Tentamen
ORIENTATIE INFORMATICA
24 augustus 2006
09.00 – 12.00

Opmerkingen vooraf:

- geef bij elke Haskell-functie ook de typering.
- in elk onderdeel mag je gebruik maken van vorige onderdelen, ook als je niet in staat bent geweest deze te maken.
- Niet elke opgave telt even zwaar. Bij elk onderdeel staat aangegeven hoeveel punten er maximaal gescoord kunnen worden. Bij 100 of meer punten is het tentamenresultaat een 10. In andere gevallen is het tentamenresultaat het aantal punten gedeeld door 10.
- Studenten die de 3ec-variant van dit vak willen doen, dienen dit expliciet bovenaan het eerste in te leveren blad te vermelden. Een score van 50 punten levert voor hen de beoordeling 10.
  De items 1 tot en met 16 horen bij deze versie van het vak, maar het is toegestaan ook andere items uit te werken om op die manier de score te verhogen.

Opgave 1

Deze opgave gaat over *bitstrings*, dat wil zeggen strings bestaande uit symbolen `'0'` en `'1'`. Een aantal van deze bitstrings zijn coderingen van boodschappen. Ter onderscheiding van willekeurige bitstrings noemen we de bitstrings die coderingen zijn van boodschappen *woorden*.

Onder de *Hamming distance* tussen twee bitstrings `x` en `y` van gelijke lengte verstaan het aantal posities waarin `x` en `y` verschillen. Bijvoorbeeld: met

```
x = "011010110"
y = "111010100"
z = "000011111"
```

is de Hamming distance tussen `x` en `y` gelijk aan 2.

[2 pt] ☐ 1. Bepaal voor `x`, `y` en `z` uit bovenstaand voorbeeld de Hamming distance tussen zowel `x` en `z` als tussen `y` en `z`.

[3 pt] ☐ 2. Geef een Haskell-functie die bij twee bitstrings van gelijke lengte de Hamming distance oplevert.
NB: een string is hetzelfde als een lijst van karakters.
Dus: `"01101"='0':"1101"=['0','1','1','0','1']`

[3 pt] ☐ 3. Wat weet je van de strings `u` en `v` (van gelijke lengte) als de Hamming distance nul is?

lees verder

Onder een *woordenboek* verstaan we een opsomming van (code-)woorden en de bijbehorende boodschappen. In Haskell representeren we zo'n woordenboek als een lijst van paren van strings. Bijvoorbeeld:

```
[ ("0000111", "aap"),  ("0001110", "noot"),  ("0010100", "mies"),
  ("0011001", "wim"),  ("0100001", "zus"),   ("0101100", "jet") ]
```

Het eerste lid van zo'n paar is het (code-)woord; het tweede lid is de boodschap. Je mag er van uit gaan dat alle codewoorden in zo'n woordenboek de zelfde lengte hebben.

[4 pt] ☐ 4. Geef een Haskell-functie die bij een bitstring `x` en een woordenboek `lst` de bij `x` behorende boodschap oplevert als `x` als codewoord in het woordenboek voorkomt en anders de string `"onbekend"`.
Je mag aannemen dat de lengte van `x` gelijk is aan de lengte van de codewoorden in het woordenboek.

Een mogelijke strategie bij het decoderen van ontvangen bitstrings is het volgende: lever als resultaat een boodschap waarvan de codering minimale Hamming distance heeft tot de ontvangen bitstring.

[2 pt] ☐ 5. Bepaal de boodschap die bij deze strategie zal worden opgeleverd bij de string `"0001001"` en het woordenboek uit het vorige voorbeeld. Laat duidelijk zien hoe je aan je antwoord komt.

[4 pt] ☐ 6. Geef een Haskell-functie die bij een bitstring `x` en een woordenboek `lst` een paar `(b, n)` oplevert. Hierin is `b` een boodschap waarvan het codewoord minimale Hamming distance heeft tot `x` en is `n` deze minimale afstand.

[2 pt] ☐ 7. Geef een Haskell-functie die bij een bitstring en een woordenboek een boodschap oplevert volgens bovenbeschreven strategie.

[4 pt] ☐ 8. Waar moet de Hamming distance tussen de codewoorden in het woordenboek aan voldoen opdat deze strategie een één-fout-corrigerende methode is? Beargumenteer duidelijk je antwoord.

Opgave 2

Deze opgave gaat over (ongerichte) grafen. Een graaf bestaat uit een verzameling punten of knopen en een verzameling takken of kanten. We nemen aan dat de knopen zijn gelabeld met gehele getallen; een tak wordt gerepresenteerd door een paar van gehele getallen: de labels van de eindpunten van de tak. Daarbij nemen we aan dat in de representatie van een tak de laagst genummerde knoop vooraan staat.
De graaf representeren we door een lijst van takken. In deze lijst komen geen duplicaten voor. Verder nemen we aan dat de graaf geen 'oren' heeft, dat wil zeggen dat er geen tak is van een knoop naar zichzelf.
Voorbeeld:

De graaf hiernaast kan worden gerepresenteerd door de lijst

```
[ (1,2), (1,6), (2,3),
  (2,4), (2,5), (2,6),
  (3,4), (4,5), (5,6) ]
```

[3 pt] □ 9. Eerst een vraag om er in te komen. Onder de graad van een knoop in een graaf verstaan we het aantal takken dat in die knoop samenkomt. Zo heeft in het voorbeeld hierboven de knoop 3 graad 2 en heeft knoop 2 graad 5.
Geef een Haskell functie **graad** die bij een knoop `k` en een graaf `g` de graad van `k` in `g` oplevert.

