# Verslag labo 1

1. **Question 1: DFS**

In de eerste opdracht werd er gekozen om het algoritme recursief te implementeren. Omdat het algoritme reeds gekend was, werd er online gezocht naar een oplossing in pseudocode. Hierna werd deze pseudocode omgezet in python en toegepast op de specifieke situatie. Het idee achter DFS is om eerst een volledige tak (branch) te volgen en het eerste pad dat naar het doel gevonden wordt als oplossing te nemen. Het spreekt voor zich dat deze oplossing niet ideaal zal zijn. Het recursieve algoritme begint bij de root node en zal iedere bekeken node als bezocht markeren zodat dezelfde locatie geen tweede keer bezocht kan worden. Vervolgens wordt de juiste bewegingsrichting toegevoegd aan het pad. Indien de goal state (lees: locatie met een dot) bereikt is, zal de recursieve functie True returnen. Indien dit niet het geval is, zal de functie opgeroepen worden op al zijn successors die nog niet bekeken zijn. De successors zijn alle locaties (en oa. de richting ten opzichte van de huidige locatie) waar Pac-Man naar zou kunnen bewegen als hij op de huidige node stond. Als een onderliggende node de goal gevonden heeft, zal deze True returnen en zal bijgevolg de node erboven (zijn parent node dus) ook True returnen. Als het einde van een tak bereikt wordt, dan wordt er False gereturned en zal de parent node de bovenste richting van het pad poppen zodat onnodige richtingen verwijderd worden. In de code zelf worden de stappen in meer detail toegelicht als commentaar. Er werd gebruik gemaakt van een Stack om het gewenste LIFO gedrag te verkrijgen.

1. **Question 2**

Placeholder.

1. **Question 3**

Placeholder.

1. **Question 4: A\* search**

A\* zal een optimaal pad zoeken en daarbij de laagste cost + heuristic als metric nemen. Op dit moment wordt er echter een nullHeuristic gebruikt, dit wil zeggen dat de heuristiek altijd nul zal zijn waardoor enkel de cost een rol speelt. De volgende node die telkens bezocht zal worden, is de node met de laagste waarde. Om dit te kunnen bepalen, zal er een PriorityQueue gebruikt worden. Bij het opvragen van een element (pop()) zal telkens de node met de laagste waarde (cost + heuristic) gegeven worden. Dit werd niet recursief opgelost waardoor er een loop gebruikt wordt totdat er een pad gereturned wordt. Ook hier werd er eerst naar een pseudo code gezocht die vervolgens geïmplementeerd moest worden in dit probleem. De cost zal telkens weergeven wat de cost is om tot een bepaalde node te geraken terwijl de heuristiek ons kan vertellen hoe wenselijk een node is om tot bij het uiteindelijke doel te geraken.

1. **Question 5**

Placeholder.

1. **Question 6**

Placeholder.

1. **Bonus question 7: Eating all the dots**

Placeholder.

1. **Bonus question 8: Suboptimal search**

Placeholder.