De tweede programmeeropgave — Roddelen
Algoritmiek voorjaar 2008, Universiteit Leiden

Het roddelprobleem. We hebben n personen, die elk precies één roddel kennen, waarbij alle n roddels verschillend zijn. Het is de bedoeling dat iedereen alle n roddels te weten komt. Hiertoe kunnen ze elkaar opbellen. Bij ieder telefoongesprek zijn precies twee personen betrokken, en die wisselen dan alle roddels uit die ze op dat moment kennen. Na afloop van het gesprek kennen beiden dus hun eigen roddels en die van de ander. Vraag: hoeveel gesprekken zijn er minstens nodig om iedereen op de hoogte te brengen van alle n roddels?

De analyse. Het is niet zo moeilijk te bewijzen dat $2n-3$ gesprekken voldoende zijn. Bewijs dit door een serie van $2n-3$ gesprekken te geven die samen inderdaad alle roddels verspreiden. Het is ook mogelijk te laten zien dat $2n-4$ gesprekken voldoende zijn. Probeer dit eveneens te bewijzen. Hint: laat zien dat het geval $n=4$ met vier gesprekken kan, en combineer dat met het bewijs van bovenstaande bewering.
Overigens is bewezen dat het voor $n \geq 4$ nooit met minder dan $2n-4$ telefoontjes kan. Conclusie: het minimale aantal gesprekken is voor $n \geq 4$ altijd precies $2n-4$.

Varianten op het roddelprobleem. Het hierboven beschreven probleem verscheen in 1999 in de Nationale Wetenschapsquiz voor $n=6$, maar het probleem is dertig jaar geleden al opgelost. Voor geïnteresseerden: kijk eens op internet bij het Nieuw Archief voor Wiskunde, http://www.nieuwarchief.nl/serie5/deel01/jun2000/pdf/craats.pdf.
Omdat een backtracking algoritme voor het vinden van een kortste gesprekkenreeks al voor kleine n erg veel tijd nodig heeft, bekijken we hier de volgende twee varianten op het probleem.

1. In de beginsituatie kent iedereen behalve zijn eigen roddel ook nog enkele andere roddels van de in totaal n stuks. Wat is dan de kortste serie gesprekken, nodig om iedereen op de hoogte te brengen van alle roddels? Dit is een generalisatie van het oorspronkelijke probleem.
2. Sommige personen hebben zo'n hekel aan elkaar dat ze elkaar beslist niet op willen bellen. Wat is nu een serie gesprekken (niet noodzakelijk de kortste) die de roddels verspreidt?

Het programma. Het te schrijven C++-programma moet gebruik maken van *backtracking met recursie*. Globaal idee: genereer steeds één voor één alle mogelijke gesprekken en als dat een nuttig gesprek is (zie onder), laat dat dan plaatsvinden (wissel dus ook de roddels uit) en doe weer hetzelfde. Houd steeds de tot dusver bekeken reeks gesprekken bij, en voor variant 1. ook (de lengte van) de tot dusver gevonden kortste gesprekkenreeks.

Het programma moet verder het volgende doen. Het aantal n moet worden ingelezen. Vervolgens moet de gebruiker kunnen kiezen uit variant 1. of 2. Bij variant 1. moet uit een file worden ingelezen wie welke roddels kent. Vervolgens moet met behulp van backtracking een minimale serie telefoongesprekken worden gegenereerd die alle roddels verspreidt. In verband met bovenstaande analyse is het minimale aantal benodigde gesprekken altijd $\leq 2n-4$. Bij variant 2. moet uit een file worden ingelezen welke tweetallen personen niet met elkaar mogen bellen. Vervolgens moet via backtracking *een* serie telefoongesprekken worden gegenereerd die de roddels verspreidt onder alle personen. In deze variant kent iedereen in het begin alleen zijn

eigen roddel. Bij deze situatie hoeft er niet altijd een oplossing te bestaan.

Opmerkingen

- Let erop dat je geen eeuwige recursie krijgt. Dit kun je bijvoorbeeld voorkomen door te zorgen dat een gesprek altijd nieuwe informatie oplevert voor (ten minste) een van beide gesprekspartners.
- Gebruik bij variant 1. ook de lengte van de tot dusver gevonden kortste reeks gesprekken als conditie bij het backtracken.
- Gebruik een klasse `roddels`, waarin in elk geval de twee member-functies die corresponderen met de twee mogelijke opties staan. Het is de bedoeling dat de member-functie bij variant 2. een eenvoudige aanpassing is van die bij variant 1.
- Bij het inlezen mag je gewoon gebruik maken van `invoer>>getal`, er hoeft dus niet karakter voor karakter uit de file te worden gelezen. Iets analoogs geldt voor het inlezen van n. Kies n niet te groot.
- De (kortste) serie gesprekken moet op het beeldscherm worden getoond.
- Boven elke functie moet een commmentaarblokje komen met daarin een (zeer) korte beschrijving van wat de functie doet. Noem daarin ook de gebruikte parameters: geef hun betekenis en geef aan hoe ze eventueel veranderd worden door de functie. Geef bij memberfuncties ook aan wat deze met de membervariabelen van het object doen. Let ook op de layout (consequent inspringen) en op het overige commentaar bij de programmacode (alleen zinvol en kort commentaar).
- Doe experimenten (voor beide varianten) met verschillende waarden van n en verschillende representatieve beginsituaties. Bekijk het verschil in tijd dat het algoritme nodig heeft om de oplossing te vinden. Bekijk het effect van het soort beginsituatie, de grootte van n, etcetera. Meer informatie over mogelijke representatieve testgevallen komen op de website te staan: (`http://www.liacs.nl/home/jlaros/edu/alg.shtml`).
- Maak een (kort) getypt **verslag**, waarin de in "De analyse" gevraagde bewijzen, alsmede een kort verslag van de hierboven genoemde experimenten. Leg in het verslag bovendien duidelijk (maar niet al te lang) uit hoe de methode backtracking voor het vinden van een kortste serie gesprekken (variant 1.) werkt en hoe de aanpassing voor het zoeken naar een serie gesprekken bij variant 2. eruit ziet.
- Eventuele extra informatie over de programmeeropdracht is te vinden op:
  `http://www.liacs.nl/home/jlaros/edu/alg.shtml`
  Voor meer informatie over het vak, laatste wijzigingen, etcetera, raadplege men:
  `http://www.liacs.nl/home/graaf/ALGO/algo2008.html`

Uiterste inleverdatum: vrijdag 25 april 2008. Voor elke week te laat inleveren gaat er een punt van het cijfer af. Het programma per e-mail sturen aan de hoofdnakijker: jlaros@liacs.nl. Listing en verslag moeten op papier worden ingeleverd en in de daartoe bestemde doos met opschrift Algoritmiek in de postkamer van Informatica (kamer 156) worden gedeponeerd. Vermeld overal duidelijk de namen van de makers.
Normering: verslag 2; commentaar en layout 1,5; modulaire opbouw en OOP 1,5; werking 5.