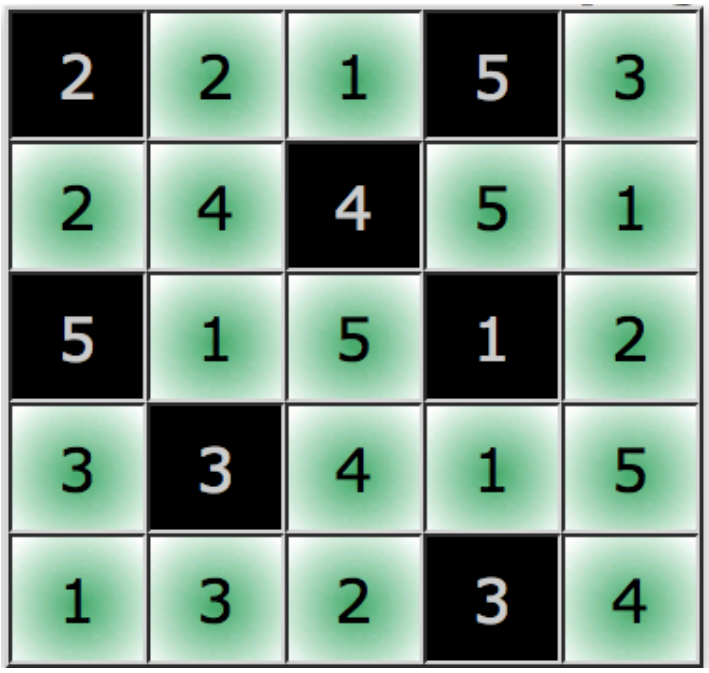
1 Toegepaste informatica
Februari 2016
Opdracht TW2: Hitori
Lector: G. Jongen, J. Van Hee

HITORI



Bron figuur: http://hitoriconquest.com/?puzzleSize=5

1 Instructies

1.1 De opdracht

Deze tekst bevat de opgaven van de opdracht 'Hitori'. Je werkt aan deze opdracht in een groepje van twee studenten, tenzij anders afgesproken met de docent. Deze opdracht vertegenwoordigt 30% van de quotering voor het opleidingsonderdeel 'Toegepaste wiskunde 2'. Bij een tweede examenkans (augustus) blijft het cijfer voor deze opdracht behouden. Het examen in augustus gaat dan ook op 14 punten.

Je lector kent de punten *individueel* toe. Het kan dus gebeuren dat jij een andere quotering behaalt dan je medestudent als daar een goede reden voor is.

Het opleidingsonderdeel 'Toegepaste wiskunde 2' heeft 3 studiepunten. Een studiepunt komt ongeveer overeen met 25 uur werk (lessen volgen, studeren, opdrachten maken, examens afleggen, ...). Omgerekend betekent dit dat we verwachten dat je ongeveer $0,30 \times 75 = 22,5$ uur werkt aan deze opdracht (wat dus neerkomt op 45 uur voor een groepje van twee).

Bij deze opdracht programmeer je functies die een willekeurige puzzel *zo ver mogelijk* oplossen en een functie die van een gegeven oplossing kan nakijken of ze voldoet aan alle voorwaarden.

We verwachten dat beide groepsleden een evenwaardige bijdrage leveren aan *beide* componenten, dus niet een werkverdeling waarbij iemand al het algoritmisch werk en de andere partner testresultaten bij mekaar zoekt.

Voor het *praktisch* gedeelte wordt de software 'Scilab' gebruikt. Van de functies verwachten we dat ze

- correct werken;
- voldoende gedocumenteerd zijn (documentatie binnen de functie-code);
- efficiënt zijn;
- de gebruiker feedback geven indien er iets fout loopt (foutboodschap met de `error`-functie);
- uitvoerig getest zijn, ook met niet-triviale voorbeelden.

