Tentamen Algoritmiek

Dinsdag 31 juli 2012, 10.00 - 13.00

Dit tentamen bevat *vier* vragen. Geef een *duidelijke* toelichting bij al je antwoorden!

Vraag 1: Wisselgeld (40 punten)

In deze opgave bekijken we het **muntenprobleem**:

Gegeven: onbeperkt veel munten van $d_1, d_2, \ldots, d_m$ eurocent, en een te betalen bedrag van n ($n > 0$) eurocent. Alle munten d_i zijn groter dan nul en onderling verschillend. In het bijzonder geldt dat $d_1 = 1$.

Gevraagd: het minimale aantal munten dat nodig is om het bedrag van n eurocent te betalen.

a) Beschrijf (in woorden of pseudo-code) een gretig algoritme voor dit probleem. Laat zien dat bij een set van 3 munten van respectievelijk 1, 2, en 5 eurocent je algoritme voor $n = 1$ tot en met $n = 5$ altijd het optimale antwoord voor het muntenprobleem geeft. Beargumenteer waarom je in deze gevallen inderdaad steeds het optimale antwoord krijgt.

b) Geef een voorbeeld van een set van 3 munten (met $d_1 = 1$) en een waarde n waarbij een gretig algoritme *niet* de optimale oplossing vindt. Laat zien wat het gretige algoritme in dit geval doet, en waarom dit niet optimaal is.

c) Een mogelijk alternatief voor een gretig algoritme in zo'n geval kan backtracking zijn. Hierbij kies je steeds een munt, totdat je het bedrag n bij elkaar hebt, of totdat je al minstens zo veel munten hebt als in de beste complete oplossing tot nu toe. **i)** Als je een munt moet kiezen, in welke volgorde zou je dan de munten proberen? **ii)** Leg uit hoe je kunt voorkomen dat je dubbel werk doet, doordat je meerdere keren (feitelijk) dezelfde oplossing zou bekijken (hint: denk aan het handelsreizigersprobleem, waar we altijd stad b voor stad c wilden bezoeken om dubbel werk te voorkomen). **iii)** Pas het backtracking algoritme toe voor een set met 4 munten van respectievelijk 1, 4, 5, en 10 eurocent, en de waarde $n = 13$. Geef de complete state-space tree en geef ook aan in welke volgorde de knopen bekeken worden.

d) Bij veel minimalisatieproblemen kun je een backtracking algoritme uitbreiden door bij deeloplossingen (net als bij branch-and-bound) ondergrenzen te berekenen voor de waarde van een complete oplossing. Doe een voorstel voor een ondergrens bij het muntenprobleem (we bedoelen niet zozeer een getal, maar een functie die bij elke deeloplossing een getal oplevert). Bij dit probleem is de waarde van een complete oplossing dus het aantal gebruikte munten.

We kijken nu naar *algemene* minimalisatieproblemen.

e) Beschrijf (in woorden) hoe een branch-and-bound algoritme voor een minimalisatieprobleem eruit ziet. Geef aan hoe deeloplossingen worden opgebouwd, wat er bij iedere knoop in de state-space-tree wordt berekend en gecontroleerd, en hoe best-fit-first branch-and-bound vervolgens werkt.

f) Stel dat we voor een minimalisatieprobleem ook een gretig algoritme hebben, dat niet altijd de optimale oplossing vindt. Leg uit hoe we de uitkomst van dit gretige algoritme kunnen gebruiken bij een branch-and-bound algoritme voor het probleem.

