Tentamen Algoritmiek
Woensdag 9 juli 2014, 10.00 – 13.00 uur

Geef een **duidelijke toelichting** bij al je antwoorden.

Veel succes !

Opgave 1. (20 punten)
We bekijken het tweepersoonsspel `(N,M)-game`, dat lijkt op het drinkspel `21-game`. Hierin zijn N en M gehele getallen en is $2 \leq M < N$.
Twee spelers, in deze opgave Tristan en Isolde, noemen om de beurt een geheel getal volgens bepaalde spelregels. Er wordt gestart met het getal $g_0 = 1$. De speler die begint moet nu een getal noemen dat groter is dan het begingetal 1 en kleiner of gelijk aan $1 + M$. Daarna is zijn tegenstander aan de beurt. De regels voor het getal g_i dat een speler mag noemen in beurt i zijn voor algemene i als volgt:

- $g_i \leq N$
- $g_{i-1} < g_i \leq g_{i-1} + M$
  De speler zegt dus een getal hoger dan het vorige getal maar maximaal M hoger.

Degene die het getal N noemt verliest. We spreken af dat Tristan begint. Het doel van dit spel is dus om te proberen de tegenstander het getal N te laten noemen.

Hieronder een mogelijk spelverloop, met $N = 8$ en $M = 3$:

```
1,T -----> 3,I -----> 4,T -----> 7,I -----> 8,T
```

In de beginsituatie is Tristan aan de beurt en hij kiest 3. Vervolgens zegt Isolde 4, waarop Tristan 7 zegt en Isolde daarna niet anders kan dan de 8 noemen. Tristan wint hier dus.

a. (3 punten)
Wat zijn voor dit spel toestanden, acties en mogelijke eindtoestanden voor algemene N en M?

b. (12 punten)
Teken de toestand-actie-ruimte voor het geval $N = 6$ en $M = 3$.
Geef bij elke eindtoestand aan of deze winnend is voor Tristan of voor Isolde. Bepaal hieruit voor *elke* toestand in je toestand-actie-ruimte voor wie deze winnend is en zet dit in je tekening erbij. Is het spel hier winnend voor Tristan of Isolde, en hoe moet deze spelen om te winnen?
Advies: teken per niveau toestanden die meer dan één keer voorkomen slechts één keer.

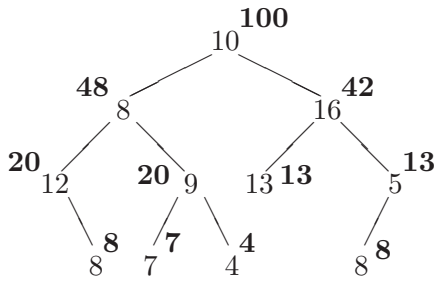
c. (5 punten)
Als N een $(M + 1)$-voud $+ 1$ is (bijvoorbeeld $(N, M) = (7, 2)$ of $(11, 4)$), is er een winnende strategie voor Tristan. Hij zegt telkens een $(M + 1)$-voud als hij aan de beurt is, en zal dan winnen. Leg uit waarom deze strategie werkt onder de gegeven spelregels. Toon daartoe aan:
(i) *Als* hij elke beurt een $(M + 1)$-voud mag zeggen dan wint hij zeker
(ii) Het is altijd mogelijk voor Tristan om een $(M + 1)$-voud te kiezen, maar –als Tristan optimaal speelt– voor Isolde niet. Laat dit zien voor Tristans eerste beurt en de daaropvolgende beurten.

