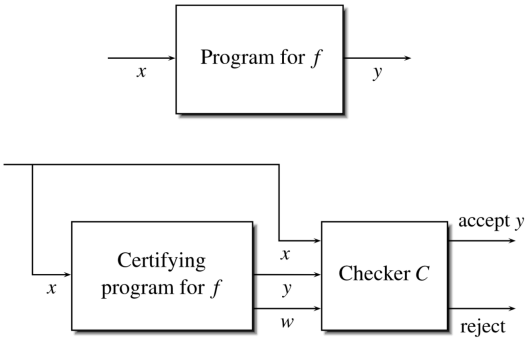
Algoritmen en Datastructuren 2
Taak 1 - Certifying Algoritmes en Grafen

Academiejaar 2023-2024 (1e zittijd)
Assistent: ir. Youri Coppens (Youri.Coppens@vub.be)

De deadline voor deze taak vind je in sectie 4.

1 Inleiding

Traditioneel wordt de correctheid van een geïmplementeerd algoritme aangetoond door het uit te voeren op enkele invoervoorbeelden waarvan de gewenste uitvoer bekend is (*unit testing*). Indien de implementatie de verwachte uitvoer produceert, wordt er van uit gegaan dat het algoritme correct werd geïmplementeerd. Deze aanpak is echter niet waterdicht: bugs in de code worden mogelijk niet vroegtijdig gedetecteerd, omdat niet elke mogelijke invoer in de testvoorbeelden kan worden opgenomen. *Certifying algoritmes* bieden een praktische oplossing voor dit probleem. De idee is dat de implementatie van het algoritme wordt uitgebreid: naast het berekenen van het gewenste resultaat dient het programma een *certificaat* (getuigenis) op te leveren. In combinatie met de in- en uitvoer van het algoritme, wordt dit certificaat vervolgens door een tweede programma gecontroleerd, die de *checker* wordt genoemd. Het is de checker die zal concluderen of (de implementatie van) het algoritme al dan niet correct werkt. Figuur 1 illustreert dit principe. Belangrijk is dat het certifying algoritme, inclusief de checker, dezelfde asymptotische tijdscomplexiteit heeft als het originele algoritme. Het opstellen van certificaten en checkers is specifiek voor ieder algoritme, maar wordt onderbouwd vanuit een theoretisch kader. Er bestaat een heel onderzoeksdomein binnen de computerwetenschappen die certificeringsversies van algoritmen probeert op te stellen, waaronder ook voor graafalgoritmen.
Een simpel voorbeeld van een certifying algoritme kan worden geïllustreerd met het algoritme van Euclides, bedoeld om de grootste gemene deler (*ggd*) van 2 natuurlijke getallen $a, b \in \mathbb{N}$ te berekenen. Een certifying versie zal tijdens het recursief proces niet enkel $\text{ggd}(a, b)$ berekenen, maar ook een 2-tupel, $(x, y) \in \mathbb{Z}^2$, als certificaat genereren met de bedoeling dat $\text{ggd}(a, b) = ax + by$. Een checker dient deze vergelijking dan simpelweg uit te rekenen en te bevestigen dat de uitvoer een geldige deler van a en b is. In het meegeleverde bestand `gcd-certified.rkt` vind je een demonstratie van dit algoritme.



Figuur 1: Diagrammen die het verschil tussen traditionele algoritmiek en certifying algoritmiek illustreren. Overgenomen uit "Certifying algorithms", door McConnell e.a., 2011, *Computer Science Review*, *5*(2), p. 120. Copyright 2010 Elsevier

2 Opdracht

Het doel van deze taak is het **implementeren en demonstreren van een certifying versie van het algoritme van Tarjan** in R^{7}RS. Hiervoor dien je je te baseren op de paper van Borovšak en Mihelič (Borovšak & Mihelič, 2016), die in bijlage werd meegeleverd met deze opgave. In die paper worden de richtlijnen voor het construeren van **zowel een certificaat als checker** voor dit algoritme besproken, alsook de correctheid van de checker bewezen. In grote lijnen bestaat het certificaat uit de geïnduceerde kerngraaf, alsook 2 bomen (of bossen) die elk een deelgraaf van de oorspronkelijk invoergraaf zijn. De checker dient per sterk-samenhangend component 4 zaken te controleren.
Je implementeert jouw certifying algoritme op basis van de instructies in dat artikel in een nieuw bestand, `tarjan-certified.rkt`, in de gelijknamige procedure, `tarjan-certified`. Vertrekkende van een kopie van het basisalgoritme in de cursuscode, nl. de procedure `scc-tarjan` uit `a-d/graph-algorithms/directed/connectivity.rkt`, breid je het algoritme van Tarjan uit zodat het algoritme naast de oorspronkelijke uitvoer, het certificaat oplevert. Vervolgens implementeer je de checker in een procedure `check-tarjan` die toelaat om de geldigheid van je algoritme te controleren. Deze procedure verwacht 3 invoerargumenten: de input van je algoritme, de resulterende output en het certificaat. Voorzie op het einde van het bestand **een demonstratie** van je checker, analoog aan de voorbeeldcode in `gcd-certified.rkt`.
Naast de paper van Borovšak en Mihelič, voorzien wij in de bijlage van deze opgave een overzichtspaper van McConnell e.a. (McConnell e.a., 2011). Deze paper laat jou toe beter bekend te geraken met het domein van certifying algoritmes. Het is noodzakelijk om secties 1, 2 (behalve subsectie 2.6) en 6 van het overzichtsartikel door te nemen. Voor een theoretisch kader kan de geïnteresseerde lezer sectie 5 van het artikel raadplegen. Merk op dat eventuele stellingen en bewijzen in die papers geen leerstof vormen!

