**qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm**

|  |
| --- |
| Functioneel Programmeren  Minorvak  30-1-2017  Thom Wouda & Eden Cox |

Inhoud

[1. Inleiding 2](#_Toc474280295)

[2. Het algoritme 3](#_Toc474280296)

[3. De Talen 4](#_Toc474280297)

[3.1 C++ 4](#_Toc474280298)

[3.2 Haskell 4](#_Toc474280299)

[4. Imperatief vs Functioneel 5](#_Toc474280300)

[5. Linq 6](#_Toc474280301)

[6. Java 7](#_Toc474280302)

[7. Haskell 8](#_Toc474280303)

# Inleiding

Voor het vak functioneel programmeren wordt een algoritme naar keuze uitgewerkt in drie verschillende talen. De eis dat er een taal imperatief, functioneel en semi-functioneel is. De functionele taal is hierbij vastgesteld op haskell. Tevens mag het algoritme niet te eenvoudig zijn. In dit verslag worden de drie verschillende codes naast elkaar gezet en worden de verschillen tussen functioneel en imperatief beoordeeld.

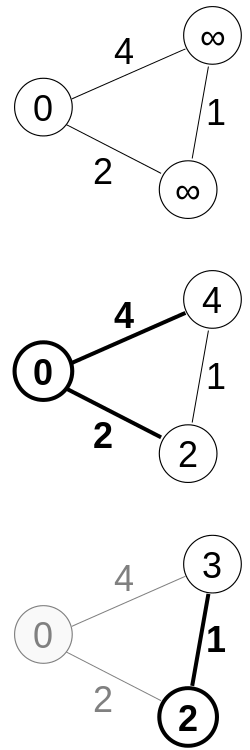
Deze opdracht is uitgevoerd door Eden Cox en Thom Wouda in de periode september 2016 – februari 2017.

# Het algoritme

Voor de realisatie van de opdracht dient er een algoritme gekozen worden. Dit moet een degelijk algoritme zijn, dus niet te eenvoudig. In deze uitwerking is er gekozen voor het Dijkstra algoritme, ook bekend als het korstepad-algoritme.

Het Dijkstra algoritme is een algoritme dat gebruikt wordt om het kortste pad in een graaf te vinden. Hierbij heeft graaf heeft verschillende punten met een begin- en eindpunt. De punten hebben een bepaalde afstand. Het dijkstra algoritme loopt door deze graaf heen en telt de afstanden op, en bepaald aan de hand van de hoogste en laagste afstand welk pad het korst is.

*Een klein voorbeeld van het Dijkstra algoritme is te zien in Figuur 1.*



Figuur 1 - Dijkstra algoritme

# De Talen

In dit verslag wordt het algoritme in drie verschillende talen met elkaar vergeleken. De eis is dat er een imperatieve, functionele en semi-functionele taal wordt gebruikt. Waarbij semi-functioneel een mix is tussen imperatief en functioneel. De functionele taal staat hierbij vastgesteld op **Haskell**.

De imperatieve en semi-functionele talen zijn optioneel, zolang ze maar aan de eis voldoen. In dit verslag is er gekozen voor **C++**(imperatief) en **Scala**(functioneel).

## C++

C++ is een imperatieve, objectgeoriënteerde programmeertaal dat gebaseerd is op C. Het is ontworpen door Bjarne Stroustrup als upgrade van C. De belangrijkste toevoegingen van C++ ten opzichte van C is de beschikbaarheid van onder ander: virtuele functies, abstracte klassen en overerving. Verschillende talen zijn gebaseerd op C++, zoals Java en C#.

## Haskell

Haskell is een functionele programmeertaal vernoemd naar de wiskundige Haskell Brooks Curry. De taal is in 1990 verschenen. Haskell is multi platform en wordt voornamelijk gecompileerd in de GHC compiler.

## Scala

# Imperatief vs Functioneel

# De verschillen

In dit verslag worden bepaalde verschillen van deze drie talen onderzocht. De verschillen die onderzocht worden zijn als volgt:

* De snelheid
* De code structuur
* Leesbaarheid

Bij de uitvoering van Dijkstra in Linq wordt er begonnen met het vullen van een graaf. De punten worden aangemaakt en toegevoegd. Vervolgens worden er connecties gemaakt tussen deze punten. Daarbij wordt een bepaalde afstand bij toegevoegd.