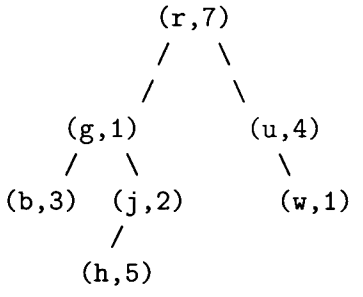
Tentamen
ORIENTATIE INFORMATICA
12 augustus 2002
09.00 – 12.00 uur; Tentamenhal

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.

Opgave 1

Deze opgave gaat over gewogen binaire bomen. Dat betekent dat er in een knoop twee data-elementen zijn opgenomen. Ten eerste de 'key' die hoort bij de knoop; ten tweede de frequentie bij deze sleutelwaarde. Om de kans op verwarring wat kleiner te maken, nemen we aan dat de keys karakters zijn. De frequenties zijn *positieve* natuurlijke getallen. Voorbeeld:



NB: Terwille van de leesbaarheid zijn in dit voorbeeld de quotes weggelaten.

In Haskell maken we gebruik van de volgende datadefinitie om bovenstaande structuur in te representeren:

```
data Tree = Nil | Node Char Integer Tree Tree
```

Verder veronderstellen we dat alle sleutelwaarden in de boom uniek zijn. Dat wil zeggen dat elk karakter hooguit eenmaal als sleutelwaarde in de boom voorkomt.

[4 pt] ☐ 1. Geef de boom uit het voorbeeld weer in deze Haskell-notatie.

[5 pt] ☐ 2. Geef een Haskell-implementatie voor de functie die bij een karakter `c` en een boom `t` de frequentie van `c` in `t` oplevert. Komt `c` niet voor in `t`, dan is het resultaat 0.

Het toevoegen van data aan deze datastructuur is lastig als er verder geen ordening in de boom zit. Daarom veronderstellen we vanaf nu, dat de boom tevens een binaire zoekboom is.

[4 pt] ☐ 3. Geef de definitie van een binaire zoekboom.

[6 pt] ☐ 4. Geef een Haskell-implementatie van de functie

```
addChar :: Char -> Integer -> Tree -> Tree
```

zo, dat `(add c k t)` de boom oplevert die ontstaat door `k` voorkomens van `c` toe te voegen aan `t`.

[5 pt] ☐ 5. Geef een Haskell-implementatie van een functie die twee gewogen binaire zoekbomen samenvoegt tot één gewogen binaire zoekboom.

lees verder

Opgave 2

Deze opgave gaat over (eindige) verzamelingen van natuurlijke getallen en over collecties van dergelijke verzamelingen. Een verzameling van natuurlijke getallen representeren we in Haskell met behulp van een (ongeordende) lijst. Zo ook zullen collecties van verzamelingen worden gerepresenteerd door lijsten van lijsten.

Aangezien een verzameling geen duplicaten bevat, mag je er vanuit gaan dat de lijsten ook geen duplicaten bevatten.

[4 pt] □ 6. Geef een Haskell-implementatie van de functie

```
isElt :: Integer -> [Integer] -> Bool
```

die bij een getal `x` en een lijst `lst` oplevert of `x` voorkomt in `lst`.

[5 pt] □ 7. Geef een Haskell-implementatie van de functie

```
intersects :: [Integer] -> [Integer] -> Bool
```

die bij twee lijsten oplevert of ze gemeenschappelijke elementen hebben.

[6 pt] □ 8. Wat is de worst-case tijdcomplexiteit van deze implementatie van `intersects`? Beargumenteer duidelijk je antwoord; we vragen geen formeel bewijs.

Laat C een collectie van verzamelingen zijn. We noemen een verzameling H die met elk element van C minstens één element gemeen heeft een *Hittingset* voor C. Bijvoorbeeld: met

$$C = \{\ \{1,2,3,4\},\ \{1,3,5\},\ \{2,6,9\},\ \{6,7,8,9\}\ \}$$

is $H = \{1,2,8\}$ een Hittingset voor C.

[5 pt] □ 9. Geef een Haskell-implementatie van een functie `isHittingset` die bij een lijst `h` en een lijst van lijsten `coll` oplevert of `h` een hittingset voor `coll` representeert.

Bekijk nu het volgende optimaliseringsprobleem:

Minimal Hittingset (MinHitSet)
Parameter: een eindige verzameling V en een collectie C van deelverzamelingen van V
Gevraagd: een kleinste hittingset $H \subseteq V$ voor C

[5 pt] □ 10. Beschrijf een algoritme voor (**MinHitSet**).
LET OP: we vragen niet om dit in Haskell uit te werken, maar om een globale beschrijving van het algoritme!

[4 pt] □ 11. Geef een schatting van de tijdcomplexiteit van je algoritme. Ook hier vragen we weer naar een goede argumentatie, geen formeel bewijs.

lees verder

Bekijk nu het beslissingsprobleem dat verwant is aan het bovenstaande optimaliseringsprobleem:

Hittingset (HitSet)
Parameter: een eindige verzameling V en een collectie C van deelverzamelingen van V en een positief geheel getal K
Gevraagd: is er een hittingset H voor C met hoogstens K elementen?

[4 pt] ☐ 12. Geef een ja-instantie voor (**HitSet**).
[4 pt] ☐ 13. Geef een nee-instantie voor (**HitSet**).
[5 pt] ☐ 14. Beargumenteer dat (**HitSet**) $\in$ **NP**.

Gebruikmakend van het feit dat (**SAT**) $\in$ **NPC** is er zelfs te bewijzen dat (**HitSet**) $\in$ **NPC**.

[4 pt] ☐ 15. Leg uit wat we verstaan onder de klasse **NPC**.
[6 pt] ☐ 16. Geef aan in welke richting de reductie moet gaan en wat de overige bewijsverplichtingen zijn.

Opgave 3

Gegeven is de volgende grammatica:

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

Het symbool `<S>` is het startsymbool.

[4 pt] ☐ 17. Beschrijf nauwkeurig de strings die kunnen worden afgeleid uit het hulpsymbool `<A>`.
[4 pt] ☐ 18. Teken een eindige automaat die van de invoer bepaalt of ze kan worden afgeleid uit het hulpsymbool `<B>`.
[4 pt] ☐ 19. Geef een afleidingsboom (parse tree) bij de string `bbabacbaaabbab`.
[4 pt] ☐ 20. Is de grammatica ambigu? Bewijs je bewering.
[5 pt] ☐ 21. Beschrijf nauwkeurig de strings die in deze grammatica afgeleid kunnen worden.
[6 pt] ☐ 22. Is er een eindige automaat die van de invoer nagaat of ze kan worden afgeleid uit het startsymbool `<S>`? Bewijs je bewering.
[4 pt] ☐ 23. Is er een Turingmachine die van de invoer nagaat of ze kan worden afgeleid uit het startsymbool `<S>`? Beargumenteer je bewering.

einde

totaal: 107 punten.