Onder een *onafhankelijke verzameling* in een graaf verstaan we een verzameling knopen waarvan geen enkel tweetal door een tak met elkaar is verbonden. In het voorbeeld hierboven is `{1, 3, 5}` een onafhankelijke verzameling en ook `{1, 4}`. Maar `{1, 3, 4}` is geen onafhankelijke verzameling.

We ontwikkelen nu een aantal (hulp-)functies om verderop een functie te maken die van een verzameling knopen in een graaf oplevert of dit een onafhankelijke verzameling is. Een verzameling knopen representeren we door een lijst. Zo'n lijst heeft geen duplicaten.

[3 pt] □ 10. Geef een Haskell functie

```
zitIn :: (Integer, Integer) -> [(Integer, Integer)] -> Bool
```

die bij een paar `(a,b)` (met `a < b`) en een graaf `g` oplevert of `(a,b)` een tak is in `g`.

[5 pt] □ 11. Geef een Haskell functie `disjunct` die bij twee grafen `g` en `h` oplevert of ze geen enkele tak gemeen hebben.

[4 pt] □ 12. Beredeneer wat de worst-case tijdcomplexiteit is van de functie `disjunct`. Formuleer je antwoord in termen van de lengtes van de twee lijsten.

lees verder

[3 pt] ☐ 13. Geef een Haskell functie

```
verbind :: Integer -> [Integer] -> [(Integer, Integer)]
```

die bij een knoop `k` en een verzameling `lst` van knopen de graaf oplevert van takken tussen `k` en elk van de knopen uit `lst`. Je mag er daarbij vanuit gaan dat `k` niet voorkomt in `lst`.

[5 pt] ☐ 14. Geef een Haskell functie `kliek` die bij een verzameling knopen de graaf oplevert waarin elk tweetal knopen uit de verzameling met elkaar is verbonden door een tak.

[4 pt] ☐ 15. Beredeneer wat de tijdcomplexiteit is van de functie `kliek`. Formuleer je antwoord in termen van de lengte van de lijst.

[3 pt] ☐ 16. Geef een Haskell functie

```
isOnafhankelijk :: [Integer] -> [(Integer, Integer)] -> Bool
```

die bij een verzameling `v` en een graaf `g` oplevert of `v` een onafhankelijke verzameling is in `g`. Je mag er daarbij vanuit gaan dat de elementen van `v` inderdaad knopen zijn in `g`.

[Aantal punten tot hier: 54]

Bekijk nu het volgende beslissingsprobleem

Onafhankelijke verzameling (INDEPSET)
Parameter: een graaf G en een positief geheel getal K
Gevraagd: heeft G een onafhankelijke verzameling met K elementen?

[3 pt] ☐ 17. Leg uit wat we verstaan onder een beslissingsprobleem.

[3 pt] ☐ 18. Geef een `ja`-instantie van (**INDEPSET**)

[3 pt] ☐ 19. Geef een `nee`-instantie van (**INDEPSET**)

[5 pt] ☐ 20. Beargumenteer dat (**INDEPSET**) $\in$ **NP**.

Er valt zelfs te bewijzen dat (**INDEPSET**) $\in$ **NPC** door gebruik te maken van het feit dat (**VC**) $\in$ **NPC**.

[3 pt] ☐ 21. Leg uit wat we verstaan onder de klasse **NPC** .

[4 pt] ☐ 22. In welke richting moet de reductie gaan? Beargumenteer je antwoord.

[5 pt] ☐ 23. Geef aan wat je bewijsverplichtingen zijn als je de NP-volledigheid van (**INDEPSET**) wilt bewijzen op grond van het feit dat (**VC**) NP-volledig is.

Opgave 3

Gegeven is de volgende grammatica:

```
<S>  ::=   'a'<A><B> | 'c'
<A>  ::=   'a'<S>
<B>  ::=   'a'<B> | 'b'<C>
<C>  ::=   'a'<C> | <C>'a' | 'b'
```

Het symbool <S> is het startsymbool.

[4 pt] □ 24. Beschrijf nauwkeurig de strings die kunnen worden afgeleid uit het hulpsymbool <C>.

[4 pt] □ 25. Teken een eindige automaat die van de invoer bepaalt of ze kan worden afgeleid uit het hulpsymbool <B>.

[5 pt] □ 26. Geef een afleidingsboom (parse tree) bij de string **aaaacabaaabbaba**.

[4 pt] □ 27. Is de grammatica ambigu? Bewijs je bewering.

[5 pt] □ 28. Beschrijf nauwkeurig de strings die in deze grammatica afgeleid kunnen worden.

[4 pt] □ 29. Is er een Turingmachine die van de invoer nagaat of ze kan worden afgeleid uit het startsymbool <S>? Beargumenteer je bewering.

einde

totaal: 106 punten.