Een functie schrijven die alle puzzels (ook de allermoeilijkste) kan oplossen is niet zo eenvoudig. Dat mag je beschouwen als een stevige uitdaging. Begin echter bij de eenvoudige stukken van het oplossingsproces en probeer zover mogelijk te

geraken. Bij de evaluatie wordt er rekening gehouden met hoe ver dat je geraakt in je oplossingen en met de moeilijkheidsgraad van de afzonderlijke strategieën.

Het resultaat van de opdracht is **werkende** programmacode:

- een programma dat controleert of een puzzel al dan niet correct opgelost is
- een programma dat een niet-opgeloste puzzel (volledig) oplost
- een bestand met testresultaten

1.2 Afgeven

1.2.1 Deadline

Week van 16 mei 2016 (dat is de laatste week van het semester) bij het begin van de les. De exacte datum vind je op Toledo bij 'Opdracht'.

1.2.2 Wat afgeven?

We verwachten concreet dat elk groepje twee dingen afgeeft:

1. Alle bestanden bundel je samen in een archief (zip) en geef je af als "Toledo opdracht".
2. Een txt-bestand dat alle programmacode bevat wordt als 'Turnitin Assignment' afgegeven via Toledo. Turnitin vergelijkt versies onderling met elkaar en signaleert plagiaat (overnemen van teksten of code van elkaar of zonder toestemming van andere auteurs). Je krijgt hierover in de les nog uitleg.

2 Hitori logische puzzel

De figuur op de voorpagina van deze tekst is een hitori. Een hitori is een Japanse puzzel. Je vindt verschillende on-line bronnen, bijvoorbeeld `http://hitoriconquest.com` of `http://www.brainbashers.com/hitori.asp`. Deze laatste site geeft je ook een aantal tips om de puzzel op te lossen (`http://www.brainbashers.com/hitorihelp.asp`). Zoek zelf nog bijkomende bronnen om meer te weten te komen over de verschillende oplossingsstrategieën. Vermeld de interessantste bovenaan als commentaar in je code.

De moeilijkheidsgraad van een hitori wordt bepaald door zijn omvang, maar ook door het aantal strategieën die nodig zijn om de puzzel op te lossen. Bij de moeilijke puzzels moet je soms enkele stappen vooruit denken om de impact van een kleur van een vakje in te schatten. Dit zorgt ervoor dat puzzels met kleine omvang soms toch moeilijk op te lossen zijn.

3 Hoe begin ik aan deze opdracht?

Het doel van deze opdracht bestaat erin dat je in Scilab functies schrijft die een logische puzzel (zo ver mogelijk) oplossen en kunnen nakijken of een gegeven oplossing juist is. *Dit is geen gemakkelijke opdracht.* Als het nodig is, kan je lector je op weg helpen.

Om inzicht te krijgen in de puzzel, besteed je best eerst wat tijd aan het zelf oplossen ervan. Op het internet vind je on-line puzzels. Er bestaan ook (gratis en betalende) apps voor tablets en smartphones, zowel voor iOS als voor android. Je begint hier best zo snel mogelijk in dit semester mee. Elke avond een uurtje voor het slapengaan een hitori oplossen ...

Je zal merken dat je verschillende oplossingsstrategieën gebruikt als je zelf een puzzel te lijf gaat. Kan je je strategie in woorden uitdrukken en opdelen in een aantal kleine logische stappen?

In het OPO Algo 1 wordt de nadruk gelegd op *herbruikbare code*. Schrijf deelfuncties die je nadien kan hergebruiken. Als je een functie schrijft die een getal vindt in een rij, kan je mits enkele kleine aanpassingen die functie ook gebruiken om een getal te vinden in een kolom of een deelmatrix. Volg ook de richtlijn die in Java opgelegd wordt: een functie bevat nooit meer dan 10 regels code. Omdat Scilab een minder compacte programmeertaal is dan Java, mag je gaan tot 20 regels.

Sciab heeft een groot aantal voorgedefinieerde functies voor matrixmanipulatie: deelmatrices maken, som berekenen van elementen van een matrix, getal vinden in vector ... Maak er gebruik van.

