Adres: Rijksstraat 2
Postcode en: 9725 EJ
Woonplaats: Groningen

Studierichting: InGmetica

Jaar van eerste inschrijving: 2003

Toetsnaam: Oriëntatie InGmetica
Datum: 27-10-2003
Naam docent: Bolduer

1

1

x	0	1	1	0	1	0	1	1	0	
z	0	0	0	0	1	1	1	1	1	

y	1	1	1	0	1	0	1	0	
z	0	0	0	0	1	1	1	1	1

2

Tussen x en z: 4 Tussen y en z: 6 ✓

2 `hammingDistance :: [Char] -> [Char] -> Integer`

3
```
hammingDistance [] [] = 0
hammingDistance (a:lst1) (b:lst2)
  | a == b  = (hammingDistance lst1 lst2)
  | otherwise = 1 + (hammingDistance lst1 lst2)
```
✓

3 3 Dan zijn alle bits aan elkaar gelijk, dus de strings zijn gelijk. ✓

4 `findInDictionary :: [Char] -> [([Char], [Char])] -> [Char]`

1
```
findInDictionary a [] = "onbekend"
findInDictionary a ((bitstring, woorde) : lst)
  |(hammingDistance a bitstring) == 0  = woorde
  |otherwise                           = (findInDictionary a lst)
```
✓

5

woord	Goedschap	hamming-distance
0000111	cap	3
0011001	wm	1
0001110	mod	3
0100001	zus	2
0110100	mies	~~3~~ 4
0101100	jet	3

2 We zien dat de Goedschap "wm" de minimale hamming-distance met de string. Deze Goedschap zal dan ook worde teruggegeven

6 `findBestInDictionary :: [Char] -> [([Char], [Char])] -> ([Char], Integer)`

~~findBestInDictionary str ((bs, wrd):lst) = (wrd, (hammingDistance str bs))~~

4
```
findBestInDictionary str ((bs, wrd) : lst)
  |a<=b    = (wrd, a)
  |otherwise = (bstr, b)
    where a = (hammingDistance str bs)
          (bstr, b) = (findBestInDictionary str lst)
```
✓

(1) [illegible] = [illegible]

where (msg, dist) = (findBestInDictionary str dict)

8 Wil deze strategie werken voor het corrigeren van één bit, dan moet het zo zijn dat voor elke mogelijke string die daarnaast één bit van de woordenboekwoord

(4) verschilt, dat ~~woordenboek~~ het eigen woord ~~woordenboek~~ is dat maar één bit van de string scheelt. ~~[illegible]~~

Alle woordenboekwoorden die nu één bit v. de string schelen, schelen automatisch twee bits van het matchende woordenboekwoord, en zijn dus verboden. Conclusie: alle woordenboekwoorden mogen onderling niet ~~[illegible]~~ twee bits verschillen. Eén bit mag ook niet, want dan kan een efout woord zelfs als helemaal goed worden gezien. Alles moet dus méér dan 2 bits verschillen.

(2) 9

```
graad :: Integer -> [(Integer, Integer)]
graad x [] = 0
graad x ((a,b):lst) =
  | a == x || b == x  = (graad x lst) + 1
  | otherwise         = (graad x lst)
```

3

10

```
zitIn :: (Integer, Integer) -> [(Integer, Integer)] -> Bool
zitIn x [] = False
zitIn (c,d) ((a,b):lst)
  | c == a && d == b  = True
  | otherwise         = (zitIn (c,d) lst)
```

11

```
disjunct :: [(Integer, Integer)] -> [(Integer, Integer)] -> Bool
disjunct [] x = True
disjunct (p:lst) x
  | (zitIn p x) = False
  | otherwise   = (disjunct lst x)
```

5

12 We bepalen eerst de worst-case tijdcomplexiteit van zitIn, uitgedrukt in het aantal elementen van de graaf, en gemeten in het aantal ((c == a) && (d == b)) vergelijkingen. ~~[illegible]~~ Noem dit $C_z(n)$. Dan geldt in het beroerdste geval dat het element niet wordt gevonden, en zitIn de hele tijd recursief wordt aangeroepen. Maar dan geldt:

$$\left.\begin{aligned} C_z(0) &= 0 \\ C_z(n) &= C_z(n-1) + 1 \quad (n \neq 0) \end{aligned}\right\} \Rightarrow C_z(n) = n$$

Kijken we naar de functie disjunct, dan ~~[illegible]~~ zullen we de tijdcomplexiteit v. deze functie, gemeten in het aantal ~~[illegible]~~ vergelijkingen in zitIn, $C_d(m,n)$ noemen met $m = \#g$, $n = \#h$. Met een beetje pech zijn de grafen inderdaad disjunct en moeten we voor elke element in m zitIn gaan aanroepen. Er geldt nu:

Adres: Rijnstraat 1 | Studierichting: Informatica | Tentamen: Discrete Wiskunde
Postcode en 9725 ET | Datum: 27-10-2003
Woonplaats: Groningen | Jaar van eerste inschrijving: 2003 | Naam docent: Bakker

4
$$\left.\begin{array}{l} Cd(0,n) = 0 \\ Cd(m,n) = C_2(n) + Cd(m-1,n) \quad (m \neq 0) \\ \quad = n + Cd(m-1,n) \end{array}\right\} Cd(m,n) = m * n.$$

Als we zeggen x := max(m,n) dan zie we dat de complexiteit van Cd, uitgedrukt in x, $O(x^2)$ is. :)

13
```
verbind :: Integer -> [Integer] -> [(Integer, Integer)]
verbind x [] = []
verbind x (a:lst)
 | x<a = (x,a):(verbind x lst)
 | otherwise = (a,x):(verbind x lst)
```
3

14
```
kliek :: [Integer] -> [(Integer, Integer)]
kliek [] = []
kliek (a:lst) = (verbind a (lst)) ++ (kliek lst)
```
3

15
Eerst kijken we naar de tijdcomplexiteit van verbind, $C_v(n)$, waar n = het aantal elementen van de lijst die langs gaan, gemeten in het aantal "x<a"-vergelijkingen. Welnu, de enige regel zoekt, onafhankelijk van het resultaat van "x<a", en weer verder met de lijst minus één element. Er geldt:

$$\left.\begin{array}{l} C_v(0) = 0 \\ C_v(n) = 1 + C_v(n-1) \quad (n \neq 0) \end{array}\right\} C_v(n) = n$$

4
De tijdcomplexiteit van klieh, $C_k(n)$ met n = de lengte v. de lijst, gemeten in het getal "x<a" vergelijkingen in verbind, vinden we op dezelfde manier:

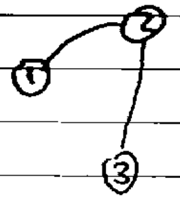
$$\left.\begin{array}{l} C_k(0) = 0 \\ C_k(n) = C_v(n-1) + C_k(n-1) \quad (n \neq 0) \\ \quad = n-1 + C_k(n-1) \end{array}\right\} \begin{array}{l} C_k(0) = 0 \\ C_k(n) = 0 + 1 + 2 + \ldots + (n-1) \\ \quad = \frac{1}{2} \cdot n(n-1) \quad (n>0) \end{array}$$

Conclusie: $C_k(n)$ is $O(n^2)$.

16
```
isOnafhankelijk :: [Integer] -> [(Integer, Integer)] -> Bool
isOnafhankelijk lst graaf = (disjunct (klieh lst) graaf)
```
3

18 $G = [(1,2),(2,3)]$

$k = 2$



3

Dit is een ja-instantie, want neem bijvoorbeeld de verzameling $\{1,3\}$. Deze is onafhankelijk en heeft precies k elementen.

19 $G = [(1,2),(2,3)]$

$k = 3$

3

De enige verzameling uit G met 3 elementen is $[1,2,3]$. Deze verzameling is niet onafhankelijk (immers, 1 is verbonden met 2, en 2 met 3). Daarom is dit een nee-instantie.

20 We weten dat (INDEPSET) $\in$ NP $\Leftrightarrow$ er is een certificaat voor (INDEPSET) dat in polynomiale tijd te controleren is. Welnu, het toegpl wil dat we net een functie hebben geschreven die inderdaad zo'n controle uitvoert. Immers, neem als certificaat een verzameling punten uit de graaf G met precies k elementen. Ons controleert de functie isOnafhankelijk of deze verzameling onafhankelijk is. Deze functie bestaat uit twee functieaanroepen, waarvan we eerder hebben laten zien, dat ze van polynomiale tijdcomplexiteit zijn. Onze conclusie luidt dus dat (INDEPSET) $\in$ NP.

3 V [illegible]

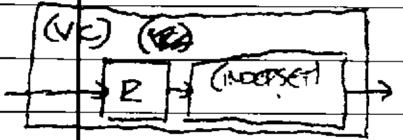
21 Beschouwing: voor elk beslissingsprobleem $x \in$ NPC geldt:

* $x \in$ NP
* Voor elk beslissingsprobleem $y \in$ NP geldt $y \propto x$. (en wel in polyn. tijd)

3

22/23 Als we kunnen bewijzen dat (VC) $\propto$ (INDEPSET) dan geldt voor elke $x \in$ NP dat $x \propto$ (VC) $\Rightarrow x \propto$ (INDEPSET). Omdat we weten dat (INDEPSET) $\in$ NP, geldt dan automatisch dat (INDEPSET) $\in$ NPC. We maken dus een reductie:

4+5



Voor deze reductie moeten we dan aantonen dat:

* R zet elke invoer voor (VC) om in een invoer voor (INDEPSET)
* R is van polynomiale tijdcomplexiteit
* ja-instantie van (VC) $\Leftrightarrow$ ja-instantie van (INDEPSET).

