# departement Computerwetenschappen



Projec

t 1: Ve

Toepassingen van meetkunde in de informatica

# Project : Bepaling van het Dichtste Puntenpaar

//TODO: nagaan of grafieken goed zijn (verdeling as, trendlijn, wat erop weergegeven wordt, titel, …)

//Nagaan of verklaringen grondig genoeg en correct zijn

//Controleren op spelling

Barbara Ameloot r0669387

**Tibo Masselis r0638667**

## 1.Hoogniveau beschrijving algoritmen

Het eenvoudig algoritme:



Doorlooplijnalgoritme, eerste variant:



Doorlooplijnalgoritme, tweede variant:



## 2. Opstellen puntenverzameling en worst-case puntenverzameling

Om onze puntenverzameling op te stellen gaan we als volgt te werk:



Om een worst-case puntenverzameling op te stellen moeten alle punten altijd in de strook V liggen. Dit is mogelijk als elk punt dezelfde x-coördinaat heeft. Dan wordt Kgem en zal de uitvoeringstijd maximaal zijn.

## 3. Vergelijking uitvoeringstijd eenvoudig algoritme en eerste variant doorlooplijnalgoritme

//TODO moeten beide algoritmen op dezelfde puntenverzameling worden toegepast? -eigenlijk wel, maar voor groot aantal punten geen probleem

We meten de uitvoeringstijd voor beide algoritmen in functie van een oplopend aantal (2-dimensionale) punten en geven de bekomen resultaten in tabel- en grafiekvorm weer.

Eenvoudig algoritme:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Aantal punten | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 | 8000 | 9000 | 10000 |
| Uitvoeringstijd  (in milliseconden) | 9 | 17 | 22 | 48 | 62 | 100 | 156 | 187 | 231 | 296 |

Eerste variant van het doorlooplijnalgoritme:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Aantal punten | 6250 | 12500 | 25000 | 50000 | 100000 | 200000 | 400000 | 800000 |
| Uitvoeringstijd  (in milliseconden) | 1 | 1 | 4 | 3 | 3 | 4 | 21 | 39 |

Er blijkt een duidelijk verschil: voor 100.000 punten heeft het doorlooplijnalgoritme amper 3 milliseconden nodig, terwijl het eenvoudig algoritme bijna 300 milliseconden nodig heeft voor 10.000 punten.  
 Hieruit besluiten we dat de eerste variant van het doorlooplijnalgoritme sneller is dan het eenvoudig algoritme voor het 2-dimensionale geval.

## 4. Kgem en Kmax in functie van N voor het 2D- doorlooplijnalgoritme

Om te onderzoeken hoe Kgem en Kmax variëren naarmate N varieert, hebben we Kgem en Kmax berekend voor enkele oplopende waardes van N. Om uitschieters tegen te gaan hebben we dit 100 keer herhaald en de gemiddelde waardes genomen.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | 10 | 100 | 1000 | 10000 | 100000 |
| Kmax | 2,26 | 5,15 | 9,8 | 22,29 | 43,95 |
| Kgem | 0,90 | 1,27 | 1,34 | 1,31 | 1,43 |

We zien voor Kmax een duidelijke toename wanneer N toeneemt. Meer bepaald verdubbelt Kmax wanneer N vertiendubbelt. We leiden volgend verband af:

Anders gezegd:

Voor Kgem volgt er uit deze resultaten echter (nog) geen duidelijk verband. Wanneer we het experiment herhalen voor hoger aantal punten krijgen we (als een gemiddelde van 20 experimenten) het volgende resultaat:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 100 | 200 | 400 | 800 | 1600 | 3200 | 6400 | 12800 | 25600 | 51200 | 102400 | 204800 | 409600 | 819200 |
| Kgem | 1,16 | 1,11 | 1,42 | 1,18 | 1,33 | 1,46 | 1,40 | 1,40 | 1,32 | 1,34 | 1,41 | 1,23 | 1,37 | 1,28 |

Hieruit blijkt dat Kgem constant is en niet toeneemt wanneer N toeneemt.

## 5. Uitvoeringstijd doorlooplijnalgoritme in hogere dimensies.

Om te onderzoeken hoe de uitvoeringstijd van het doorlooplijnalgoritme varieert naarmate het aantal dimensies toeneemt, hebben we het algoritme op een puntenverzameling van 10000 punten toegepast. Hierbij hebben we telkens de dimensie laten toenemen van 2 tot 9. Dit experiment hebben we 10 keer herhaald. Hieruit kregen we volgende resultaten:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dimensie | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Uitvoeringstijd  (in milliseconden) | 0 | 3 | 12 | 18 | 38 | 66 | 103 | 164 |

We zien duidelijk dat wanneer het aantal dimensies toeneemt, ook de uitvoeringstijd toeneemt.

## 6. Evolutie Kgem voor stijgend aantal dimensies

Om te onderzoeken hoe Kgem varieert naarmate het aantal dimensies toeneemt, hebben we het doorlooplijnalgoritme op een puntenverzameling van 10000 punten toegepast. Hierbij hebben we telkens de dimensie laten toenemen van 2 tot 9. Dit experiment hebben we 10 keer herhaald. Hieruit kregen we volgende resultaten:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dimensie | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Kgem | 1 | 23 | 86 | 2301 | 404 | 681 | 962 | 1279 |

We zien duidelijk dat wanneer het aantal dimensies toeneemt, Kgem ook toeneemt. Bovendien valt op dat de grafiek van Kgem en de uitvoeringstijd dezelfde stijging vertonen. Dit is te verklaren aan de hand van de tijdscomplexiteit van het algoritme:

Als Kgem toeneemt zal de uitvoeringstijd evenredig toenemen.

## 7. Rekentijden beide varianten doorlooplijnalgoritme

We willen de uitvoeringstijd van de eerste variant van het doorlooplijnalgoritme vergelijken met die van de tweede variant voor het 2-dimensionale geval.

We hebben beide algoritmen getest op een oplopend aantal punten. Hieruit kregen we volgende resultaten:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Aantal punten | 100 | 200 | 400 | 800 | 1600 | 3200 | 6400 | 12800 | 25600 | 51200 | 102400 | 204800 | 409600 | 819200 |
| 1e variant uitvoeringstijd  (in milliseconden) | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 2 | 5 | 3 | 11 | 7 | 21 | 43 |
| 2e variant uitvoeringstijd  (in milliseconden) | 1 | 0 | 3 | 2 | 2 | 4 | 3 | 8 | 5 | 18 | 16 | 30 | 61 | 150 |

Hieruit blijkt dat de uitvoeringstijd van de eerste variant minder snel toeneemt dan die van de tweede variant.

## 8. Speciale gevallen

We hebben bij de implementatie van de algoritmen geen rekening gehouden met het triviaal geval waarbij er slechts 1 punt is.

Er wordt niet gecontroleerd op samenvallende punten, deze zullen dus resulteren in een minimale afstand = 0.