Toepassingen van meetkunde in de informatica
Project: Bepaling van het Dichtste Puntenpaar

Wolfgang Möllmann & Robbe Degrève

24 mei 2018

1 Beschrijving van opstelling van puntenverzameling

Voor het opstellen van een input-file met een aantal gegeven parameters om willekeurge punten te creëren gebruiken we de functie "makeRandom". Deze functiezal 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 willekeurge coördinaten. De functie zal de gegeven parameters eerst neerschrijven in de inFile. Daarna gebeurd een initialisatie van `Random r`. Daarna zullen we per regel van het bestand itereren van 0 tot en met de dimensie en voor iedere dimensie een willekeurige waarde neerschrijven in het bestand. De willeukeurige coördinaten kunnen waardes aannemen tussen $[0, 5[$.

Voor de *worst-case* puntenverzameling voor de eerste variant van het doorlooplijnalgoritme voor $\mathrm{M} = 2$ hebben we een javafunctie `makeWorstCase`. 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 $\frac{(n+1)}{2}$ 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()
StringoutString =
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**
    $currentDist$ = 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

Algorithm 3 Bereken het dichtste Puntenpaar volgens variant 1

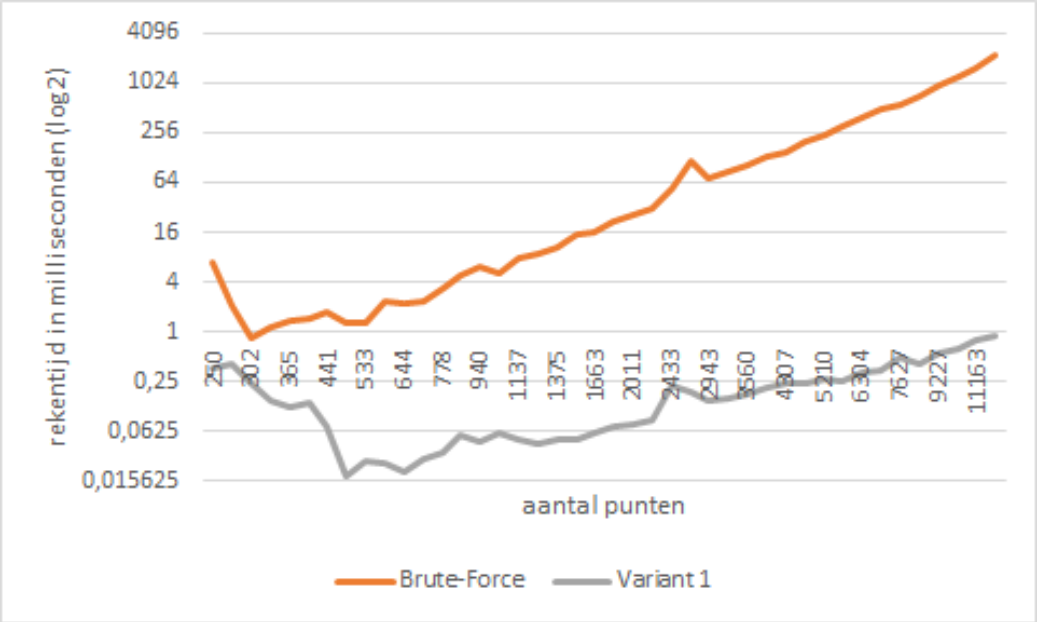
Input: rij: Array met N punten (gesorteerd naar stijgende x-coördinaat)
$d = +\infty$
$dpp1 = 0$, $dpp2 = 0$
$currentDist = 0$
for $i = 1$ to $length(rij)$ **do**
  for $j = i - 1$ to 0 **do**
    if $rij[i].x - rij[j].x > d$ **then**
      $break$
    end if
    $currentDist$ = calculate_dist($rij[i]$,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