Vraag 2: Pre- en postorder (15 punten)

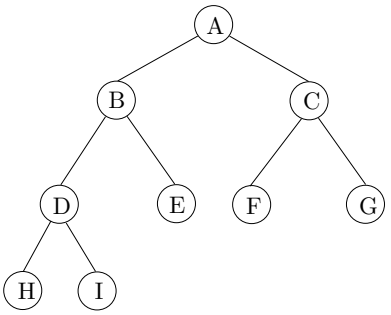
Gegeven de volgende datastructuur en twee algoritmen, waarbij `func` een nog niet nader gespecificeerde hulpfunctie is.

```
struct knoop {
   int waarde;
   knoop* ouder;
   knoop* links;
   knoop* rechts;
}; // knoop


void preorde (knoop* wortel) {
   if ( wortel != NULL ) {
      wortel->waarde = func(wortel);
      preorde (wortel->links);
      preorde (wortel->rechts);
   } // if
} // preorde


void postorde (knoop* wortel) {
   if ( wortel != NULL ) {
      postorde (wortel->links);
      postorde (wortel->rechts);
      wortel->waarde = func(wortel);
   } // if
} // postorde
```

a) Zie onderstaande binaire boom met negen knopen A-I. De bijbehorende datastructuur bestaat uit knopen, waarbij de waarden van `ouder`, `links` en `rechts` al ingevuld zijn.



Geef de volgorde waarin de functie `func` voor iedere knoop van de boom wordt aangeroepen bij **i)** de preorde functie en **ii)** de postorde functie. Je antwoord is dus voor zowel **i)** als **ii)** een rijtje $A = \ldots B = \ldots C = \ldots$ met op de puntjes getallen die de volgorde van aanroep van `func(wortel)` en daarmee het toekennen van een waarde aan `waarde` aangeven.

b) We interpreteren de variabele `waarde` van iedere knoop als de lengte van het pad van de wortel naar die knoop (met waarde(wortel) = 0). Initieel is deze waarde nog niet ingevuld in de datastructuur. Schrijf een implementatie (in C++ of in pseudo-code) van de functie `int func( knoop *wortel )` die voor een bepaalde knoop de waarde bepaalt, en beargumenteer of jouw functie door de *postorde* of *preorde*-functie aangeroepen moet worden.

c) Nu interpreteren we de variabele `waarde` anders, namelijk als een bit die waarde 1 of 0 aan kan nemen. Voor de *bladeren* van de boom geldt dat de waarde al gezet is. Voor de knopen die geen blad zijn, geldt dat de waarde berekend moet worden: de waarde voor zo'n knoop is een logische AND van de waarden van de kinderen (als een knoop maar 1 kind heeft, is de waarde gelijk aan de waarde van het kind). Schrijf

wederom een implementatie (in C++ of in pseudo-code) van de functie `int func( knoop *wortel )` die voor een bepaalde knoop de waarde bepaalt (voor bladeren is de waarde al gezet; voor die knopen kun je gewoon de reeds aanwezige waarde retourneren), en beargumenteer of jouw functie door de *postorde* of *preorde*-functie aangeroepen moet worden.

Vraag 3: Goedkoop op vakantie (25 punten)

Twee studenten, Robert en Bauke, willen met een bijna lege portemonnee toch op vakantie gaan. Ze besluiten met de bus van Amsterdam naar Barcelona te gaan (een fietsvakantie zien Robert en Bauke niet meer zitten). Ze willen zo goedkoop mogelijk reizen, al dan niet met tussenstops. Er rijdt een aantal busmaatschappijen op dit traject die allemaal een tussenstop maken in Brussel, Parijs en Toulouse. Bauke heeft al opgezocht wat de goedkoopste kaartjes zijn op alle deeltrajecten tussen Amsterdam en Barcelona. De prijzen daarvan staan in onderstaande tabel `prijs`:

	naar				
van	Amsterdam (0)	Brussel (1)	Parijs (2)	Toulouse (3)	Barcelona (4)
Amsterdam (0)	0	12	18	24	31
Brussel (1)	-	0	5	13	20
Parijs (2)	-	-	0	7	13
Toulouse (3)	-	-	-	0	8
Barcelona (4)	-	-	-	-	0

Laat kosten[i] de prijs van de goedkoopste busreis van Amsterdam naar de stad met nummer i voorstellen. We zoeken dus kosten[4].

