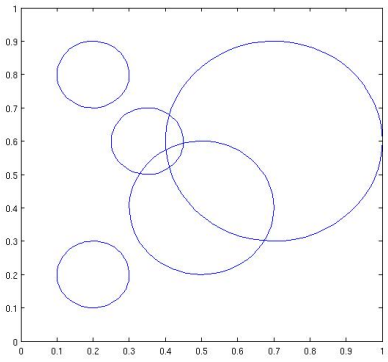
Toepassingen van meetkunde in de informatica

Practicum: Snijdende cirkels

In een vlak liggen N cirkels, waarvan de middelpunten $p_1, \ldots, p_N$ en de stralen $r_1, \ldots, r_N$ gegeven zijn. We willen alle snijpunten tussen de cirkels bepalen. Merk op: een cirkel kan volledig binnen een andere cirkel liggen: deze cirkels snijden elkaar niet.
Ontwerp een algoritme en schrijf vervolgens een programma dat dit probleem oplost:

1. een eenvoudig algoritme met rekencomplexiteit $O(N^2)$,
2. een doorlooplijnalgoritme met een 'slechtste-geval' rekencomplexiteit $O(N^2)$,
3. *(facultatief)* een doorlooplijnalgoritme met rekencomplexiteit $O((N+S)log_2(N))$, met S het aantal snijpunten.



1 Hoogniveau beschrijving van de algoritmen

Geef voor de twee of drie algoritmen een *hoog-niveau*[1] beschrijving.
Een doorlooplijnalgoritme maakt gebruik van een doorlooplijn die zich van links naar rechts door het vlak beweegt. Indien je bijhoudt welke schijven "actief" zijn voor een bepaalde positie van de doorlooplijn (d.w.z. cirkels die de doorlooplijn snijden) kan je het aantal testen op snijding beperken. In het slechtste geval zal toch voor elke cirkel een test op snijding met elke andere cirkel nodig zijn.
Indien je gepaste gegevensstructuren gebruikt, kan de rekencomplexiteit beperkt worden tot $O((N+S)log_2(N))$.

2 Implementatie

Je mag de implementatie maken in een programmeertaal naar keuze.

[1] d.w.z. op de manier waarop algoritmes in de cursus en de oefenzittingen worden voorgesteld, onafhankelijk van een specifieke programmeertaal

2.1 Algoritmen

Implementeer de twee of drie algoritmen om de snijpunten van de cirkels te bepalen. Voor de implementatie van de doorlooplijnstatus mag je gebruik maken van een voorgedefinieerde binaire zoekboom (bijvoorbeeld voor Java `http://algs4.cs.princeton.edu/32bst/BST.java.html`).

2.2 Invoer en uitvoer

Maak een uitvoerbaar bestand (bijvoorbeeld voor Java een executable jar) waarmee de verschillende algoritmen eenvoudig getest kunnen worden.
De invoer wordt gegeven als een txt-bestand dat op de volgende manier is opgebouwd:

- De eerste lijn bevat één getal: het nummer van het algoritme dat gebruikt moet worden $1, 2$ of 3
- De volgende lijn bevat één getal: het aantal cirkels N
- De volgende N lijnen zijn een opeenvolging van drie reële getallen x_i, y_i en r_i van elkaar gescheiden door een spatie. De getallen stellen de x- en y-coördinaat van het middelpunt, gevolgd door de straal van de i-de cirkel voor.

1		
5		
0.200000000000000	0.200000000000000	0.100000000000000
0.500000000000000	0.400000000000000	0.200000000000000
0.700000000000000	0.600000000000000	0.300000000000000
0.350000000000000	0.600000000000000	0.100000000000000
0.200000000000000	0.800000000000000	0.100000000000000

Tabel 1: voorbeeld input.txt

De uitvoer van het gekozen algoritme moet weggeschreven worden naar een txt-bestand dat er als volgt moet uitzien (zie Tabel 2):

- Het bestand bevat slechts één regel: Dit algoritme is niet geïmplementeerd.

  of

- De eerste S lijnen, met S het aantal gevonden snijpunten bevatten telkens de x- en y-coördinaat van een snijpunt, gescheiden door een spatie.
- Een blanco lijn
- De laatste lijn bevat een getal dat de uitvoeringstijd weergeeft in milliseconden.

2.3 Experimenten

De geimplementeerde algoritmen moeten gebruikt worden om enkele experimenten uit te voeren op willekeurige verzamelingen van cirkels.

1. Je werkt best met cirkels waarvan de x- en y-coördinaten van de middelpunten en de stralen tussen 0 en 1 liggen. Je kan hierbij gebruik maken van een randomgenerator die bewegende kommagetallen tussen 0 en 1 genereert. Java levert zulke generator met de methode `random()` in de klasse `java.lang.Math`. In Matlab kan je de functie `rand` gebruiken.

0.673858901432946	0.301141098567054
0.401141098567053	0.573858901432946
0.449794736614283	0.593596052460712
0.328205263385717	0.502403947539288
0.410714285714286	0.679459269504596
0.410714285714286	0.520540730495404

2

Tabel 2: voorbeeld output.txt

2. Begin met een experiment met een beperkt aantal kleine cirkels, bv. $N = 10$, stralen $r_i < 0.1$. Verhoog dan het aantal cirkels zodat de rekencomplexiteit van de verschillende algoritmen duidelijk wordt. Ga na of de berekende oplossing correct is! Herhaal dan de experimenten met grotere stralen, zodat er meer snijdingen optreden.

3 Verslag

Schrijf een verslag waarin je enkel de resultaten van je werk beschrijft (geen opgave herhalen, ...). Geef hierin zeker de volgende dingen weer:

1. Een hoogniveau beschrijving van al de uitgewerkte algoritmen
2. Een beschrijving van hoe je de experimenten hebt opgesteld en hoe je de correctheid van de code hebt nagegaan.
3. Een kritische en grondige *bespreking* van de resultaten van je experimenten. Verklaar ook de rekentijden.

4 Praktisch

De deadline van het project is **donderdag 15 mei** om 14u.

4.1 Code

De code moet geupload worden op Toledo (door 1 persoon van het groepje indien je met twee gewerkt hebt). Zet al de code in één zip-bestand met jullie achternamen in de bestandsnaam,
bv. codeJanssensPeeters.zip. Het zip-bestand moet zeker het gevraagde uitvoerbaar bestand bevatten uit §2.2 waarmee wij je code kunnen testen.

4.2 Verslag

Het verslag moet afgegeven worden in de studentenbrievenbus in 200A. Vergeet niet je naam, het vak en de bestemming (D. Roose) te vermelden! Indien je met twee hebt gewerkt, volstaat één verslag met beide namen op.