Image Compositing Algoritme

Robbe Degrève

1 Bachelor Informatica

Inhoudsopgave

Inhoudsopgave 2

Inleiding 3

1 De sorteeralgoritmen 4

1.1 Selection sort 4

1.2 Insertion sort 4

1.3 Quicksort 5

2 Experimenten 6

2.1 Experiment 1: Relatie tussen grootte en aantal vergelijkingen 6

2.1.1 Het experiment 6

2.1.2 Analyse 7

2.2 Experiment 2: ‘Doubling ratio’ experiment 7

2.2.1 Selection sort 8

2.2.2 Quicksort 9

2.2.3 Een algoritme 10

Besluit 12

Inleiding

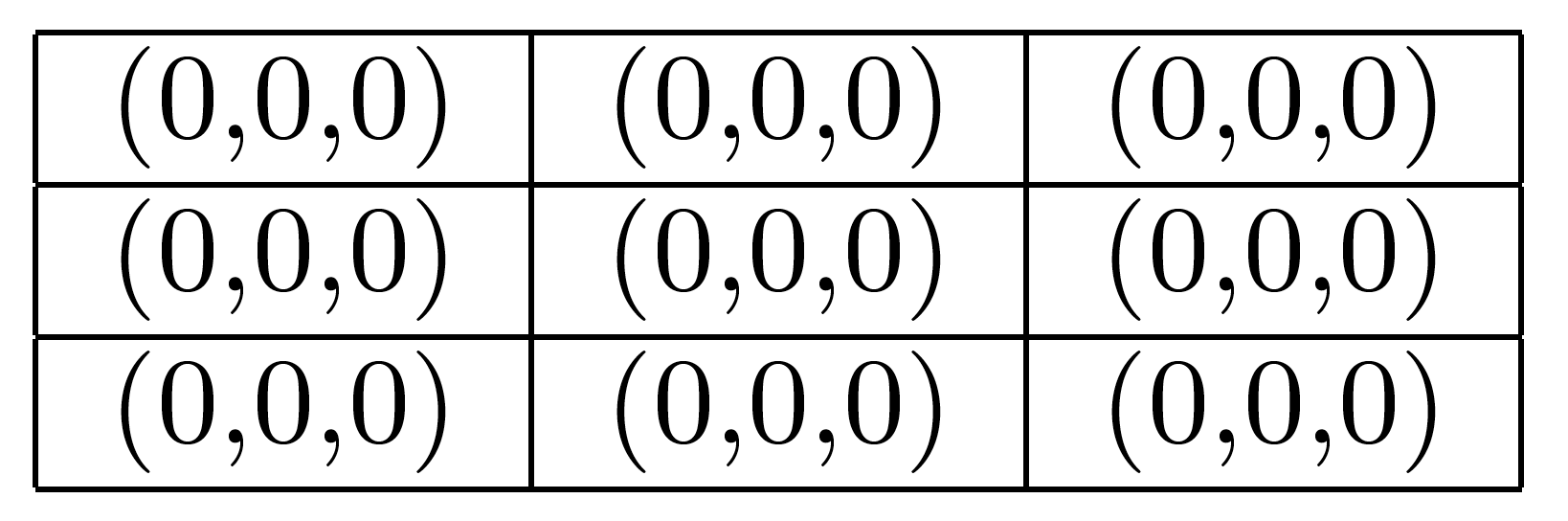
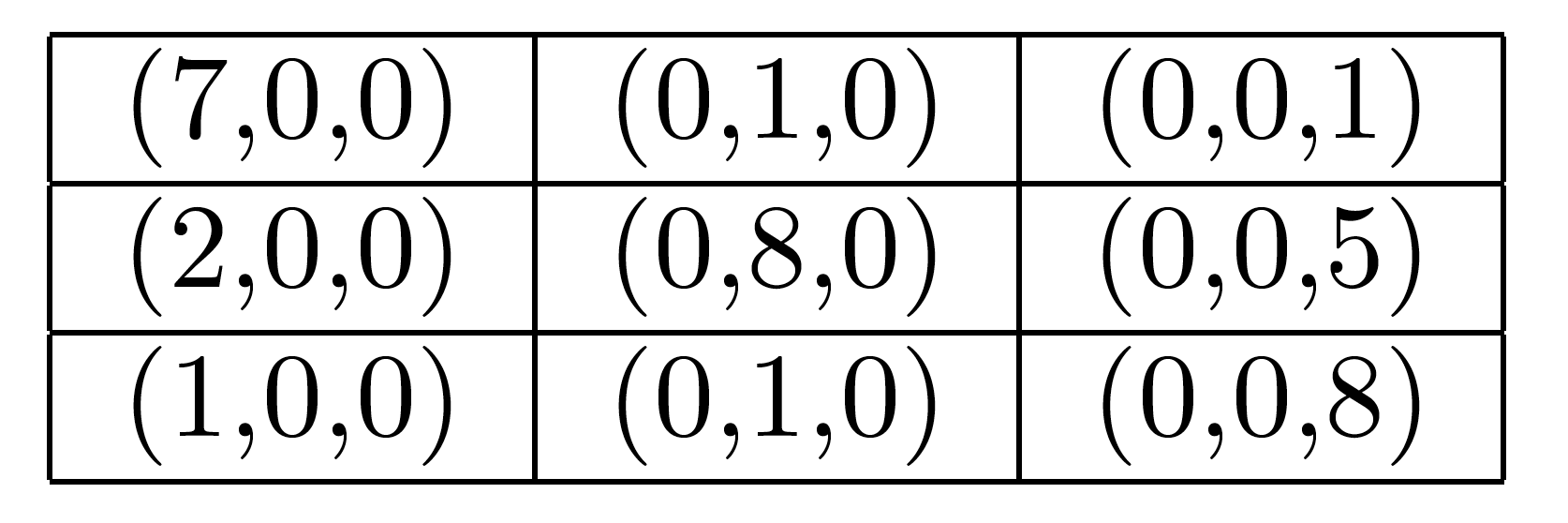
In dit practicum zullen we de efficientie van drie verschillende sorteeralgoritmes, namelijk selection sort, insertion sort en quicksort, bekijken en bespreken. We gaan daarvoor eerst kijken naar de werking van deze algoritmes. Zodra we weten hoe ze werken, voeren we twee experimenten uit om de efficientie ervan te achterhalen.

