Hoofdstuk 6

Toepassingen van Diepte Eerst Zoeken

Zoals in Grafen I is ook hier *n* het aantal knopen en *m* het aantal verbindingen.

6.1 Enkelvoudige Samenhang van Grafen

6.1.1 Samenhangende componenten van een *ongerichte graaf*

Een ongerichte graaf is **samenhangend als er een weg bestaat tussen elk paar knopen**.

Een niet samenhangende ongerichte graaf bestaat eigenlijk uit een hoop samenhangende deelgrafen. Aangezien sommige algoritmen verwachten dat een graaf samenhangend is, is vaak vereist die componenten te bepalen. Zo kunnen ze apart behandeld worden.

Om te bepalen of een graaf al dan niet samenhangend is **kan met DEZ**. Als de DEZ-boom alle knopen van de graaf bevat, dan is de ongerichte graaf samenhangend. Als er meerdere bomen nodig zijn, dan bevat elke boom de knopen van een samenhangende component. Dit kan eigenlijk ook met BEZ.

Testen of een ongerichte graaf samenhangend is / de samenhangende componenten bepalen is dus $\Theta(n+m)$ voor een ijle graaf en $\Theta(n^2)$ voor een dichte graaf.

TL;DR - Met DEZ kan je nagaan of een ongerichte graaf samenhangend is, en zelfs de samenhangende componenten bepalen.

6.1.2 Sterk samenhangende componenten van een *gerichte graaf*

Een gerichte graaf met een weg tussen elk paar knopen in **beide richtingen** is **sterk samenhangend**. Een gerichte graaf met een weg tussen elk paar knopen is zwak samenhangend.

Een niet samenhangende gerichte graaf bestaat uit zo groot mogelijke sterk samenhangende componenten. Ook hier gaan sommige algoritmen ervan uit dat de graaf sterk samenhangend is. We moeten dus eerst die componenten identificeren.

Daarvoor kan een **componentengraaf** opgesteld worden. Die heeft een knoop voor elke sterk samenhangende component met een verbinding naar een andere knoop als er in de originele graaf een verbinding bestaat tussen die twee componenten. De componenten kunnen dan behandeld worden door deze componentengraaf te volgen.

Ook hier kan **DEZ gebruikt worden voor het vinden van de componenten**. Er bestaat een algoritme (Kosaraju, Sharir) dat twee keer DEZ gebruikt, en dus trager is, maar gemakkelijker uit te leggen.

1. **Omgekeerde graaf opstellen.** De richtingen worden omgedraaid.

2. **DEZ post-order nummering.** Diepte eerst zoeken door deze omgekeerde graaf, waarbij de knopen post-order genummerd worden. Hoe groter het nummer, hoe later de knoop werd afgewerkt.

3. **DEZ originele graaf met als startknoop de knoop met hoogste nummer**. Het resultaat is een diepte-eerst bos, waarvan de bomen de gezochte sterk samenhangende componenten zijn.

We kunnen besluiten dat knopen uit verschillende bomen niet tot dezelfde sterk samenhangende component kunnen behoren. Verder tonen we aan dat al de knopen in een boom ook in beide richtingen met elkaar zijn verbonden.

- In de boom is er zeker een weg van de wortel *w* naar elk van de knopen *u* in de boom, dus er is een weg van *u* naar *w* in de omgekeerde graaf.
- *w* is steeds een voorouder van *u* in een diepte-eerst boom van de omgekeerde graaf. *W* heeft immers altijd een hoger post-order nummer dan *u*. Daaruit volgt dat er een weg bestaat van *w* naar *u* in de omgekeerde graaf. Dit betekent dat er een weg is van *u* naar *w* in de originele graaf.

Deze methode is $\Theta(n+m)$ bij ijle en $\Theta(n^2)$ bij dichte grafen. De performantie van het omkeren van de graaf wijzigt dit asymptotisch gedrag niet.

TL;DR - Maak de omgekeerde graaf, doe daar DEZ post-order nummering op, en doe dan DEZ op de originele graaf met telkens de knoop met het hoogste nummer als startknoop. Zo vind je de sterk samenhangde componenten.

6.2 Dubbele Samenhang van Ongerichte Grafen

Vergelijk een ongerichte graaf met een communicatienetwerk. Het netwerk moet zodanig geconfigureerd worden dat er meerdere verbindingen of knopen moeten uitvallen vooraleer een deel van het netwerk onbereikbaar wordt.

Twee definities van dubbele samenhang:

- Een verbinding die een ongerichte graaf doet uiteenvallen als ze wegvalt noemt men een **bridge**. Een graaf zonder bruggen noemt men **dubbel lijn-samenhangend** (2-edge-connected). Logischerwijs krijg je bij het verwijderen van alle bruggen een hoop componenten die elk dubbel lijn-samenhangend zijn.
  Eigenschap: Tussen elk paar knopen van een dubbel lijn-samenhangende graaf bestaan minstens twee wegen zonder gemeenschappelijke verbindingen.
- Een knoop die de graaf doet uiteenvallen in tenminste twee deelgrafen als hij wegvalt is een **scharnierpunt** (articulation point). Een graaf zonder scharnierpunten is **dubbel samenhangend** (biconnected). Een graaf met scharnierpunten kan onderverdeeld worden in bruggen en dubbel samenhangende componenten. Hierbij kunnen twee componenten wel gemeenschappelijke knopen hebben, maar nooit gemeenschappelijke verbindingen. Een verbinding is immers ofwel een brug, ofwel deel van één component.

Eigenschap: Tussen elk paar knopen van een dubbel samenhangende graaf bestaan minstens twee wegen zonder gemeenschappelijke knopen.

