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 "brute force" algoritme met rekencomplexiteit $O(N^2)$,
2. een efficiënter doorlooplijnalgoritme met een 'slechtste-geval' rekencomplexiteit $O(N^2)$,
3. een efficiënt doorlooplijnalgoritme met rekencomplexiteit $O((N+S)\log_2(N))$, met S het aantal snijpunten.

1 Integriteitscode

Het practicum zal worden gequoteerd, en het onderwijs- en examenreglement is dan ook van toepassing. Soms zal is er echter wat onduidelijkheid over wat toegestaan is en niet inzake samenwerking bij opdrachten zoals deze.
De oplossing en/of verslag en/of programmacode die ingediend wordt moet volledig het resultaat zijn van werk dat je zelf gepresteerd hebt. Je mag je werk uiteraard bespreken met andere studenten, in de zin dat je praat over algemene oplossingsmethoden of algoritmen, maar de bespreking mag niet gaan over specieke *code* of *verslagtekst* die je aan het schrijven bent, noch over specieke *resultaten* die je wenst in te dienen.
Als je het met anderen over je practicum hebt, mag dit er dus **nooit** toe leiden, dat je op om het even welk moment in het bezit bent van een geheel of gedeeltelijke kopie van het opgeloste practicum of verslag van anderen, onafhankelijk van wie de code of het verslag geschreven heeft (mede-studenten, eventueel uit andere studiejaren, volledige buitenstaanders, internet-bronnen, e.d.). Dit houdt tevens ook in dat er geen enkele geldige reden is om je code of verslag door te geven aan mede-studenten, noch om dit beschikbaar te stellen via publiek bereikbare directories of websites.
Elke student is verantwoordelijk voor de code en het werk dat hij of zij indient. Als tijdens de beoordeling van het practicum er twijfels zijn over het feit of het practicum zelf gemaakt is (bvb. gelijkaardige code, grafieken, of oplossingen met andere practica), zal de student gevraagd worden hiervoor een verklaring te geven. Indien dit de twijfels niet wegwerkt, zal er worden overgegaan worden tot het melden van een onregelmatigheid, zoals voorzien in het onderwijs- en examenreglement (zie `http://www.kuleuven.be/onderwijs/oer/`).

2 Hoogniveau beschrijving van de algoritmen

Het idee achter de drie algoritmen is als volgt:

1. Het "brute force"-algoritme test op alle mogelijke snijdingen zonder bijkomstige optimalisaties.
2. Een doorlooplijnalgoritme maakt gebruik van een doorlooplijn die zich van links naar rechts door het vlak beweegt. Indien je voor dit practicum 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.

3. Met de juiste implementatie (in het bijzonder: de juiste gegevensstructuren) kan de rekencomplexiteit beperkt worden tot $O((N+S)\log_2(N))$.

Je kunt je hiervoor inspireren op andere doorlooplijnalgoritmen die in de colleges of oefenzittingen besproken zijn.

3 Implementatie

Je mag de implementatie maken in Python of Java.

Implementeer de beschreven algoritmen om de snijpunten van de cirkels te bepalen. Je mag gebruik maken van bibliotheken voor vaak voorkomende gegevensstructuren die je uit andere cursussen kent. Daarvoor vind je vele implementaties online. Zo is er bijvoorbeeld de volgende implementatie van een binaire zoekboom: `http://algs4.cs.princeton.edu/32bst/BST.java.html`.
Het is niet noodzakelijk om rekening te houden met alle uitzonderlijke situaties (randgevallen), maar vermeld wel in welke gevallen je algoritme (of implementatie) niet werkt.

3.1 Experimenten

De geïmplementeerde 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 reële getallen tussen 0 en 1 genereert.

2. Experimenteer zelf met de verdeling van cirkels: verschillende verdelingen, dichtheden, bijzondere gevallen, etc. Het is de bedoeling dat je vanuit eigen inzicht kunt voorspellen voor welke soort data je algoritme beter of slechter presteert en dat je die verwachtingen experimenteel kunt bevestigen.

3.2 Afspraken voor uitvoering

We gaan je programma automatisch testen. Besteed dus geen tijd aan een mooie gebruikersinterface en **hou je strikt aan de volgende afspraken** (beschouw het als een API maken).
Maak een programma dat we kunnen uitvoeren met één van de volgende twee commando's (zonder bijkomstige argumenten):

1. Python: `$ python3 main.py`

2. Java: `$ java -jar Main.jar`

De invoer wordt gegeven in een vast bestand `in.txt` 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 vast bestand `uit.txt` dat er als volgt uitziet:

- 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 lijn open laten
- De laatste lijn bevat een getal dat de uitvoeringstijd weergeeft in milliseconden.

We zullen dit programma uittesten op de systemen van het departement computerwetenschappen, die draaien op Ubuntu 18.04 "Bionic Beaver". Test dus goed op voorhand of je code op de werkstations van de computerklassen werkt! Ervaring leert dat inloggen op afstand moeilijkheden meebrengt. Raadpleeg bij problemen de volgende hulppagina: `https://system.cs.kuleuven.be/cs/system/wegwijs/computerklas/`.

4 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 beschrijving van al de uitgewerkte algoritmen, met een argumentatie waarom het correct werkt. Beschrijf het idee in woorden en geef pseudo-code, waarin enkel de begrippen uit dit vak in detail uitgewerkt zijn.
2. Een analyse van de rekencomplexiteit.
3. Een beschrijving van hoe je de experimenten hebt opgesteld (zodat de resultaten reproduceerbaar zijn).
4. Een kritische en grondige *bespreking* van de resultaten van je experimenten. Verklaar ook de rekentijden. Zorg dat je experimenten divers genoeg zijn, zodat het duidelijk wordt in welke gevallen elk algoritme beter werkt dan andere; behalve het totale aantal cirkels zijn er nog tal van parameters waarmee je kunt experimenteren. Denk voor elke grafiek na hoe je het best de assen schaalt (bijv. de x- en/of y-as logaritmisch) om duidelijk de rekencomplexiteit te verifiëren.
5. Een bespreking van in welke randgevallen het algoritme potentieel niet werkt en waarom. Hieronder verstaan we situaties zoals o.a. meer dan twee collineaire punten of meer dan twee snijdende lijnen in hetzelfde punt. Een suggestie voor een oplossing is wenselijk maar niet verplicht.

5 Praktisch

De deadline van het project is **maandag 21 december 2020** om 12u (middag).
Upload op Toledo twee bestanden:

1. Het verslag als een PDF-bestand met als bestandsnaam “verslag”gevolgd door je voor- en familienaam.

2. De code in een zip-bestand met bestandsnaam “code”gevolgd door je voor- en familienaam. Controleer voor het indienen dat het programma precies werkt zoals beschreven in sectie 3.2, met het juiste in-/uitvoerformaat en de juiste bestandsnaam voor het programma. Voeg hierbij de broncode en (indien van toepassing) de JAR-executable.

6 Vragen

Keer je voor vragen naar het discussieforum op Toledo of stel ze tijdens de oefenzitting. Als een vraag potentieel weggeeft hoe je algoritme werkt, kun je deze ook mailen naar nick.dewaele@kuleuven.be.