-vervolg op pagina 2

Opgave 2. (17 punten)
Gegeven een binaire boom met ingang (ofwel: wortel) `wortel` die gehele getallen bevat. Hierin is `wortel` een pointer naar een `knoop`, waarbij:

```
struct knoop {
   knoop* links;
   knoop* rechts;
   int info;
   int som;
}; // knoop
```

Voorbeeld:



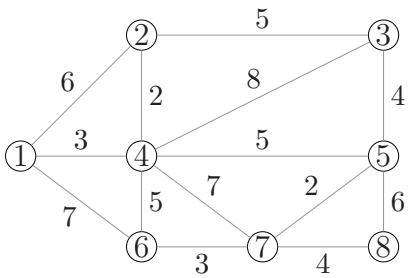
In het voorbeeld staat in de knopen de waarde van het `info`-veld. Bij elke knoop is tevens dikgedrukt de inhoud van het te berekenen `som`-veld aangegeven.

a. (10 punten)
We nemen aan dat de `som`-velden nog geen waarde hebben. Schrijf nu een *recursieve* C++-functie `optellen(knoop* wortel)`, die in elke knoop het `som`-veld vult met de som van alle `info`-waarden uit de subboom met die knoop als wortel. Zie bovenstaand voorbeeld.

We noemen een binaire boom *in balans* als *elke* knoop inhoud-gebalanceerd is:
- Een knoop met twee kinderen is inhoud-gebalanceerd als de som van de `info`-waarden uit de linkersubboom gelijk is aan de som van de `info`-waarden uit de rechtersubboom.
- Knopen met minder dan twee kinderen zijn per definitie inhoud-gebalanceerd.
De voorbeeldboom als geheel is niet in balans: de wortel en de knoop die 9 bevat zijn beide niet inhoud-gebalanceerd, de andere knopen wel.

b. (7 punten)
We nemen aan dat de `som`-velden gevuld zijn zoals in **a.** bedoeld. Schrijf nu een *recursieve* C++-functie `bool balans(knoop* wortel)`, die aangeeft of de binaire boom met ingang `wortel` in balans is.

Opgave 3. (10 punten)
Het algoritme van Dijkstra bepaalt voor gewogen grafen de (lengtes van) kortste paden vanuit een gegeven knoop naar alle andere knopen.
(i) Pas het algoritme van Dijkstra toe op onderstaande graaf, beginnend in knoop 1. Geef voor elke stap van het algoritme duidelijk aan welke knoop erbij wordt gekozen in U (= verzameling knopen waarvan de kortste afstand vanaf knoop 1 bekend is) en welke labels door die keuze veranderen en hoe. Leg ook uit hoe je uitwerking gelezen moet worden (legenda).
(ii) Geef de uiteindelijke boom van kortste paden met daarin de bijbehorende lengtes van de kortste paden vanaf knoop 1.



-vervolg op pagina 3

Opgave 4. (28 punten)
Gegeven een array $A = A[0], A[1], \ldots, A[n-1]$ dat n (≥ 2) nullen en enen bevat.
We willen weten of er ergens in het array twee nullen naast elkaar staan. We gaan dit probleem op drie manieren oplossen: met brute force, met decrease-and-conquer en met divide-and-conquer.

Voorbeeld: in de rijtjes $0, 1, 1, 0, 0, 1, 0, 1$ en $0, 0, 0, 1, 0, 0, 1, 1$ komen twee nullen direct naast elkaar voor; in het rijtje $1, 0, 1, 1, 0, 1, 0, 1$ is dit niet het geval.

We beschouwen de nullen en enen niet als getallen, maar als tekens (dus `char` in C++). Ze moeten in deze vraag dan ook als zodanig behandeld worden.

a. (6 punten)
Geef een eenvoudig (brute force) iteratief algoritme dat `true` oplevert als er in het array A ergens twee nullen naast elkaar voorkomen, en `false` als dat niet het geval is. Schrijf hiervoor een C++-functie `bool bevat00(char` A`[ ], int` n`)`.

b. (8 punten)
Geef een decrease-by-one algoritme voor bovenstaand probleem. Schrijf daartoe een *recursieve* C++-functie `bool dubbelnul(char` A`[ ], int` i`)`, die het probleem oplost voor het (deel)array $A[0], A[1], \ldots, A[i]$ ($i \geq 1$). De aanroep `return dubbelnul(`$A, n-1$`);` geeft dan uiteraard het antwoord voor het hele array.