Postcode en 975 GT
Woonplaats: Groningen

Studierichting Informatica

Jaar van eerste inschrijving: 2003

Tentamen Discrete Informatica

Datum: 27-10-2003

Naam docent: Boldeer

(3) 24

De strings uit C worden beschreven met de taal:

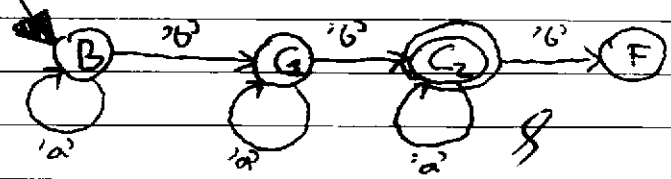
$$C = \{a^k b a^\ell \mid k, \ell \in \mathbb{N}\}$$

4 (N.B. $\mathbb{N} = \{0, 1, 2, \ldots\}$)

(immers, er komt één 'b' in voor e daarvoor a daarna de willekeurig veel 'a's)

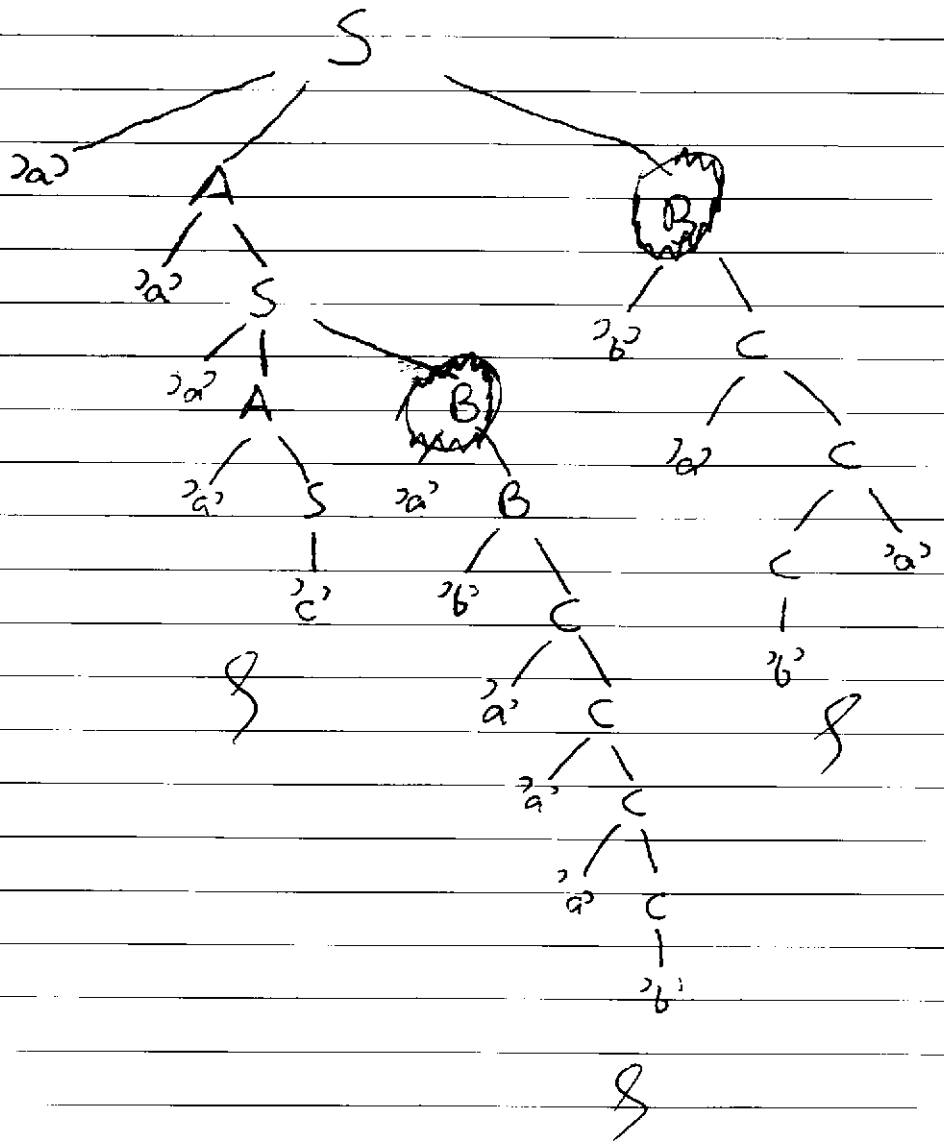
25

4



G2 is de accepteerde toestand.