3 Beoordeling

Je code wordt niet alleen beoordeeld op functionaliteit en performantie, maar ook op het gebruik van goede abstracties. Het voldoende documenteren van je code door middel van commentaren en een beknopt document is verplicht en maakt integraal deel uit van de evaluatie. Als onderdeel van de evaluatie van je taak dien je jouw inzending te verdedigen op het mondeling examen van dit vak. De assistent zal je ondervragen over de werking van je ingediende oplossing, alsook de motivatie achter implementatiekeuzes en kan verdere inzichtsvragen stellen over deze taak in kader van de geziene leerstof.

4 Oplevering

Upload je oplossing naar Canvas ten laatste op de aangegeven deadline die voor jou van toepassing is in de daarvoor voorziene ruimte in de 'Opdrachten' rubriek:

- **Deadline voor de Dagstudenten**: **zondag 28 april 2024 om 23u59**.
- **Deadline voor de Werkstudenten**: **maandag 27 mei 2024 om 08u00**.

De inzending dient te voldoen aan de volgende criteria. Jouw oplossing zit in een zip-bestand met je naam en studentennummer, bijv. `Youri-Coppens-123456.zip`. Dit bestand bevat één enkele map met **identieke naam**, bijv. `Youri-Coppens-123456/` met als inhoud:

1. Een `README.pdf` bestand waarin je een **beknopte** beschrijving geeft van je oplossing. Je beschrijft en motiveert de keuzes die je gemaakt hebt om tot je oplossing te komen. Je licht ook kort toe welke bestanden van de `a-d` folder je gebruikt en/of gewijzigd hebt. Eventuele gebruikte oplossingen uit het WPO moeten ook in dit document vermeld worden. Beperk dit document tot max. 2 pagina's A4, lettergrootte 10 (of groter).

2. Het bestand `tarjan-certified.rkt` waarin je jouw taak implementeert en demonstreert.

3. De map `a-d`: deze bevat alle code die nodig is om het hoofdbestand (`tarjan-certified.rkt`) uit te voeren, zijnde cursuscode inclusief gewijzigde bestanden. ADTs of modules die niet nodig zijn voor deze taak mag je weglaten uit deze folder om de grootte van je inzending in te perken. Zorg er wel voor dat alle nodige ADTs om je code te runnen wél aanwezig zijn! Dit kan je best apart testen alvorens je code op te sturen. **Het moet namelijk mogelijk zijn om je code te runnen indien enkel en alleen deze `a-d` folder geïnstalleerd is via de DrRacket Package Manager**.

5 Opmerkingen

- De deadline is strikt na te leven: taken die te laat worden ingediend, zullen niet verbeterd worden en als afwezig worden gequoteerd.
- Indien je vragen hebt over de taak, neem je tijdig contact op met de assistent.

5.1 Wat is toegelaten?

De taak dient strikt individueel en op eigen kracht gemaakt te worden. Dat betekent dat je jouw taak op een zelfstandige basis moet maken en je jouw werk moet kunnen uitleggen, onder toezicht moet kunnen herimplementeren, en je werkwijze moet kunnen verdedigen. Werk (code, tekst, etc.) overnemen van of delen met derden (bv. medestudenten, websites, GitHub, etc.) is verboden. Merk op: het is dus ook expliciet verboden om werk gegenereerd door AI tools (bv. gebaseerd op "large language models") in te dienen als eigen werk. Elektronische hulpmiddelen worden gebruikt om alle inzendingen met online bronnen en met elkaar te vergelijken, zelfs over verschillende academiejaren heen.

- De enige code die mag overgenomen worden, is de code gegeven in deze cursus: zowel de gegeven cursuscode op Canvas als de oplossingen van de WPOs van dit vak. Verder mag je enkel gebruikmaken van de bibliotheken die deel uitmaken van de R^{7}RS-standaard, met uitzondering van de Racket-procedure `format`.
- **Indien je oplossingen van het WPO hergebruikt of aanpast, vermeld je het respectievelijke WPO als bron in de README.**

5.2 Wat is plagiaat?

Elke handeling van een student die afwijkt van de gegeven instructies en niet in overeenstemming is met het examenreglement wordt beschouwd als onregelmatigheid. Plagiaat is eveneens een onregelmatigheid. Onder plagiaat wordt begrepen het gebruik maken van andermans werk, al dan niet in bewerkte vorm, zonder nauwkeurige bronvermelding (cf. OER[1], Artikel 118§2). Plagiaat kan betrekking hebben op verschillende vormen van werk zoals tekst, code, beeld, etc.

5.3 Wat gebeurt er als studenten verdacht worden?

Elk vermoeden van plagiaat zal onverwijld aan de decaan van de faculteit worden gerapporteerd. Zowel gebruiker als verlener van zulke code zullen worden gerapporteerd en zullen worden behandeld volgens de plagiaatregels van het examenreglement (cf. OER, Artikel 118). De decaan kan beslissen tot (een combinatie van) examentuchtsancties, gaande van 0 punten op het werkstuk tot een verbod tot (her)inschrijving voor één of meerdere academiejaren (cf. OER, Artikel 118§5).
Contacteer ons als je twijfelt of iets al dan niet als plagiaat zou beschouwd worden.

[1] Onderwijs- en Examenreglement

Referenties

Borovšak, T., & Mihelič, J. (2016). Certifying algorithm for strongly connected components. *Proceedings of the 25th International Electrotechnical and Computer Science Conference*, 7–10.

McConnell, R. M., Mehlhorn, K., Näher, S., & Schweitzer, P. (2011). Certifying algorithms. *Computer Science Review*, *5*(2), 119–161.