c. (4 punten)
Vergelijk het aantal tests dat de algoritmen uit **a.** en **b.** doen in het slechtste geval. We bedoelen hierbij de tests waarbij $A[j]$'s betrokken zijn, zoals een test of $A[j]$ gelijk is aan 0. Welk van de twee algoritmen heeft dan de voorkeur en waarom?

d. (10 punten)
Neem aan dat n een 2-macht is. Geef nu een divide-and-conquer algoritme dat het probleem oplost. Het array dient hiervoor in twee gelijke delen te worden verdeeld.
Schrijf een *recursieve* C++-functie `bool nulnul(char` A`[ ], int links, int rechts)` die het probleem oplost voor het deelarray A[`links`], ..., A[`rechts`] ter lengte een 2-macht.

-vervolg op pagina 4

Opgave 5. (25 punten)
In de vijfde eeuw voor Christus was het in de stadstaat Sparta gebruikelijk dat de regering (een raad van oude mannen) behalve over regeringszaken en geloofszaken ook besliste over wie met wie moest trouwen[1]. Eens per jaar kwam de regering bijeen om n mannen en n vrouwen aan elkaar te koppelen. In een vergadering die wel een week kon duren werd uitgebreid overlegd over hoe goed elk mogelijk paar (vrouw, man) bij elkaar paste. Dit resulteerde in een tabel waarin elke combinatie (vrouw, man) een cijfer tussen 1 en 10 kreeg, dat aangaf hoe succesvol men die combinatie vond. Ten slotte werd uit de tabel de koppeling (toewijzing) met de hoogste totaalwaarde bepaald, en de volgende dag trouwden deze koppels met elkaar. Natuurlijk leefden zij daarna nog lang en gelukkig.

De bedoeling in deze vraag is dat wij het toewijzingsprobleem van de Spartanen oplossen. Gegeven n vrouwen en n mannen en een tabel `succes` waarin `succes`$[i][j]$ aangeeft hoe goed vrouw i en man j bij elkaar passen. De bedoeling is nu om een koppeling (toewijzing) te vinden van de mannen aan de vrouwen (één man per vrouw en één vrouw per man) met maximale totale succeswaarde.

Voorbeeld:

	Menelaos	Nestor	Odysseus	Patroclos
Antigone	3	7	5	8
Berenice	7	4	5	9
Cassandra	6	8	4	7
Demeter	5	9	7	6

De koppeling AN, BP, CM, DO heeft totale succeswaarde 29. Dit is *niet* maximaal.

a. (10 punten)
Leg uit hoe best-fit-first branch and bound werkt voor maximalisatieproblemen in het algemeen. Geef daarbij duidelijk aan hoe (deel)oplossingen gegenereerd worden, wat met branch bedoeld wordt en wat met bound, wat de betekenis van de bound is, wat best-fit-first betekent, wanneer gesnoeid wordt en waarom dat mag.

b. (5 punten)
Beschrijf hoe je best-fit-first branch and bound concreet gebruikt om bovenstaand probleem op te lossen, dus om een optimale koppeling te vinden van n vrouwen aan n mannen. Leg dus uit hoe je in dit geval je deeloplossingen genereert en wat voor afschatting je gebruikt voor deeloplossingen en waarom die geldig is.

c. (10 punten)
Pas je branch and bound algoritme zoals beschreven in **b.** toe op het voorbeeld en teken de bijbehorende state-space-tree. Geef daarin aan in welke volgorde de knopen bekeken worden en welke deeloplossingen gesnoeid worden en waarom.
Leg ook uit hoe jouw oplossing gelezen moet worden (legenda).

[1] Dit is echt waar; het waarheidsgehalte van de rest van de opgave is twijfelachtig.