# Notities bij Ontwerp

Bijgevoegd een document met code voor het Globescope Assesment

In het assesment wordt gevraagd om een programma te maken, waarmee minimaal 10 specifieke vragen beantwoord kunnen worden. Dit programma is er op gericht om deze 10 vragen te beantwoorden, maar er kunnen ook gelijkaardige vragen mee beantwoord worden met iets andere input.

Ik ben begonnen om een gemeenschappelijke deler tussen de 10 vragen vast te stellen. Binnen de vragen kunnen een aantal eenheden gedefinieerd worden:

* Node = plaats die geidentificeerd wordt met een letter van het alfabet
* Edge = verbinding tussen twee nodes
* Weight = afstand van de verbinding tussen twee nodes
* Route = georderde lijst van afwisselend nodes en edges, die de verbinding tussen een start node en een eind node vormen.

Dit vormde een uitgangspunt bij de opzet van de progamma code.

**Uitleg bij zoekalgoritme:**

Het algoritme start met het onderscheid tussen twee typen routes.

* Hoofdroutes, met een begin- en een eindpunt.
* Extra routes, met het zelfde begin en eindpunt.

Het begint met de hoofdroutes van begin- naar eindpunt. Begin en eindpunt kunnen gelijk zijn, maar ook verschillen van elkaar. Deze routes kunnen uitgebreid worden met de extra routes, waarbij de route van het eindpunt naar het eindpunt loopt. In principe kan dat tot in het oneindige gebeuren.

De functie routes worden alle mogelijke routes tussen een begin- en een eindpunt bepaald. De functie routes wordt gestart met een beginpunt. Vanuit dit beginpunt worden voor alle mogelijke verbindingen een route gestart. Bij elk tussenpunt gebeurt dit.

Een route stopt zodra:

* Eindpunt bereikt is.
* De knoop geen verdere verbinding heeft.
* Een knoop bereikt is die al eerder gepasseerd in de route

Als een route het eindpunt bereikt dan wordt eventueel routes eruit gefilterd waarbij tussenpunten incorrect zijn.

Dit doe je voor zowel de hoofdroutes als de extra routes. Met de hoofdroutes en de extra routes ga je verder in de class Rfinder. Hier kun je verder de vragen mee oplossen

Voor vragen over de kortste route hoef je niet verder te kijken dan de hoofdroutes. Voor de andere vragen geldt dat een route langer mag zijn. De uitbreiding gebeurd met de subroutes.

De hoofdroutes kunnen uitgebreid worden met subroutes. Deze subroutes worden geplakt achter de hoofdroute.

**Programma opzet**

De progamma code heb ik opgesteld in Python, omdat ik veel ervaring heb met deze taal.

De code bestaat uit 4 bestanden:

* Main4.py: rechtstreeks aanroepbare functies
* Rf\_input.py: input
* Rf\_processing: algoritme voor verwerking gegevens
* unittest\_gsa4.py: (unit)test voor het programma.

In principe worden het programma gestart met functies uit het main4.py bestand.

**Module Rf\_input**

In Rf\_input zijn de basis classes “Node” en “Edge” gedefinieerd. De edges en nodes worden samengesteld uit de graph. informatie, graph informatie wordt omgezet naar node en edge-objecten via een aparte functie.

In rf\_input worden 2 variabelen voor rf\_processing gegenereerd:

1. functie make\_nodeorder: wordt een list nodeorder
2. Nodescoll object.

Nodescoll-object bevat uit de alle unieke nodes in de graphdef.

Met de functie make\_nodeorder word een list met nodes op volgorde van een begin node via tussen nodes, naar een eind node.

**Module rf\_processing**

In rf processing is de class Route gedefinieerd.

De verwerking in rf\_processing is gesplitst:

Functie routing: De functie om hoofdroute en de subroutes te maken.

Class Rfinder: voor het beantwoorden van specifieke vragen op basis van een zoals de 10 assesment vragen.

De functie routing maakt een list met Route -objecten aan. De route heeft naast een aantal functie , twee attributen. De attribuut route bevat een list met een afwisseling van Node en Edge objecten. Met de attribuut end wordt aangeven of het einddoel van de route al dan niet bereikt is.

Rfinder heeft verschillende methodes voor het beantwoorden van vragen.

* return\_shortestdistance
* return\_countmaxstops
* return\_countnrstops
* return\_countmaxdist

Voor deze functie moet minimaal een nodeorder opgegeven. De methoden roepen allemaal routing aan.

Op basis van de lijst aan routes worden vragen beantwoord.

**Module main4**

Deze module is bedoeld om een programma op te zetten. De modules rf\_input en rf\_processing worden geïmporteerd en kunnen gebruikt worden om een volwaardig programma te maken. Als voorbeeld heb ik een paar functies gemaakt.

**(Unit)test**

De unittest wordt gestart door het aanroepen van het bestand unittest\_gsa4.py

Met de test worden de uitkomsten van de 10 vragen van de assesment getest en een paar testen voor rf\_input .

**Bekende beperkingen/bugs**

* rf\_input-Edge-class: Find\_node method: niet uitgewerkt voor inout =’out’
* rf\_input- Edge-class Find\_node method: mogelijkheid bestaat dat noderes niet gedefinieerd wordt.
* Rf\_processing-Rfinder : Implementatie \_no\_such\_route is afwijkend voor methode return\_ shortestdistance anders dan voor de andere methodes in Rfinder
* Rf\_processing-Rfinder Het onbeperkt uitbreiden met subroutes is verwerkt in countmaxdist Eigenlijk hoort dit in de routing functie geïmplementeerd te worden, zodat het toegankelijk is voor alle functies Rfinder.
* Nog geen specifieke testen voor rf\_processing ontwikkeld.
* Algemeen: Antwoorden op andere vragen dan de 10 uit het assesment zijn niet geverifieerd.