ALGORITHMS & DATA STRUCTURES

LES 3 : RECURSIE EN SORTEREN

OPDRACHTEN

THEORIE

Actie	**Hoofdstuk**	**2010**	**2012**
Bestudeer	Recursion	7.1, 7.3	7.1, 7.3
Bestudeer	Sorting algorithms	8.1 t/m 8.4	8.1 t/m 8.4

WEB

Animatie van verschillende sorteeralgoritmes: www.sorting-algorithms.com

THEORIE OPGAVEN

OPGAVE 1 (8.3ABC)

Sort the sequence 6, 0, 9, 3, 0, 5, 7, 3, 1, 5 by using

a) Insertion sort
b) Shellsort for the increments {5, 3, 1}
c) Mergesort

OPGAVE 2

Gegeven de volgende lijst: 6, 2, 5, 3, 4, 1, 0, 7. Sorteer de lijst met behulp van:

a) Insertion sort
b) ShellSort (increments 5, 3, 1)
c) MergeSort

OPGAVE 3 (8.7)

When the input has been sorted in reverse order, what is the running time of

a) Mergesort
b) Insertion sort

OPGAVE 4 (8.14)

When all keys are equal, what is the running time of

a) Mergesort
b) Insertion sort

OPGAVE 5 (8.15)

When the input has been sorted, what is the running time of

a) Mergesort
b) Insertion sort

PRAKTIJK

OPDRACHT 1

Van het college en uit het boek ken je de faculteitsfunctie. Die faculteitsfunctie kan zowel recursief als niet recursief worden geïmplementeerd. Implementeer de faculteitsfunctie zowel recursief als niet-recursief en vergelijk de rekentijd van beide implementaties. Merk je grote verschillen op?

Run de aanwezige unittests.

OPDRACHT 2

a) De theorie geeft aan, dat het in het algemeen niet verstandig is om de reeks van Fibonacci recursief te berekenen. Ga dit na, door voor de eerste 35 elementen uit de reeks het aantal aanroepen te bepalen. Uiteraard schrijf je hiervoor een programma.
b) Implementeer nu een niet-recursieve versie van dit algoritme.
c) Run de aanwezige unittests.

OPDRACHT 3

Gegeven een functie die de som van een aantal getallen "om en om" uitrekent. Bijvoorbeeld:

f(5) = 5 + 3 + 1 = 9
f(6) = 6 + 4 + 2 = 12
f(7) = 7 + 5 + 3 + 1 = 16 etcetera.

a) Maak de recursieve functie `int OmEnOm(int x)`. Deze method berekent de som van een aantal getallen "om en om" volgens bovenstaand principe.
b) Zorg dat bij een negatieve input de exceptie `OmEnOmNegativeValueException` wordt afgevuurd.

c) Run de aanwezige unittests.

OPDRACHT 4

Schrijf een recursieve method `int Enen(int n)` dat het aantal enen berekent in de binaire representatie (http://nl.wikipedia.org/wiki/Binair) van een niet negatief getal N.

Bijvoorbeeld:

- Het getal 7 wordt binair weergegeven als 111, dus Enen(7) = 3
- Het getal 9 wordt binair weergegeven als 1001, dus Enen(9) = 2
- Het getal 16 wordt binair weergegeven als 10000, dus Enen(16) = 1

Maak bij het schrijven van de functie gebruik van het feit dat het aantal enen gelijk is aan het aantal enen in N DIV 2, plus 1 als N oneven is:

Enen(9) = 1 + Enen(9 DIV 2) = 1 + Enen(4) = 1 + Enen(4 DIV 2) = 1 + Enen(2) = 1 + Enen(2 DIV 2) = 1 + Enen(1) = 1 + 1 = 2

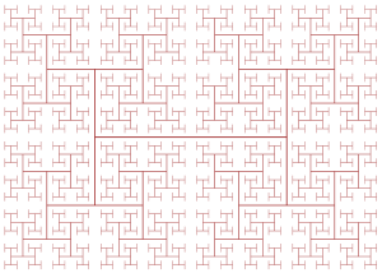
Je mag er vanuit gaan dat $N \geq 0$.

Run de aanwezige unittests.

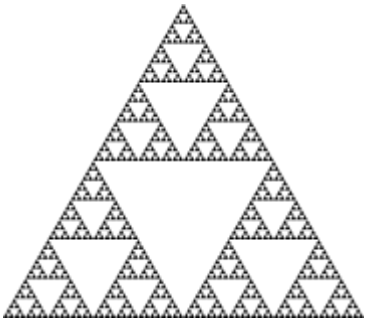
OPDRACHT 5

Maak een programma dat één (of meerdere natuurlijk!) van de onderstaande fractals kan tekenen (of een andere naar keuze):

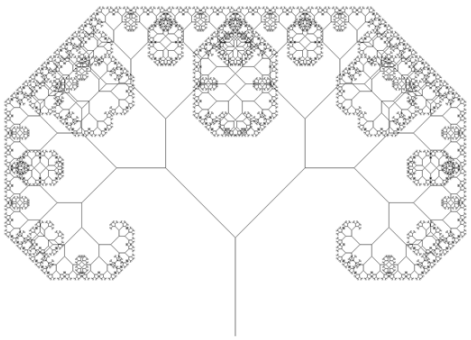
- H Tree (http://en.wikipedia.org/wiki/H_tree):



- Sierpinski driehoek (http://en.wikipedia.org/wiki/Sierpinski_triangle):



- Boom van Pythagoras (http://www.ies.co.jp/math/java/geo/pytreea/pytreea.html):



De boom is, zoals je ziet, getekend door horizontale resp. verticale lijnen door schuine lijnen te verbinden. Steeds is de lijn in x-richting, in y-richting of in beide richtingen (in geval van een schuine lijn) kleiner gemaakt door de oorspronkelijke lengte met een bepaalde factor (kleiner dan 1) te vermenigvuldigen. Als stopcriterium zou je een depth kunnen gebruiken, maar je zou bijvoorbeeld ook de lengte van de te tekenen lijn kunnen nemen: als een lijntje bijv. 1 pixel lang is geworden wordt het tijd om te stoppen.

OPDRACHT 6

Om over een lijst te itereren, gebruiken we meestal een for-loop of een soortgelijke constructie. In deze opdracht zullen we echter *recursief* door een lijst lopen. Doorgaans is dat niet de aangewezen manier, maar het is zinvol om het toch een keer uit te programmeren.

Implementeer de volgende method:

```
string ForwardString(List<int> theList, int i)
```

Als `theList` de elementen 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 bevat, dan is het resultaat van `ForwardString(theList,3)` de string "3 4 5 6 7 8 9".

Implementeer ook de volgende method:

```
string BackwardString(List<int> theList, int i)
```

Als `theList` de elementen 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 bevat, dan is het resultaat van `BackwardString(theList,3)` de string "9 8 7 6 5 4 3".

Je mag in de implementatie van beide functies geen extra variabelen of lussen (`for`, `while`, etc) gebruiken.

Run de aanwezige unittests. De unittests zijn zo geschreven dat eventuele extra spaties worden genegeerd (een uitkomst van "1 2 3 " is dus hetzelfde als "1 2 3").

OPDRACHT 7

Implementeer de sorteeralgoritmen InsertionSort, MergeSort, ShellSort.

Er zijn unittests bijgeleverd om je oplossing te testen.

Test de algoritmen met verschillende inputs, van heel klein tot heel groot.