatie staat op de unit testing. Splits hiervoor het probleem op in kleine(re) deelfuncties en schrijf hiervoor unit tests uit. Vooral de controles op een veilige plaatsing van een ploeg kan je via unit testing ontwikkelen.

Praktisch

- Dit is een **individuele** taak.
- Voor de upload moet je het Java-project inpakken (in zip, rar, 7z of iets anders) en uploaden op Toledo. Je mag ook een link van een Git-repository uploaden als je met Git werkt.
- Zorg er voor dat de naam van de projectmap begint met je achternaam gevolgd door je voornaam, bijvoorbeeld ObamaBarack
  Dus in géén geval dingen zoals taak-backtracking of zo, want dan weet ik totaal niet van wie de oplossing is.
- Zet bovenaan de klasse Main.java in commentaar het aantal uren dat je +- aan deze taak gewerkt hebt, en zet ook je naam bij @author
- De deadline is geprikt op **1 mei 2022.**

Kris Aerts
24 maart 2022