2.3 Variant 2 algorithm

2.3.1 hoogniveau beschrijving

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-coordinaat

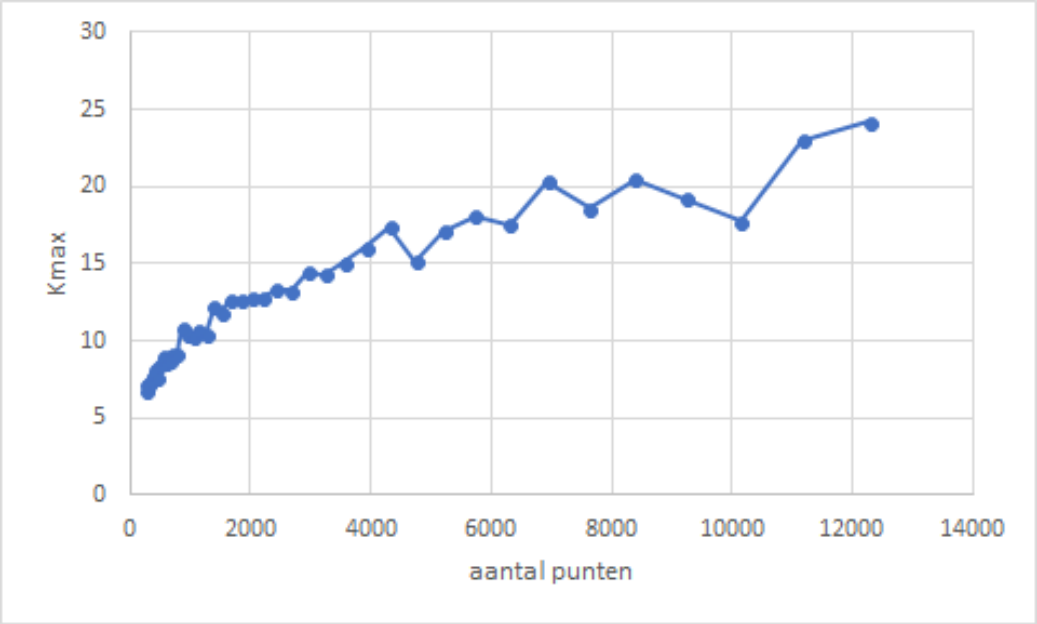


Figuur 1: De plot van de rekentijden tussen het brute-force algoritme en variant 1

3 Grafieken

In Figuur 1 zullen we de rekentijden tussen het brute-force algoritme en variant1 vergelijken. We zullen op de x-as het aantal punten uitzetten. De gekozen aantal punten zijn logaritmisch bepaald met als basis 1.1 beginnend vanaf 302 tot en met 12279. Op de y-as zetten we de rekentijd uit in milliseconden. Hiervoor gebruiken we een logaritmische functie met basis 2. We kunnen concluderen uit deze plot dat variant 1 voor alle onze gekozen punten een effeciënter algoritme is. We kunnen ook zien dat de het brute-force algoritme een steiler verloop heeft, hieruit kunnen we afleiden dat voor zeer grote aantallen punten het verschil tussen de twee algoritmes steeds groter worden.

In figuur 2 zullen we het verband tussen de Kmax en het aantal punten plotten. Op de x-as zetten we het aantal punten uit van 250 tot 12279 met een logaritmische functie met basis 1,1. Op de y-as zetten we de Kmax uit die in dit geval zal gaan van een minimum van 2 tot en met een maximum van 25. We kunnen op de plot zien dat de grafiek Kmax een licht stijgend

