Universiteit Twente

Module 7 - 201400433

Discrete Structuren & Effeciënte Algoritmes

Het Graaf Isomorfisme Probleem

En hoe dit effeciënt gemaakt kan worden met Preprocessing

Projectgroep 1

Auteurs:
Joeri Kock
Remco Brunsveld
Frank Bruggink
Daniël Schut

Projectbegeleider:
Prof. Dr. J.C. van de Pol

March 27, 2015

Inhoudsopgave

1 Het Graaf Isomorfisme (GI) probleem

Het Graaf Isomorfisme probleem is het probleem dat gaat over het vaststellen of twee grafen isomorf zijn, dat wil zeggen structureel gelijk aan elkaar. Als twee wiskundige objecten isomorf zijn, dan is elke eigenschap, waarvan de structuur bewaard blijft door een isomorfisme en die geldt voor een van de twee wiskundige objecten, ook geldt voor het andere wiskundige object.

Hoewel het op het eerste gezicht niet zo lijkt te zijn, geldt er een isomorfisme tussen de twee grafen hiernaast. Voor elke node van de eerste graaf geldt dat de buren daarvan exact hetzelfde zijn als de buren van diezelfde node in de tweede graaf.

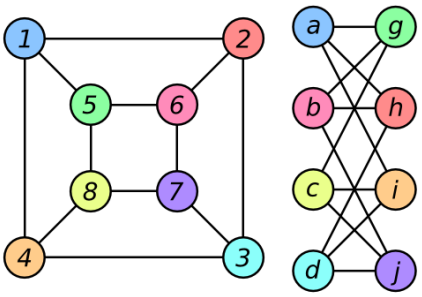

Figure 1: Twee isomorfe grafen

Een methode om isomorfisme vast te kunnen stellen tussen deze twee grafen is color refinement. Dit algoritme kent kleuren toe aan de verschillende nodes, die een indicatie geeft welke buren deze node heeft. De groene node bijvoorbeeld heeft in de eerste graaf drie buren: de lichtblauwe, de roze en de gele. Dit geldt ook voor de tweede graaf. Als dit voor elke node in de twee grafen geldt, zijn ze isomorf. Dit kan als volgt gecontroleerd worden: wanneer de lijst met alle kleuren (zonder duplicaten) van de eerste graaf gelijk is aan die van de tweede graaf, kan er gezegd worden dat ze isomorf zijn ten opzichte van elkaar. In de afbeelding hierboven is dat ook het geval. Dit is een goede manier om isomorfisme te controleren, omdat het op het eerste gezicht bij deze afbeelding lijkt alsof de twee grafen niet isomorf zijn. Ook is deze methode nuttig bij grotere grafen met veel nodes.

1.1 Relevantie van het GI probleem

Nu duidelijk gemaakt is wat het GI probleem inhoudt en hoe het gecontroleerd kan worden, blijft de vraag hoe relevant dit probleem is voor de Discrete Wiskunde. Waarom willen we weten of twee grafen (of zelfs wiskundige problemen in het algemeen) isomorf zijn ten opzichte van elkaar?

Isomorfisme heeft al voor het oplossen van vele wiskundige problemen gezorgd. Dat kan als volgt gedaan worden: een nog op te lossen wiskundig probleem X kan herleid worden tot een eenvoudiger probleem Y, wat makkelijker te begrijpen is. Neem een reeds opgelost probleem Z. Als aangetoond kan worden dat Y en Z isomorf zijn ten opzichte van elkaar, kan de conclusie getrokken worden

dat op dat moment probleem X ook opgelost is (omdat probleem X, Y en Z structureel hetzelfde zijn). In de Discrete Wiskunde zijn er al veel problemen met deze methode opgelost, wat het controleren op isomorfisme erg relevant maakt. Naast de wiskunde is het ook nuttig voor de informatica, omdat daar veel structuren als grafen weergegeven worden (denk aan netwerksystemen).

1.2 Speciale gevallen

Er zijn een aantal speciale gevallen waarvan bewezen is dat het in polynomiale tijd opgelost kan worden. Het eerste geval is een graaf die een boomstructuur heeft, dat wil zeggen: elk punt is met een ander punt verbonden, zonder cycli. De tweede zijn de planaire grafen, dat zijn de grafen die zo in het platte vlak getekend kunnen worden dat geen van de randen een andere rand kruist. Ook als je een restrictie legt op het aantal keren dat de randen elkaar minimaal kruisen, of als je een restrictie legt op de maximale graad van alle hoekpunten is het in polynomiale tijd op te lossen.