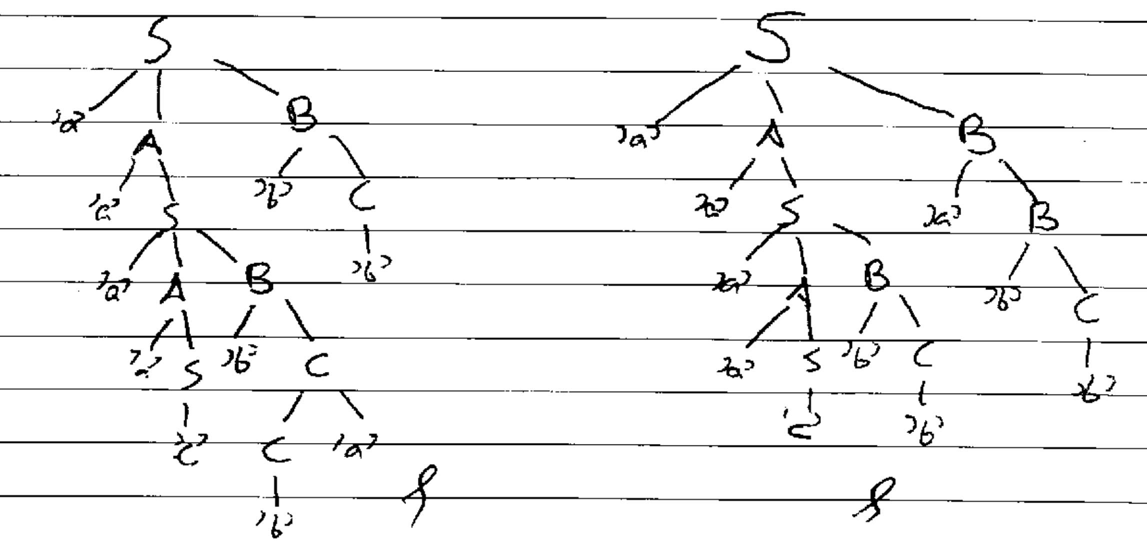
26



5

4

[illegible] voor deze string:



4

Voor deze string bestaan twee verschillende afleidingsbomen, dus de grammatica is ambigu. ✗

28 Als we naar S kijken, hebben we twee herleidingskeuzes. Als we 'a'<A><B> kiezen, herleidt dit weer via <A> tot S. We kunnen dus het aantal keer kiezen dat we <S> via 'a'<A><B> herleiden. Noem dit getal s. Voor elke s krijgen we twee a's erbij, één uit <S> en één uit, enz. Daarna volgt een 'c', gevolgd door een s-tal <B>'s. Dus

$L_S = $ [crossed out] $\{a^{2s} \cdot c \cdot \langle B \rangle^{s^2} \mid s \in \mathbb{N}\}$ ✗ (1)

Maar wat is B? B bestaat uit een willekeurig aantal 'a''s, gevolgd door een 'b', gevolgd door een <C>. Dus:

$L_B = \{a^{l} \cdot b \cdot \langle C \rangle \mid l \in \mathbb{N}\}$. (2)

5 C hadden we al eerder in een taal gevat. Invullen van <C> in (2) en van (2) in (1) geeft dan:

~~$S = \{a^{2s} \cdot c \cdot (a^{l} \cdot b \cdot a^{n} \cdot b \cdot a^{m})^{s} \mid s, l, n, m \in \mathbb{N}\}$~~

$L = \{a^{2s} \cdot c \cdot (a^{l_i} \cdot b \cdot a^{n_i} \cdot b \cdot a^{m_i})^{s} \mid s \in \mathbb{N}, \{l_i, n_i, m_i \in \mathbb{N} \mid i \in \{1, 2, \ldots, s\}\}\}$

0 29 Er is een stelling die zegt dat voor iedere grammatica een TM te bouwen is, dus i.h.b. geldt dat ook voor deze grammatica. Dit is dus inderdaad mogelijk.

je beschrijving hierboven is correct, maar deze notatie betekent iets anders dan je bedoelt. Waarom laat je het niet bij een beschrijving in woorden? Dat werd gevraagd.

Hebben we die behandeld? (Is namelijk niet waar)