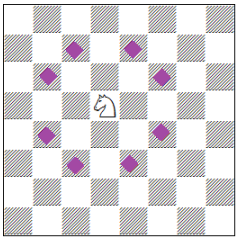
Practicum OO – lente 2016 **Opgave 2**

Leerdoelen

Na het maken van deze opgave kun je

- multidimensionale rijen in Java maken en gebruiken;
- klassen en rijen combineren tot nieuwe datatypen.

De rondgang van een paard op een schaakbord



Het paard is een van de zes stukken uit het schaakspel. Een paard beweegt 2 velden horizontaal met 1 veld verticaal of 2 velden verticaal met 1 veld horizontaal.

Het vinden van een reeks zetten die ieder veld van het schaakbord precies één keer bezoekt is een meer dan 1000 jaar oude puzzel. Zo'n pad heet een *rondgang van het paard* of een *Knight's tour*. Als het paard van het laatste veld van die rondgang met een geldige zet weer met naar het begin kan springen is die rondgang *gesloten*. In plaats van een 8×8 bord bekijkt men vaak een $m \times n$ bord, met $m \leq n$. Allen Swenk bewees dat er altijd zo'n gesloten rondgang bestaat tenzij m en n allebei oneven zijn, of $m \in \{1, 2, 4\}$, of $m = 3$ én $n \in \{4, 6, 8\}$. Het kleinste vierkante bord met een oplossing is dus 6×6.

Een van de $26,534,728,821,064 \approx 3 \times 10^{13}$ mogelijke oplossingen voor een gesloten pad op een 8×8 bord staat hiernaast.

Voor een 8×8 bord zijn er 4×10^{51} mogelijke paden van 64 paardensprongen over het bord. Het is dus, zelfs voor een snelle computer, erg veel werk om die allemaal te bekijken. In deze opdracht maak je een programma dat één oplossing probeert te vinden door een deel van een pad systematisch te verlengen.

7	24	21	18	35	32	27	16	37
6	19	34	23	26	17	36	31	28
5	22	25	20	33	46	29	38	15
4	63	50	45	56	61	40	47	30
3	44	55	62	51	48	57	14	39
2	3	64	49	60	41	6	11	8
1	54	43	2	5	52	9	58	13
0	1	4	53	42	59	12	7	10
	0	1	2	3	4	5	6	7

Datastructuren

Voor het vinden van een pad representeren we het bord als een tweedimensionale matrix van getallen. Het pad geven we aan als een serie van opeenvolgende getallen. In bovenstaand voorbeeld zijn we linksonder met 1 begonnen.

Een positie op het bord is een paar van twee indexen. We beginnen hier bij 0 te tellen. De mogelijke sprongen van het paard zijn ook paren van getallen: $(1, 2), (1, -2), (2, 1), (2, -1), \ldots$ Door zo'n sprong bij de huidige positie op te tellen krijgen we de nieuwe positie.

In de cursus programmeren zouden we direct een twee-dimensionale rij voor het huidige bord en een rij van sprongen gebruiken. In deze cursus maken we natuurlijk klassen voor de positie en het bord. Omdat het zoeken lang kan duren is het handig om te zorgen dat je algoritme met borden van verschillende afmetingen overweg kan. Je mag je beperken tot vierkante borden. Deze klassen hebben methoden voor het bepalen van de positie na een sprong, het bezetten van een veld op de gegeven positie, etc.

Algoritme

Er is geen algoritme bekend dat voor alle borden direct een geschikt pad construeert. Ons algoritme zal dus stapsgewijs met *trail-and-error* zo'n pad moeten construeren. Het beste kunnen we dit doen door systematisch een deel van het pad uit te breiden tot een geheel pad. Op het bord markeren we de velden die gebruikt zijn. Vanaf de huidige positie proberen we daartoe systematisch alle mogelijk sprongen. Als zo'n sprong naar een onbezet veld op het bord leidt zoeken we van daaruit recursief verder totdat alle velden bezet zijn.

Heuristieken

Voor een 6×6 bord hoort het beschreven algoritme goed te werken. Bij een 8×8 bord zal gemiddeld ongeveer 1 van de 10^{20} mogelijke paden van 64 opeenvolgende paardensprongen een oplossing zijn. Het zoeken kan dus lang duren. Door de volgorde van de zetten te ordenen volgens een heuristiek kan het zoeken vaak aanzienlijk versneld worden. Er zijn enkele bekende heuristieken die voor veel borden werken, maar niet altijd.

- In 1823 stelde Warnsdorff voor om altijd te kiezen voor het veld met de minste opvolgers, dat is het veld van waaruit de minste sprongen mogelijk zijn. We kijken hierbij naar de huidige bezetting van het het bord en niet naar het aantal mogelijke sprongen op een leeg bord.

  Als er meerdere velden met het zelfde aantal mogelijkheden zijn koos Warnsdorff er willekeurig een. Ira Pohl stelde in 1967 voor om dan te kijken naar het aantal mogelijkheden een stap verder.

- Arne Roth stelde voor om te kiezen voor het veld dat het verst van het centrum van het bord ligt. Dat wil zeggen liever bij de rand dan midden op het bord en liever in een hoek dan elders aan de rand.

Beide heuristiek geven de voorkeur aan velden met weinig vervolgstappen. De uitwerking die wij gemaakt hebben sorteert de velden waar het paard heen kan, naar de som van de afstanden tot de dichtstbijzijnde randen van het bord. Het algoritme doorloopt deze velden van klein naar groot. Omdat er maximaal 8 sprongen mogelijk zijn, voldoet hiervoor het meest eenvoudige sorteeralgoritme.

Deze heuristieken suggereren ook dat het verstandig is om bij het zoeken van een rondgaand pad in een hoek te beginnen.

Opdracht

Maak een Java programma dat voor een bord van gegeven afmetingen een rondgang van het paard zoekt en afdrukt. Via een boolean is te kiezen of dit pad al of niet gesloten moet zijn. Implementeer een heuristiek zodat ook voor 8×8 borden in redelijke tijd een oplossing gevonden wordt. Geef in JavaDoc een kort toelichting op de heuristiek die jullie gebruikt hebben.

Optioneel: Uitbreidingen

Er zijn verschillende mogelijkheden om dit algoritme uit te bereiden:

- In een *Magic Knight's Tour* is de som van alle rijen en kolomen (en eventueel de diagonalen) gelijk.
- Voor grote borden werkt het om het bord op te splitsen in kleinere delen (bijvoorbeeld 6×6). Het oplossen van zo'n kleiner deel gaat efficiënt met het boven ontwikkelde algoritme.

Inleveren

Lever **zondag 14 februari, vóór 23:59 uur**, via Blackboard alle `.java` files uit de package in.