Een recurrente betrekking voor dit probleem is als volgt:

$$\text{kosten}[i] = \begin{cases} 0 & \text{als } i = 0 \\ \min_{0 \le k < i}(\text{kosten}[k] + \text{prijs}[k][i]) & \text{als } i \ge 1 \end{cases}$$

We gaan met bottom-up Dynamisch Programmeren dit probleem oplossen, gebruik makend van deze recurrente betrekking.

a) Leg het verschil uit tussen bottom-up en top-down dynamisch programmeren (niet specifiek voor dit probleem, maar in het algemeen). Geef van beide soorten Dynamisch Programmeren een mogelijk voordeel.

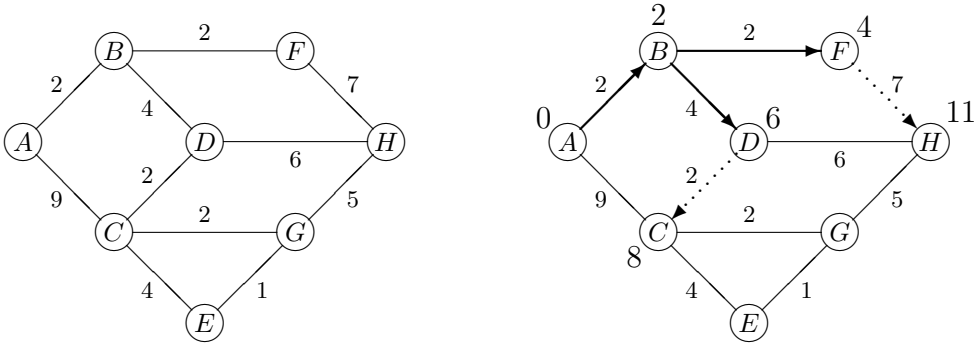
b) Geef een algoritme (in C++ of in pseudo-code) dat berekent wat de *laagste kosten* zijn om van Amsterdam in Barcelona te komen. Je algoritme moet gebruik maken van bottom-up dynamisch programmeren. Leg uit wat **i)** de deeloplossingen zijn in dit probleem en **ii)** in welke volgorde de tabel kosten[i] gevuld wordt.

c) Pas het algoritme toe op de waarden in bovenstaande tabel. Laat voor iedere waarde van i (dus voor $i = 0, 1, 2, 3, 4$) zien hoe je de waarde van kosten[i] berekent.

d) Geef aan (in woorden) hoe je algoritme aangepast kan worden om niet alleen de laagste kosten terug te geven, maar ook de plaatsen waarin overgestapt moet worden om deze kosten te kunnen bereiken. Deze plaatsen moeten niet alleen berekend worden, maar ook gepresenteerd.

Vraag 4: Dijkstra (20 punten)

In deze opgave bekijken we Dijkstra's algoritme voor het vinden van alle kortste paden vanaf een startknoop A in de graaf.



a) Zie bijgaande situatie waarbij knopen A, B, D en F reeds zijn geselecteerd. Welke knopen worden er in deze fase van het algoritme *bekeken* ? Welke knoop wordt *gekozen*? Wat is het effect van het kiezen van deze knoop op de waarden van de knopen C, D, E, G, en H (licht je antwoord toe voor ieder van deze knopen!)?

b) Kan, na het kiezen van de knoop bij **a)** en het bijwerken van de waarden van de knopen als gevolg hiervan, de waarde van C hierna (in principe) nog veranderen? En de waarde van H? Motiveer je antwoorden!

c) Maak het algoritme af, vanaf de keuze van de knoop bij onderdeel a), en maak duidelijk wat er in elke stap gebeurt, d.w.z., geef aan welke knopen bij iedere stap in het algoritme bekeken worden, welke knoop uiteindelijk gekozen wordt, en hoe de waarden van de verschillende knopen aangepast wordt.

d) Laat met een tegenvoorbeeld zien dat Dijkstra's algoritme niet werkt als negatieve padkosten toegestaan zijn. Laat zien wat het algoritme van Dijkstra in dit geval doet, en waarom dit niet optimaal is.

Einde van dit tentamen. Veel succes!