4 En wat als ik vast zit?

Dan moet je wroeten tot je terug los geraakt! Je staat er niet alleen voor. We zullen een aantal lessen vrijmaken zodat je vragen kan stellen en kan laten zien waar je mee bezig bent. Via Toledo laten we weten in welke lessen er aan de opdracht

gewerkt kan worden. Kom in elk geval voorbereid naar deze lessen, zodat je gericht vragen kan stellen. Aangezien je altijd je laptop bij hebt, kunnen we flexibel zijn en bvb. ook een stukje van een les aan de opdracht werken als er op het einde van een les nog wat tijd over blijft. Het grootste deel van deze opdracht moeten jullie thuis uitvoeren.

5 Wat wordt er van je verwacht?

5.1 Lijst met hitori's

Maak een lijst met al dan niet opgeloste hitori's van verschillende moeilijkheidsgraad. Je kan een puzzel niet zelf maken, je moet voorbeelden zoeken op het internet, in de krant ... De puzzel oplossen is wel aan jou. Werk eventueel samen met klasgenoten. Als 5 groepjes elk twee puzzels (opgave en oplossing voor de controlefunctie) intikken en uitwisselen, beschik je al over 10 puzzels om te testen.

5.2 Hitori modelleren in Scilab

Werk met twee matrices: een matrix N (*numbers*) met de gegeven cijfers en de matrix C (*colors*) met de kleurcode voor elk vakje.

Afspraak: In de matrix met kleurcodes stellen we een leeg vakje voor met '-1', een zwart vakje met '0' en een wit vakje met '1'. Voor de leesbaarheid van de code is het aan te raden om deze waarden niet hard te coderen, maar drie variabelen `leeg=-1`, `zwart=0` en `wit=1` te definiëren.

5.3 Controlefunctie

Schrijf een functie `check(N,C)`. Input zijn twee matrices: de opgave met getallen N en een oplossing met kleuren C. Uitvoer is de boolean `true` als de matrix C een correct opgeloste hitori voor opgave N is en `false` anders.

5.4 Solver

Schrijf een functie `y=losOp(N)`. Input is een niet-opgeloste hitori (matrix met getallen). Uitvoer is een matrix die de oplossing van de puzzel voorstelt (matrix met kleurcodes).

- Schrijf leesbare code. De functie `losOp` roept dus verschillende (kleinere) methodes op die elk een aparte oplossingsstrategie hanteren. Geen spaghetti-code!
- Schrijf herbruikbare code.
- Documenteer je code. Schrijf bij elke (deel)functie wat die functie doet.
- Als `losOp` een puzzel niet volledig opgelost krijgt, geeft de functie een half opgeloste puzzel terug. **Omwille van onze testfuncties moet de uitvoer steeds een matrix zijn!**

5.5 Testen

Voorzie een bestand met testresultaten. Voeg in commentaar toe of je de puzzel al dan niet hebt kunnen oplossen.

5.6 Minimale vereisten

Na de paasvakantie zullen we je een indicatie geven welke de moeilijkheidsgraad is die je code moet kunnen oplossen om de helft van de punten te behalen. Je code moet in ieder geval aan volgende vereisten voldoen:

- werkende code
  - functies kunnen uitgevoerd worden
  - functies geven gewenste uitvoer (dus geen foutboodschap zoals 'Sorry, maar deze puzzel kan ik nog niet oplossen')
- efficiënte code
- functie `check(N,C)` die controleert of een puzzel al dan niet opgelost is (uitvoer: boolean `%T` of `%F`)
- functie `losOp(N)` die de opgeloste puzzel teruggeeft.

5.7 Dit doe je niet

Je kan hitori's ook oplossen met functies die *backtracking* of *brute force* gebruiken. Dit levert je voor deze opdracht geen extra punten op en is zeker niet voldoende als 'minimale vereiste'.