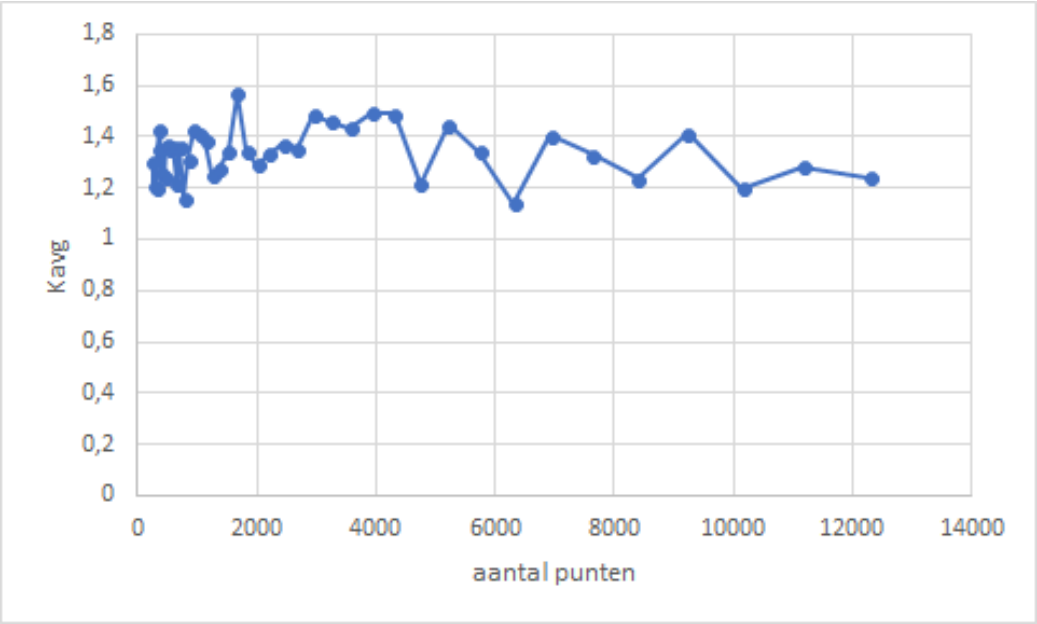


Figuur 2: De plot tussen het aantal punten en Kmax

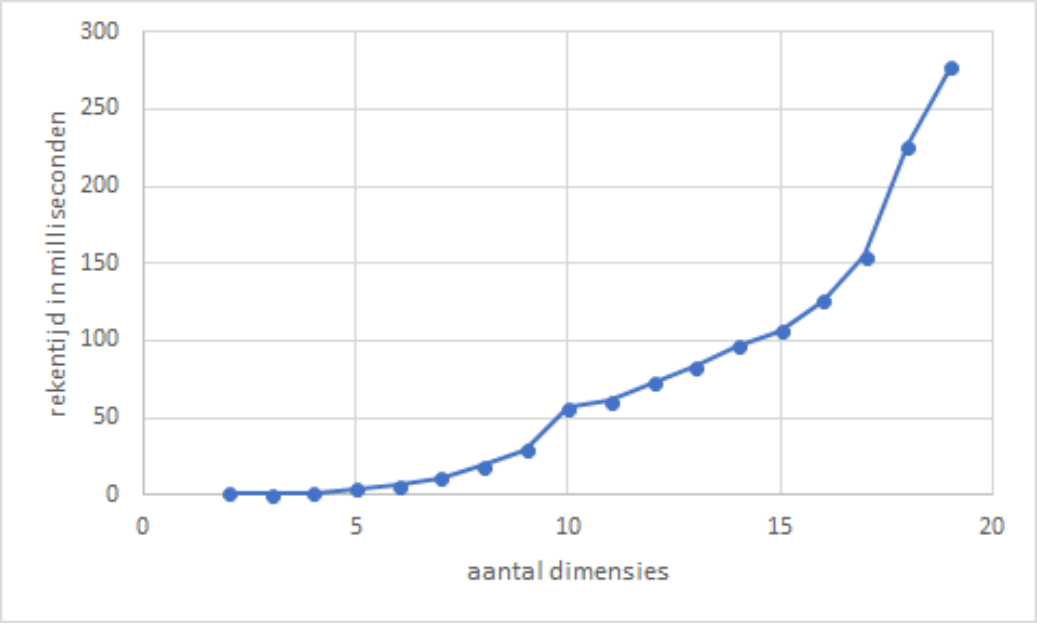
karakter heeft.

In figuur 3 zullen we het verband tussen de Kavg en het aantal punten plotten. Op de x-as zetten we het aantal punten uit van 250 tot 12279 met een logaritmische functie met basis 1.1. Op de y-as zetten we de Kmax uit die in dit geval zal gaan van een minimum van 0,1480 tot en met een maximum van 2,14186. Op deze plot is duidelijk te herkennen dat Kavg ongeveer constant zal blijven ongeacht het aantal punten. In ons experiment was deze constante rond 1,35.

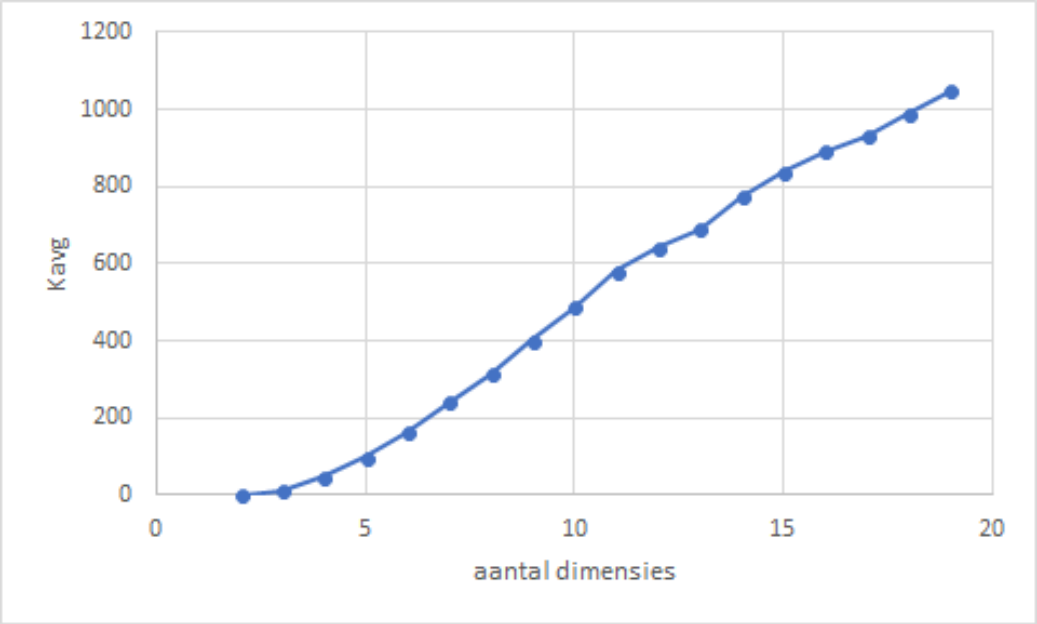
In Figuur 4 zullen we de het aantal dimensies plotten in functie van de rekentijd voor variant 1. We zullen op de x-as het aantal dimensies uitzetten van 2 tot en met 19. Op de y-as zetten we de rekentijd uit in milliseconden.



Figuur 3: De plot tussen het aantal punten en Kavg



Figuur 4: De plot tussen het aantal dimensies en de rekentijd van variant 1 (voor 2500 punten)



Figuur 5: De plot tussen het aantal dimensies en Kavg (voor 2500 punten)