De tweede vorm van samenhang, "dubbele samenhang", is sterker dan de eerste, "dubbele lijn-samenhang". Bij de eerste vorm moet immers een knoop met al zijn verbindingen wegvallen, bij de tweede is een verbinding al fataal.

Vinden van scharnierpunten, bruggen, en de samenhangende componenten kan met DEZ.

1. **DEZ boom opstellen**, **pre-order nummering**.
2. **Voor elke knoop *u* bepaal je de laagst genummerde knoop die bereikbaar is vanuit *u* via een weg bestaande uit nul of meer dalende boomtakken gevolgd door één terugverbinding**. Dit kan via een tweede post-order doorgang van de boom.
3. **Een knoop is <u>geen</u> scharnierpunt als alle kinderen ervan een knoop kunnen bereiken die hoger ligt in de boom dan hemzelf.**
   De wortel is een scharnierpunt als hij meer dan één kind in de boom heeft. Want de enige verbinding tussen deze kinderen loopt via de wortel.

Pseudocode scharnierpunten vinden (zonder aparte test voor wortel):

```
void rec_zoek_scharnierpunten(int i){        // vanuit knoop i
   ontdekt[i] = true;
   pre[i] = num++;                  // preorder nummering
   laagst[i] = pre[i];              // elke knoop bereikt minstens zichzelf

   for(alle buren j van i){
      if(!ontdekt[j]){
         ouder[j] = i;
         rec_zoek_scharnierpunten(j);

         if(laagst[j] < laagst[i]) laagst[i] = laagst[j];      // nieuw minimum
         else if(laagst[j] >= pre[i]) MELD I IS EEN SCHARNIERPUNT VOOR J;
      }
      else {
         if(j != ouder[i]){
            if(pre[j] < laagst[i]){
               laagst[i] = pre[j];           // want i->j is een terugverbinding
            }
         }
      }
   }
}

void zoek_scharnierpunten(){
   for(int i=0; i<n; i++){
      ontdekt[i] = false;
      ouder[i] = -1;
   }
   int num = 0;
   rec_zoek_scharnierpunten(0);
}
```

Merk op dat een knoop meermaals scharnierpunt kan zijn. Als knoop *a* een scharnierpunt is voor zijn kind *b*, dan behoren de meest recent ontdekte verbindingen tot ene nieuwe dubbel samenhangende component. Want er is geen enkele weg die vanuit b gevonden werd die naar een knoop hoger in de boom dan *a* leidt.

Het volstaat dus om al de ontdekte verbindingen op een stack bij te houden en ze eraf te halen bij het ontdekken van een scharnierpunt *a*, tot en met de boomtak (a, b).

Deze methode is $\Theta(n+m)$ bij ijle en $\Theta(n^2)$ bij dichte grafen.

TL;DR - Dubbel lijn-samenhangend bevat geen bruggen. Dubbel samenhangend bevat geen scharnierpunten. Dubbel samenhangend is sterker dan dubbel lijn-samenhangend. De scharnierpunten en bruggen vinden kan met pre-ordering van een DEZ-boom.

6.3 Eulercircuits

Een **Eulercircuit** is **een gesloten omloop in een graaf die alle verbindingen één keer bevat**. Niet elke graaf heeft een Eulercircuit. Een Eulercircuit kan niet eenvoudig gevonden worden door middel van DEZ.

6.3.1 Ongerichte Grafen

Een **Eulergraaf** is een graaf met een Eulercircuit.

Equivalente eigenschappen :

1. **Een samenhangende graaf G is een Eulergraaf.**
2. **De graad van elke knoop van G is even.**
3. **De verbindingen van G kunnen onderverdeeld worden in lussen.**

Eigenschap 2 volgt uit 1. Telkens een knoop voorkomt op een Eulercircuit verhoogt dat zijn graad met 2. Elke verbinding komt één keer voor op het circuit, dus de graad is even.

Eigenschap 3 volgt uit 2. Er zijn minstens n-1 verbindingen, want G is samenhangend. Geen enkele knoopgraad is één (zie eig. 2), dus er zijn minstens n verbindingen. G bevat dus minstens één lus. Verwijderen we die lus bekomen we samenhangende componenten met knopen met even graden. We kunnen dit blijven doen tot alles verdeeld is in lussen.

Eigenschap 1 volgt uit 3. Stel dat L een lus is in G. Als L een Eulercircuit is, dan is G een Eulergraaf. Anders bestaat er een andere lus L' die een gemeenschappelijke knoop heeft met L. Lus L' kan tussengevoegd worden in L bij die knoop, wat een grotere lus oplevert. Herhalen we dit, bekomen we een Eulercircuit.

Algoritme van Hierholdzer. Om de eerste lus L te vinden beginnen we bij een willekeurige knoop en volgen verbindingen tot we weer in die knoop komen. De volgende lus L' begint bij een van de knopen van L waarvan nog niet alle verbindingen doorlopen zijn. Ook hier volgen we verbindingen tot we terug in de startknoop komen.

6.3.2 Gerichte Grafen

Een Eulercircuit in een gerichte graaf is enkel mogelijk als de graaf een Eulergraaf is (sterk samenhangend dus). De voorwaarden voor een gerichte Eulergraaf zijn analoog aan die van een ongerichte: de ingraad van elke knoop moet gelijk zijn aan zijn uitgraad. Ook de constructie van het Eulercircuit verloopt analoog.

TL;DR - Een Eulercircuit is een gesloten omloop in een graaf die alle verbindingen bevat. Een samenhangende graaf is een Eulergraaf, de graad van elke knoop is even, de verbindingen kunnen opgedeeld worden in lussen.