

Toepassingen van meetkunde in de informatica

Project: Bepaling van het Dichtste Puntenpaar

Wolfgang Möllmann & Robbe Degrève

22 mei 2018

1 Beschrijving van opstelling van puntenverzameling

Voor het opstellen van een input-file met een aantal gegeven parameters om willekeurige punten te creëren gebruiken we de functie "makeRandom". Deze functie zal de volgende parameters als input vragen:

1. inFile: De puntenverzameling zal in dit bestand weggeschreven worden, indien het bestand niet bestaat zal het aangemaakt worden.
2. alg: het nummer van het algoritme dat gebruikt moet worden: 1 (eenvoudige algoritme), 2 (eerste variante van het doorlooplijnalgoritme) of 3 (tweede variante van het doorlooplijnalgoritme)
3. dim: de dimensie van de punten $M \geq 2$
4. size: het aantal punten N

Deze functie heeft als taak om een inputbestand op te stellen met willekeurige coördinaten. De functie zal de gegeven parameters eers neerschrijven in de Infile. Daarna gebeurt een initialisatie van Random r. Daarna zullen we per regel van het bestand itereren van 0 tot en met de dimensie en een willekeurige waarde neerschrijven in het bestand. De willekeurige coördinaten kunnen waardes aannemen tussen $[0, 5[$.

Voor de *worst – case* puntenverzameling voor de eerste variant van het doorlooplijnalgoritme voor $M = 2$ hebben we een javafunctie "makeWorst-Case". In deze functie maken een puntenverzameling aan van één kolom, waarbij alle punten dezelfde x-waarde hebben. De y-waarde speelt hierbij geen rol. In onze rij-implementatie sorteren we de punten met dezelfde x-waarden volgens de y-waarde. Het tweede punt zal dus enkel vergelijken met het punt boven hem. Het derde punt zal nadien vergelijken met het eerste en het tweede punt. Uiteindelijk zal het laatste (onderste) punt zal dus met alle punten vergelijken. Dit zal ons uiteindelijk $n(n + 1)/2$ vergelijkingen geven. De complexiteit zal dus $O(n^2/2)$ zijn.

Algorithm 1 De javafunctie "makeRandom"

Input: *inFile*, *alg*, *dim*, *size*

infile.println(*alg*)

if *dim* < 2 **then**

print Dimension needs to be greater or equal to 2.

Exit

end if

Random *r* = new Random()

Iterator < Double > *i* = *r*.doubles(*size* * *dim*, 0.0, 5.0).iterator()

*String**outString* =

while *i.hasNext()* **do**

outString =

for *j* = 0; *j* < *dim*; *j*++ **do**

outString += String.format(Locale.US, "%17.16f ", *i.next()*)

end for

infile.println(*outString*)

end while

inFile.flush()

inFile.close()

2 Opdracht 1: Hoog-niveau beschrijving van verscheidene algoritmes

2.1 Brute-force algorithm

2.1.1 hoogniveau beschrijving

Algorithm 2 Bereken het dichtste Puntenpaar met brute-force

Input: *rij*: Array met N punten (gesorteerd naar stijgende x-coördinaat)
 $d = +\infty$
 $dpp1 = 0, dpp2 = 0$
 $currentDist = 0$
for i to $length(rij) - 1$ **do**
 for $j = i + 1$ to $length(rij)$ **do**
 $currentDist = \text{calculate_dist}(rij[i], rij[j])$
 if $currentDist < d$ **then**
 $dpp1 = rij[i]$
 $dpp2 = rij[j]$
 $d = currentDist$
 end if
 end for
end for
return $dpp1, dpp2, d$

2.2 Variant 1 algorithm

2.2.1 hoogniveau beschrijving

2.3 Variant 2 algorithm

2.3.1 hoogniveau beschrijving

Algorithm 3 Bereken het dichtste Puntenpaar volgens variant 1

Input: rij : Array met N punten (gesorteerd naar stijgende x-coördinaat)
 $d = +\infty$
 $dpp1 = 0, dpp2 = 0$
 $currentDist = 0$
for $i = 1$ to $length(rij)$ **do**
 for $j = i - 1$ to 0 **do**
 if $rij[i].x - rij[j].x > d$ **then**
 $break$
 end if
 $currentDist = calculate_dist(rij[i], rij[j])$
 if $currentDist < d$ **then**
 $dpp1 = rij[i]$
 $dpp2 = rij[j]$
 $d = currentDist$
 end if
 end for
end for
return $dpp1, dpp2, d$

Algorithm 4 Bereken het dichtste Puntenpaar volgens variant 2

Input: rij : Array met N punten (gesorteerd naar stijgende x-coördinaat)
 $d = +\infty$
 $dpp1 = 0, dpp2 = 0$
 $currentDist = 0$
 t : gegevensstructuur waarin punten links van de doorlooplijn opgeslagen zijn, gesorteerd naar stijgende y-coördinaat
