Schakelprogramma: Master in de toegepaste informatica

Gegevensstructuren en algoritmen

Practicum 3

Dries Janse
r0627054
24 mei 2019

Inhoud

1. Inleiding

Dit verslag is geschreven naar aanleiding van het derde practicum van het opleidingsonderdeel: Gegevensstructuren en algoritmen. Dit als onderdeel van het schakelprogramma: Master in de toegepaste informatica.

In dit practicum bespreek ik mijn implementatie van het Image Compositing algoritme en geef ik een antwoord op de vragen gesteld in het opgavedocument.

2. Kortste pad gegeven twee afbeeldingen

Volgende twee afbeeldingen zijn gegeven. Elke afbeelding is opgebouwd uit een raster van kleuren, waarbij elke kleur in de vorm (rood,groen,blauw) is:

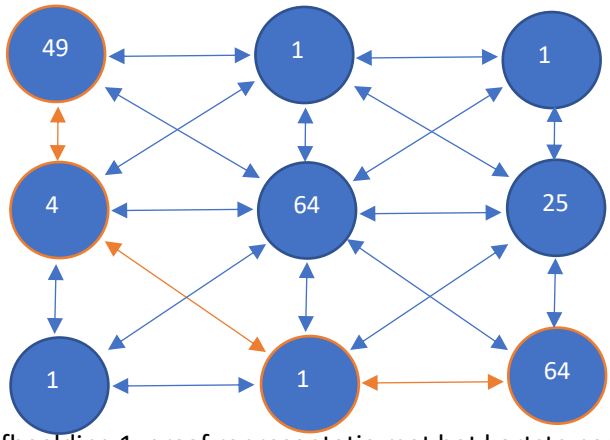
(7,0,0)	(0,1,0)	(0,0,1)
(2,0,0)	(0,8,0)	(0,0,5)
(1,0,0)	(0,1,0)	(0,0,8)

Tabel 1: gegevens afbeelding 1

(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(0,0,0)

Tabel 2: gegevens afbeelding 2

Uit deze twee afbeeldingen kan een graaf geconstrueerd worden. Als gewichten gebruiken we het kwadraat van de afstand tussen twee kleuren. Dit geeft weer hoe hard twee keuren verschillen.



Afbeelding 1: graaf representatie met het kortste pad

Afbeelding 1 geeft de gericht, gewogen graaf weer die het resultaat is van de twee gegeven afbeeldingen. In deze applicatie willen we het kortste pad vinden vanuit de knoop in de linkerbovenhoek tot de knoop in de rechterbenedenhoek. Het kortste pad is weergegeven met oranje de pijlen.

3. Andere afstandsfunctie

In de vorige sectie werd er gebruikt gemaakt van de euclidische afstand met steeds de drie kleur waarden: $\sqrt{(r_x - r_y)^2 + (g_x - g_y)^2 + (b_x - b_y)^2}$. Nu wordt er de vraag gesteld of er een verschil is als we de euclidische afstand berekenen met enkel de eerste twee kleurwaarden: $\sqrt{(r_x - r_y)^2 + (g_x - g_y)^2}$?

Het maakt wel degelijk uit of we alle punten in rekening brengen of enkel de eerste twee. Er kunnen afbeeldingsparen gevonden worden waarop het programma met de nieuwe afstandsfunctie een snellere of tragere uitvoeringstijd heeft. We tonen dit aan door middel van twee voorbeelden. In onderstaande voorbeelden maak ik gebruik van dezelfde conventie als deze in de opgave om posities van een afbeelding voor te stellen, namelijk: (y,x). Waarbij het eerste element van de tupel de y-coördinaat is het tweede element de x-coördinaat.

(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(0,0,0)

(25,25,0)	(0,25,0)	(0,25,25)
(0,5,0)	(0,0,500)	(0,0,0)
(0,5,0)	(0,5,0)	(0,0,0)

Tabel 3: snellere uitvoeringstijd dan de standaard afstandsfunctie

In tabel 3 is een voorbeeld gegeven waarbij de uitvoeringstijd, van de afstandsfunctie die enkel gebruik maakt van de rode en groene kleur, sneller is. Doordat het geen gebruik maakt van de derde waarde, is het kortste pad de diagonaal: (0,0)-(1,1)-(2,2). Dijkstra's algoritme is tot deze oplossingen kunnen komen na twee stappen.

