Universiteit van Groningen

Gevorderde Algoritmen en Datastructuren

door

Jos van der Til & Rene Zuidhof

21 januari 2011

Inhoudsopgave

Hoofdstuk 1

Introductie

Dit verslag maakt deel uit van de cursus Gevorderde Algoritmen en Datastructuren van de Rijksuniversiteit Groningen. In dit verslag zal de tweede practicum opdracht behandeld worden. Deze opdracht omvat het vinden van een maximum flow in een flow network en de verschillende manieren, om dit te doen, te analyseren.

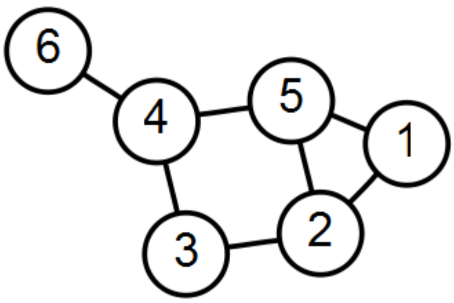
In hoofdstuk 2 zal het probleem van het vinden van een maximum flow en het gebruikte algoritme beschreven worden. De hoofdstukken 3, 4 & 5 beschrijven de algoritmen die gebruikt worden om een pad te vinden door het netwerk. De conclusies van dit onderzoek staan in hoofdstuk 6.

Hoofdstuk 2

Maximum flow probleem

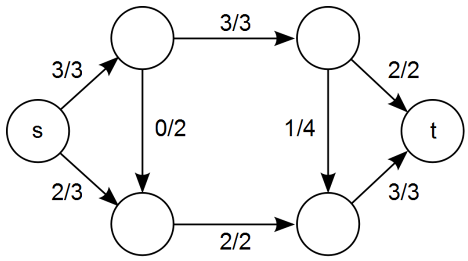
Om het probleem van het vinden van een maximum flow te kunnen begrijpen, volgt hier een korte introductie in de grafentheorie.

Een graaf is een verzameling punten (knopen) die verbonden zijn door lijnen (kanten). De kanten van een graaf kunnen een richting en/of een gewicht hebben. Een voorbeeld van een simpele graaf is te zien in figuur 2.1.



Figuur 2.1: Een ongerichte en ongewogen graaf met 6 nodes

In figuur 2.2 is een voorbeeld van een flow network te zien. De flow in deze afbeelding is maximaal, immers de capaciteit van de beide kanten die leiden naar t is volledig benut. Tevens is het duidelijk dat dit een gerichte (pijlen in plaats van lijnen als kanten) en gewogen (getallen bij de kanten) graaf is.



Figuur 2.2: Voorbeeld van een flow netwerk met een maximum flow van s naar t. De getallen zijn flow / max capaciteit.

Het probleem is nu om een flow te vinden van s naar t die maximaal is. Om dit op te lossen is er het Ford-Fulkerson algoritme, vernoemd naar L.R. Ford en D.R. Fulkerson die dit algoritme publiceerden in 1956. Deze wordt nader toegelicht in paragraaf 2.1.

2.1 Ford-Fulkerson algoritme

Het algoritme van Ford & Fulkerson werkt eigenlijk volgens een heel simpel principe. Zolang er een pad is van s naar t met beschikbare capaciteit, dan wordt de flow daar langs gestuurd. Dit wordt herhaalt totdat er geen pad meer mogelijk is. Een pad van s naar t met beschikbare capaciteit wordt een 'augmenting path' genoemd.

De eisen die gesteld worden aan een geldige flow zijn:

- De flow mag nooit groter zijn de de capacity van een kant. $0 \leq flow(u,v) \leq capacity(u,v)$
- De netto flow van een node is gelijk aan 0. Dit geldt niet voor s of t.

  $$\sum_{e \in E^-} \sum_{v \in E^+} flow(e) - flow(v) = 0$$

  Waar E^- de verzameling van uitgaande kanten is en E^+ de verzameling uitgaande kanten van knoop E is.

Omdat het Ford-Fulkerson algoritme niet aangeeft op welke manier er een 'augmenting path' gevonden dient te worden, zijn er meerdere methodes beschikbaar. De methodes die onderzocht zullen worden in dit document zijn:

1. Depth-first search;
2. Breadth-first search;
3. Priority-first search.

2.1.1 Pseudocode

De pseudocode van het algoritme is te vinden in algoritme 1.

Algorithm 1 Ford-Fulkerson Algorithm

```
Require: Input: Flow network N containing graph G
  for all edge e ∈ N do
    flow(e) ← 0
  end for
  stop ← false
  repeat
    traverse G starting at s to find an augmenting path to t (π)
    if an augmenting path π exists then
      Δ ← +∞
      for all edge e ∈ π do
        if  residual capacity(e) ≤ Δ then
          Δ ← residual capacity(e)
        end if
      end for
      for all edge e ∈ π do
        if e is a forward edge then
          flow(e) ← flow(e) +Δ
        else
          flow(e) ← flow(e) −Δ
        end if
      end for
    else
      stop ← true
    end if
  until stop
```

Hoofdstuk 3

Depth-first search

Hoofdstuk 4

Breadth-first search

De tweede manier om een augmenting path te vinden in een graaf is de breadth-first search. Deze methode, die ook gebruikt wordt in het Edmonds-Karp algoritme, vind het kortste pad van s naar t. Het korste pad is in dit geval gedefineert als het pad met het laagste aantal kanten.

4.1 Pseudocode

Hoofdstuk 5

Priority First Search

Hoofdstuk 6

Conclusie

Lijst van figuren

Lijst van tabellen

Bijlage A

Source Code

Source code HIER