Bij het eerste experiment staat de relatie tussen de grootte van de invoer en het aantal vergelijkingen centraal. Deze relatie is belangrijk, omdat het aantal vergelijkingen ook de uitvoeringstijd van het algoritme grotendeels zal bepalen. We voeren dit experiment uit voor de drie algoritmes. Het tweede experiment is een ‘doubling ratio’ experiment. We gaan dus kijken met welke factor de uitvoeringstijd verdubbelt als we de grootte van de invoer verdubbelen. Dit experiment voeren we uit of insertion sort en quicksort.

# Afbeeldingen als een graaf

Een populaire manier om afbeeldingen voor te stellen, die ook in dit practicum gebruikt wordt, is een kleur voor te stellen door drie getallen: het eerste getal stelt de hoeveelheid rood voor, het tweede de hoeveelheid groen en het derde de hoeveelheid blauw. De getallen kunnen varieren van 0 tot 255. Iedere pixel heeft dus zijn eigen drie getallen. De afstand tussen deze pixels, en dus de kost van de boog in een graaf tussen de pixels, zullen we berekenen met de formule

Dit is de afstand tussen de pixel en .

Gegeven onderstaande afbeeldingen:

De resulterende grafe voor het kortste pad algoritme is:

//grafe

Het resulterende kortste pad is dan:

met als totaal gewicht .

# Afstandsfunctie en

Besluit

Uit het eerste experiment was al duidelijk dat quicksort veruit het minste aantal vergelijkingen uitvoert, gevolgd door insertion sort en daarna selection sort. Deze waarden gaven echter alleen het aantal vergelijkingen en zijn dus enkel een richtlijn bij het berekenen van de uitvoeringstijd van een algoritme. Het tweede experiment gaf hier dus een antwoord op. We konden een ‘doubling ratio’ berekenen en dus ook een tijdsfunctie opmaken om zo de uitvoeringstijd accuraat te kunnen berekenen voor de machine waarop de experimenten zijn uitgevoerd. De ‘doubling ratio’ blijft wel gelden voor de implementatie van de algoritmes die in het experiment werden gebruikt.

We hebben ook kort gekeken naar de ‘doubling ratio’ van een algoritme, om dit te vergelijken met de andere drie algoritmen.

Om deze experimenten tot een goed einde te brengen, heb ik veel geleerd over de drie sorteeralgoritmen en eventuele fouten die kunnen opkomen bij de implementatie. Daarnaast is het nut en de noodzaak van het ‘doubling ratio’ experiment duidelijk: het geeft ons een zeer accurate uitvoeringstijd voor een bepaald algoritme en dit is handig als efficientie belangrijk is voor een bepaalde toepassing. Het was dus zeker een leerrijke ervaring.