departement Computerwetenschappen



Projec

t 1: Ve

Toepassingen van meetkunde in de informatica

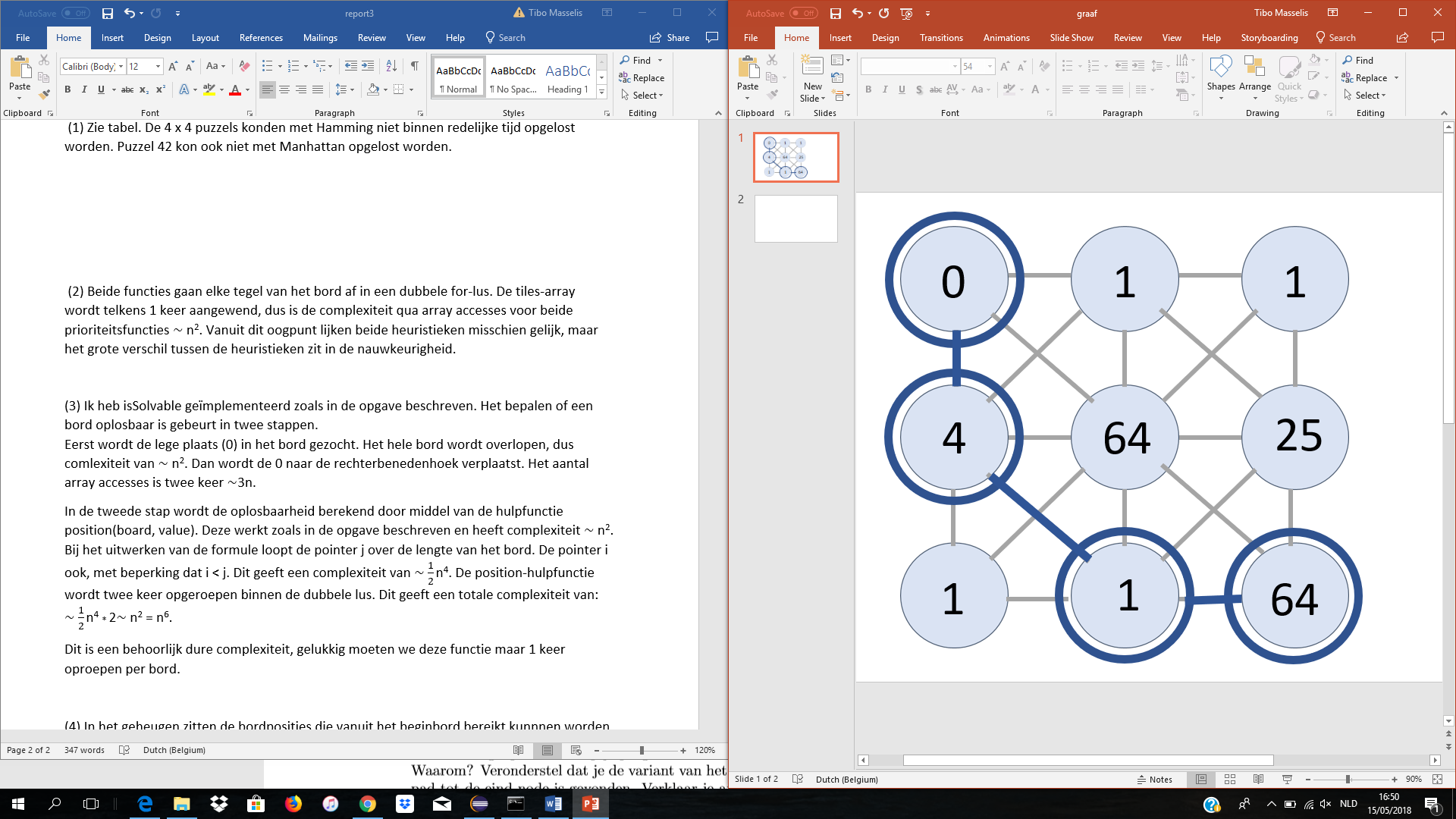
# Project : Bepaling van het Dichtste Puntenpaar

**Tibo Masselis r0638667**

**Barbara Ameloot r0669387**

(1) De gewichten zijn gemakkelijk te berekenen, gezien de tweede afbeelding volledig wit is en elke pixel maar één kleurwaarde heeft. Hieronder zijn de gewichten in de knopen weergegeven.

Het resulterende kortste pad loopt steeds van de linkerbovenhoek (0 in de grafe) naar de rechterbenedenhoek. Het is hieronder aangegeven in het donkerblauw.



(2) Het weglaten van een bepaalde kleur bij het berekenen van de afstandsfunctie heeft interessante gevolgen. Neem bij voorbeeld de kleur blauw, zoals in de opgave. Stel dat we twee afbeeldingen willen samenvoegen: op één staat de zee, op de ander de lucht, met de horizon als de logische seam. De zee en de lucht zijn beiden (typisch) erg blauw, dus soms moeilijk van elkaar te onderscheiden. Daarom zal het algoritme het niet makkelijk hebben om een goede seam te vinden.  
Als we blauw nu weglaten, zal er een duidelijker verschil zijn tussen de lucht en zee. Deze laatste is vaak wat groenig, terwijl dit niet zo is bij de lucht. Dit maakt het makkelijker voor het algoritme om de horizon te vinden, en dus zal het minder tijd kosten.

Het omgekeerde kan natuurlijk ook voorvallen. Stel dat we een afbeelding die overwegend paars is, willen combineren met een afbeelding die vooral rood is. Met deze alternatieve afstandsfunctie zouden de paarse kleuren in rood ‘veranderen’, en daardoor zal het verschil tussen beide afbeeldingen minder duidelijk zijn, dit maakt het moeilijker voor het kortste-pad algoritme, dat meer punten overloopt en dus een langere looptijd zal hebben.

(3) De tijdscomplexiteit van het algoritme op een 1xN (of Nx1) afbeeldingen is lineair (~n), sneller dan bij een normale NxN afbeelding. Dit is eenvoudig te verklaren. Bij een afbeelding van slechts 1 pixel breed, heeft de seam functie maar één keuze: de seam loopt over de hele afbeelding want bij het vinden van het kortste pad heeft elke pixel maar 1 buur die nog niet in de seam zit. Dit in tegenstelling tot de NxN, waar elke pixel gemiddeld veel meer buren heeft (max. 8 – 1 = 7) die nog niet zijn overlopen.

(4) Om te voorkomen dat de seam complexe vormen aanneemt, passen we de getNeighbours() methode aan. Deze geeft normaal alle mogelijke buurposities van een pixel (max. 8). Als we de code aanpassen zodat bepaalde buren niet worden bekeken, worden deze posities niet als buren beschouwd en kan de seam dus niet naar deze buren lopen.

Om te verkomen dat de seam terug naar boven kan lopen, verwijderen we upLeft, up en upRight.   
Om te verkomen dat de seam terug naar links kan lopen, verwijderen we upLeft, left en downLeft.

(5) Het langste pad dat niet meer dan 1x eenzelfde node bezoekt kunnen we eenvoudig vinden met wat inzicht. Het langst mogelijke pad in de grafe is het pad dat alle nodes bezoekt. De manier waarop maakt niet uit. Gezien elke node op deze manier tot de seam behoord, en de seam wordt opgevuld met SEAM.IMAGE2, zal de stitch van image1 en image2 gewoon gelijk zijn aan image2.