De standaard afstandsfunctie gaat een langer pad construeren: (0,0)-(1,0)-(2,0)-(2,1)-(2,2). Om tot deze oplossing te komen heeft het algoritme 4 stappen nodig.

(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(0,0,0)

(25,25,0)	(5,0,500)	(0,0,500)
(0,0,500)	(5,5,0)	(0,0,505)
(0,0,500)	(0,0,505)	(0,0,0)

Tabel 4: tragere uitvoeringstijd dan de standaard afstandsfunctie

Omgekeerd aan tabel 3, kan een voorbeeld gegeven worden waarbij de uitvoeringstijd, van de afstandsfunctie die enkel gebruik maakt van de rode en groene kleur, trager is. Doordat het geen gebruik maakt van de derde waarde gaat het 4 stappen nodig hebben. Het geconstrueerde kortste pad is: (0,0)-(1,0)-(2,0)-(2,1)-(2,2). De standaard afstandsfunctie kan dit al in 2 stappen, namelijk: (0,0)-(1,1)-(2,2).

We kunnen dus besluiten dat als we een gegeven niet in rekening brengen dat er dan de mogelijkheid bestaat om tot een andere oplossing te komen. Dit kan ook gebruikt worden om bijvoorbeeld een gegeven sterker te laten doorwegen dan een ander gegeven. Bijvoorbeeld, willen we dat het verschil in rode kleur harder doorweegt dan de groene en blauwe kleuren. Dit kunnen we doen door de rode waarde te vermenigvuldigen met een positieve constante.

4. Tijdscomplexiteit 1 x N

De tijdscomplexiteit voor afbeeldingen van de vorm 1xN of Nx1 is $\sim N$ omdat de implementatie gebruik maakt van een priorityqueue die steeds maar 1 element bevat. Dijkstra's algoritme neemt steeds de volgende positie. Het gaat niet terug naar een positie die het al berekend heeft. Het kiest dus steeds het optimale (enige) pad.

Om de worst-case tijdscomplexiteit te bepalen voor afbeeldingen van de vorm NxN beginnen we met het bepalen hoeveel bogen er zijn.

- De 4 hoeken hebben steeds 3 bogen. →4*3
- De buitenste vertices die geen hoeken zijn, hebben steeds 5 bogen. →5*4*(N-2)
- De binnenste vertices hebben steeds 8 bogen. →8*(N-2)*(N-2)

Voor N strik groter dan één kunnen we zeggen dat er $8N^2 - 12N + 4$ bogen zijn. In het slechtste geval geeft dit dan $\sim(8N^2 - 12N + 4)\log_2 N^2$ als tijdscomplexiteit. De tijdscomplexiteit hiervan is een stuk trager, dit komt omdat er meer knopen en meer bogen zijn dan het vorige voorbeeld. Dijkstra's algoritme gaat niet onmiddellijk het kortste pad ontdekken van de doel knoop. Dit gebeurt enkel in het best-case scenario.

5. Seam complexe vormen

De seam methode maakt gebruik van Dijkstra's algoritme. Dijkstra's algoritme gaat bij het ontdekken ("relax") van een knoop steeds alle buren toevoegen aan de priorityqueue. Hierop kunnen extra beperkingen gelegd worden.

Als men wil voorkomen dat de seam naar links of naar boven gaat lopen, kan dit makkelijk aangepast worden. Dit kan geïmplementeerd worden door alle buren toe te voegen behalve deze links en boven. Zo kunnen de paden in deze richtingen nooit ontdekt worden.

6. Langste pad

Het langste pad, dat niet meer dan één keer eenzelfde node bezoekt, is gevraagd. Dit pad kan voorgesteld worden door het pad dat alle nodes precies één keer bevat. Dit kunnen we aantonen omdat een toevoeging van een node (steeds een positief gewicht) het pad gaat verlengen. Elk ander pad, dat niet alle nodes bevat, gaat korter zijn. De door mij geïmplementeerde methoden krijgen steeds het overlappende gedeelte van de foto's. Deze overlapping gaat enkel gelabeld worden als "Stitch.SEAM". Er zijn geen "Stitch.IMAGE1" of "Stitch.IMAGE2" toekenningen. Dit resulteert dus in een foto waarbij het overlappende gedeelte geen kleuring heeft.