2 Ons algoritme

–Nog in te vullen–

3 Preprocessing

Om het algoritme een handje te helpen bij het effecint oplossen van het GI probleem, kunnen er voor het color refinement-algoritme al stappen ondernomen worden, zodat het algoritme zelf minder tijd nodig heeft en dus het algoritme sneller en effecinter wordt. Voor het algoritme kunnen er namelijk grafen gelimineerd worden uit de lijst die niet in aanmerking komen om isomorf te zijn met een andere graaf uit de lijst. Als je van tevoren deze grafen uit de lijst haalt, hoeft het color refinement-algoritme minder grafen te behandelen, waardoor het effecinter wordt.

Voor het testen gebruik je een lijst van grafen, bijvoorbeeld een lijst met 4 grafen. Als er één graaf is in de lijst die een van de eigenschappen van het preprocessing niet heeft en de andere drie wel, kan deze graaf uit de lijst gehaald worden. Het is dan namelijk niet meer mogelijk dat deze graaf isomorf is met één van de andere grafen. Maar als twee grafen de eigenschappen wel hebben en twee niet, zetten we deze paren van grafen in twee aparte lijsten, waar vervolgens het color refinement-algoritme op uitgevoerd kan worden.

3.1 Methoden

We controleren de grafen van tevoren op de volgende eigenschappen:

3.1.1 Grootte

De grootte van de graaf, d.w.z. het aantal vertices en het aantal edges. Als er in de lijst n graaf is met een ander aantal vertices of edges dan de andere grafen, kan deze graaf uit de lijst gehaald worden. Hij kan dan niet meer isomorf zijn met een andere graaf. De pseudo-code voor het testen op grootte staat hieronder:

```
Data: Een lijst van grafen
Result: Een lijst van grafen die mogelijk isomorf zijn
resultaat = lege lijst
for Elke graaf i in de lijst do
    for Elke andere graaf j in de lijst do
        if Het aantal vertices van graaf i en j zijn gelijk then
            if Het aantal edges van graaf i en j zijn gelijk then
                Voeg i en j toe aan het resultaat
            end
        end
    end
end
return resultaat
```

3.1.2 Connectiviteit

Als een graaf als enige in de lijst niet connected is (en de andere grafen dus wel), kan deze ook geëlimineerd worden. Met een combinatie van deze eerste twee eigenschappen wordt ook gelijk gecontroleerd op het feit of een graaf een boom is of niet. Als twee connected grafen een gelijk aantal vertices en edges hebben, zijn ze namelijk beide een boom of beide niet. Hier hoeft dus niet meer op gecontroleerd te worden. De pseudo-code voor het testen op connectiviteit staat hieronder:

```
Data: Een lijst van grafen
Result: Een lijst van grafen die mogelijk isomorf zijn
resultaat = lege lijst
for Elke graaf i in de lijst do
    for Elke andere graaf j in de lijst do
        if Grafen i en j zijn beide connected then
            Voeg i en j toe aan het resultaat
        end
    end
end
return resultaat
```

3.1.3 False twins

Na afloop van het (voor de eerste keer) uitvoeren van het color refinement-algoritme, kan er nog een stap ondernomen worden voordat we verder gaan met het individual refinement. Er kan namelijk gekeken worden of twee grafen false twins hebben. Dit zijn twee nodes die in de graaf exact dezelfde buren hebben. Als er in de lijst een graaf is die als enige false twins heeft, kan deze niet meer isomorf zijn met één van de andere grafen.

3.2 Validatie preprocessing methoden

–Nog in te vullen–

4 Testen

Natuurlijk gaan we ons color refinement-algoritme en onze preprocessing-algoritmes ook nog testen. Er worden verschillende testinstanties (lijsten van grafen) aangeleverd die we kunnen gebruiken, en we kunnen ook onze eigen grafen ontwerpen om onze code op te testen. We testen ons project op twee delen:

- De werking van het color refinement-algoritme en het algoritme voor het controleren op automorfisme. Dit zijn verplichte onderdelen van het project, dus we zullen met verschillende grafen gaan testen of deze algoritmes ook naar behoren werken.
- Het preprocessing. We testen hierbij eerst of de verschillende manieren van preprocessing werken, en vervolgens gaan we een experiment doen dat (hopelijk) aan gaat tonen dat preprocessing ook het gehele proces sneller en effeciënter maakt.

5 Conclusie

–Volgt nog–